Active Learning for Speaker Diarization

Beltrán Castro Gómez
Supervisors: Bertrand Higy, Hervé Bredin

Master in Machine Learning and Data Mining
Unniversité Jean Monnet
Saint-Étienne, France

June 2023

Contents

1 Introduction

Active learning or query-based learning is a subfield of machine learning which allows a supervised learning model to actively select which data it wants to learn from, aiming to obtain good performance with less training data [1].
Since not all kinds of data require the same effort to be labeled, active learning is notably helpful in those cases where labeling data is costly - either because it is tedious, time-consuming, economically expensive or even if it requires a lot of computational power or proficient knowledge. Domain-specific datasets (e.g. related to biomedical science or physics) usually require expertise in the topic to be complete and useful. Similarly, in the context of speech processing, labeling can be very hard since human annotators have to listen to pre-recorded data in order to generate annotations, which may also require a high degree of detail depending on the task.

The ultimate goal of active learning is to find the most *informative* examples in a given dataset so that our model can learn quicker and achieve a good generalization while needing fewer data than in the normal scenario. By looking for the best examples to learn from, we try to break power law scaling, which has been empirically observed in different machine learning applications when learning from randomly selected data, in favour of exponential scaling [2]. To assess whether an example is informative or not, however, can be quite challenging, since we do not have a clear definition of *informativeness* and it strongly depends on the kind of measure that we use for evaluating it.

In an active learning setup, a common starting point is to begin with a very small training set and a large pool of unlabeled data - or even a pretrained model and just unlabeled data. From there on, we train a model on this initial training set (if available) and then use a sampling technique to try to find the most useful or informative examples, label them, and include them in our training set.
Consequently, we can retrain our model on the new versions of the training set and iteratively repeat this process until we have done it a sufficient number of times or until our model reaches a certain target performance.

While manual annotation of unlabeled data poses a limitation to modern deep learning, active learning has been growing in popularity in the past years, specially motivated by

1. the natural growth in the amount of data needed by supervised models for being able to keep up the good performance as they evolved into more complex architectures and

2. the availability of vast pools of unlabeled data for certain tasks (i.e. internet stock images for computer vision or social media texts for natural language processing).

2 Context of the project at Ava

Ava is a French-American technology startup founded in 2014 with the goal of empowering deaf and hard-of-hearing people to fully accessible communication. It offers a mobile and desktop app solution that provides:

1. Online sub-titling of single or multi-speaker conversations (either in person or remote) or of any web media (Youtube videos, podcasts, etc) and

2. Further information to help understand the previous better (e.g. by tagging *who spoke when?*).

In order to achieve this, Ava system makes use of modern speech diarization technologies. In addition, the company has access to large amounts of unlabeled audio data, suitable for training and/or evaluating diarization models to tackle the who spoke when problem. However, annotating it is very costly and time-consuming. One potential way of optimizing an annotation pipeline is the use of active learning.
Still, neural networks present difficulties when adapted to an active learning setup in practice. Active learning approaches that use classical machine learning models usually rely on learning from small amounts of data, labeling a few data points or even just one example at a time and then update a given model with them. This is not very compatible with deep models, which need large amounts of data to achieve good performance.
So, it is important to find a trade-off in the size of the data batches that we want to annotate. It would be nonsense and costly to retrain a neural network every time we obtain new labels for an example. On the other hand, waiting for all of the available data to be annotated or annotate random batches of data would be equivalent to not using active learning at all.
The most appropriate way of handling this would be to use active learning sampling techniques to select data in batches of 2 to 5 hours of audio, so that when we get their annotations it is sufficient to evaluate the improvement of a neural network's performance, but also we do not have to wait for weeks or months for having them.
The goal of this project is to adapt already-existing active learning strategies to speaker diarization and test their performance.

3 Survey of the state of the art

3.1 Speaker Diarization

Speaker diarization or speech diarization is the task of identifying who spoke when in an audio recording. Modern speaker diarization approaches can be divided into two main steps: speaker segmentation and speaker clustering.
Speaker segmentation, illustrated in Figure 1, consists in partitioning a conversation into speaker turns. It can also be divided into sub-tasks: voice activity detection, speaker change detection, and overlapped speech detection.

Speaker clustering consists in grouping speech segments produced by one same speaker during a conversation. It is done by generating embeddings for each speech segment and then comparing them.

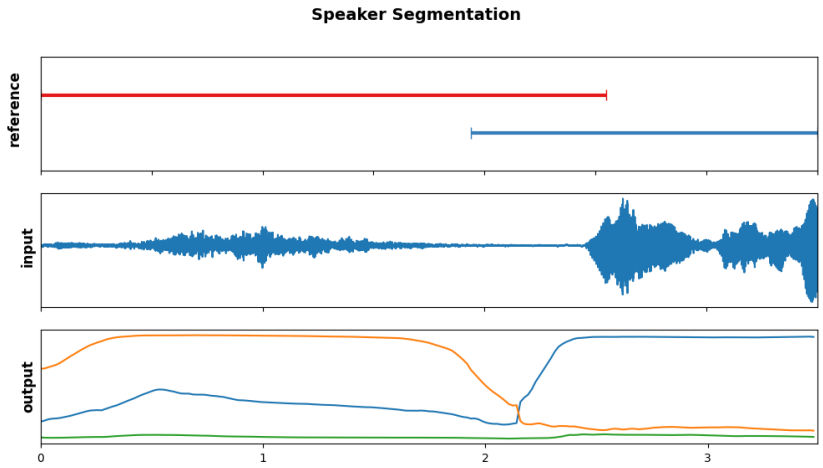
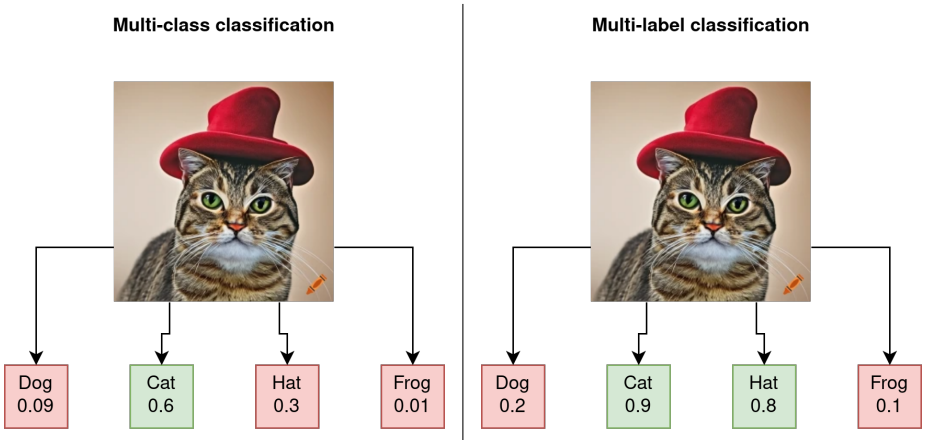


Figure 1: Example of speaker segmentation for a 3.5 seconds segment. First row shows reference annotations, that correspond to segments where a speaker is speaking and their identity, in this case visualized with different colors. Second row corresponds to the waveform input and third row shows the model's speaker activity output probabilities.

The first speaker diarization approaches started to appear in the 1990s, and during that decade they were mainly addressing tasks related to radio and television broadcast news, air traffic control or call center telephonic conversations. These early models relied on signal processing for extracting acoustic features and on classical machine learning methods like clustering, Hidden Markov Models and Gaussian Mixture Models [3]. Conventional diarization systems were composed of several sub-modules and usually presented architectures divided into multiple stages, among which we can find pre-processing steps, feature extraction, speaker representation, voice activity detection, segmentation, speaker clustering or post-processing steps.
The start of the 2000s brought the consolidation of research groups worldwide and the extension of diarization to more complicated contexts like meeting conversations, in which we can usually find a larger number of speakers [3].
Before the boom of deep models, most diarization approaches were clustering-based [4], relying on segmenting the audio into sufficiently small blocks or chunks to assume that there is only one speaker per block. Then, based on this assumption, the remaining steps are the computation of speaker embeddings for each block and their clustering. This idea has been shown to work well in general but has one main drawback - most of the errors made by this kind of systems

come from overlapped speech, which they are intrinsically unable to handle due to the unique speaker per chunk premise.
The revolution of deep learning during the 2010s also had an impact on speaker diarization. Initially, traditional models started to be progressively substituted by neural networks for tasks like speaker embeddings extraction, but later we also saw end-to-end neural-network-based diarization approaches appear [5]. These systems were conceived to tackle the speech overlap issue of cluster-based models.
Contrarily to modular approaches that divide diarization into several stages, end-to-end systems use a single neural network that takes frame-level spectral features as an input and directly produce the diarization outputs. This is possible by formulating speaker diarization as a multi-label classification problem and using permutation-invariant training.
In the standard single-label multi-class classification problem (see Figure 2a), we have one unique output class assigned to each example. Oppositely, in multi-label classification (see Figure 2b), it is possible to have more than one class assigned to one instance, or even none. For example, we may have an image containing both a cat and a hat[1], we could have another one showing all of the four entities, and even an image with none of them. Similarly, in speaker diarization we can have one or more speakers active at the same time, all of them, or not having speaker activity at all.



(a) Example of multi-class classification where the input image can only be assigned to one class.

(b) Example of multi-label classification where the input image can be assigned to more than one class.

Figure 2: Comparison of multi-class classification (left) with multi-label classification (right).

Permutation-invariant training (PIT) is needed to address the label ambiguity

[1]Image generated via Craiyon.

problem, which arises when formulating diarization in a multi-label classification setting since we cannot be sure that the order of the speakers in our model's output will be the same as the one in the ground truth. Since speaker s_1 may be active or not at timeframe t independently from speaker s_2, we could permutate the order of the speakers in a sequence and it would be still valid as a label, as shown in Table 1.
Therefore, in PIT the loss function has to be computed over all possible label permutations so as to find the one that minimizes the overall loss.

	s_1	s_2
t_1	0	1
t_2	0	1
t_3	1	1
t_4	0	0
t_5	1	0
t_6	0	1

	s_2	s_1
t_1	1	0
t_2	1	0
t_3	1	1
t_4	0	0
t_5	0	1
t_6	1	0

Table 1: Example of speaker activity labels that shows how permutation-invariant training may output the labels for a speaker in either of the nodes and still be valid.

Even though end-to-end could handle overlap more robustly than clustering approaches, it still had limited performance when dealing with long audios (more than 10 minutes) and situations with more than 3 speakers.

Current diarization technologies usually present an hybrid architecture in which they combine the strengths of both families of models: clustering-based and end-to-end approaches. One of this hybrid approaches is EEND-vector clustering [6, 7]. [8] also follows this approach but in an online fashion.
EEND-vector clustering method starts by segmenting audio inputs into homogeneous blocks. A sequence of frame features [2] are calculated for each block in order to serve as an input for the neural network. This network is trained to generate both diarization results and speaker embeddings of each block independently with an output size of S_{Local}, which refers to the maximum number of active speakers per block.

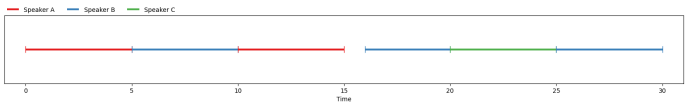


Figure 3: Example of reference annotations for a 30 seconds segment, containing 3 different speakers.

[2]Original EEND work uses log-scaled Mel-filterbank-based features.

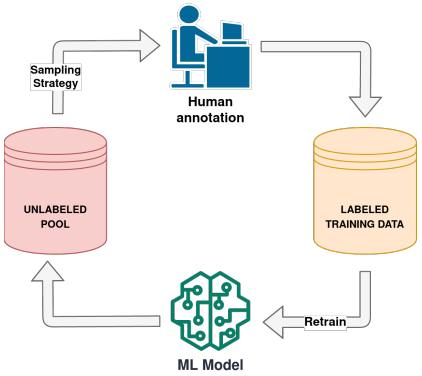
If we were to divide the audio input illustrated in Figure 3 into 15 second blocks, we may encounter the ambiguity of having Speaker B appearing in different nodes across blocks: it may be outputted in the first node for the first block and in the second node for the second one, or vice versa.
To solve this inter-block ambiguity, all of the speaker embeddings are clustered into S_{Global} clusters, where S_{Global} is a hyperparameter corresponding to the total number of speakers present in the full original input audio. Ideally, this would allow us to cluster the local embeddings corresponding to red, blue and green speech into three different clusters. By solving this ambiguity, it would be possible to finally regroup the speech turns of every speaker throughout the full audio.

3.2 Active learning

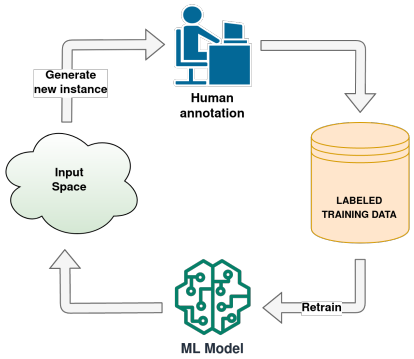
Active learning techniques have been investigated under three different kind of setups, depicted in Figure 4.
In pool-based sampling (see Figure 4a), the setup chosen for this project, we have a large pool of unlabeled data from where instances are sampled greedily by evaluating or scanning the entire pool in advance.
In membership query synthesis (see Figure 4b), the active learner requests labels for artificially generated instances that are expected to be the most informative for the model.
In stream-based sampling (see Figure 4c), instances are sampled from a distribution and the active learner decides whether or not to ask for their labels on a sequential manner.

All of the previous active learning scenarios require a sampling technique, which we can divide into two main groups: prediction-based sampling, that focuses on selecting samples that are challenging or informative for the model, and diversity sampling, which aims to select instances that are diverse or dissimilar from each other.
Within the first group, we can find some of the most studied and extended active learning strategies. One of them, uncertainty sampling, aims to select the instances whose labels the model is less certain about. There are several ways to interpret this said uncertainty in a multi-classification setting [9]:

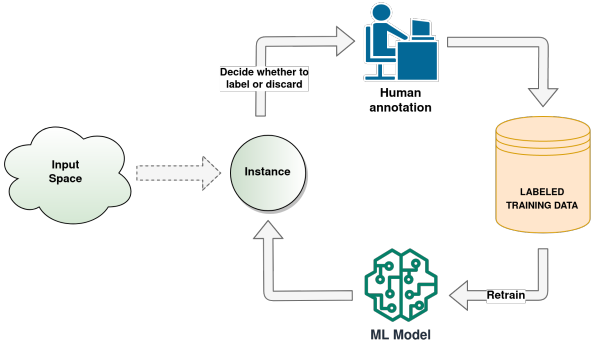
- **Least confidence**. Difference between the most confidently predicted label and 100% confidence.
- **Margin of confidence**. Difference between the first and second most confident predictions.
- **Ratio of confidence**. Ratio between the first and second most confident predictions.
- **Entropy**. Uncertainty expressed as entropy, same as in classical informa-



(a) Illustration of the pool-based learning loop.



(b) Illustration of the membership query synthesis learning cycle.



(c) Illustration of the stream-based learning setup.

Figure 4: The three main active learning setups: pool-based sampling (upper left), membership query synthesis (upper right) and stream-based sampling (bottom).

tion theory [10]:

$$entropy(x) = -\sum_{y \in Y} p(y|x)\, log\, p(y|x) \tag{1}$$

Another active learning technique that has been widely studied is Query-By-Committee, which relies on keeping a committee of models trained on the same data that span different areas of the hypothesis space. The different members of the committee can vote on query candidates and the queries are chosen based on their disagreement.

Expected Model Change is a family of techniques that share the idea of looking for the instances that would have the biggest impact on our model's parameters if we had their labels. One strategy of this kind is Expected Gradient Length or EGL, which makes use of the training gradient as a natural measure of *change*

or *informativeness* when using gradient descent optimization.

We defined prediction-based sampling techniques as those which aimed for selecting samples that would cause uncertainty, disagreement or change to our models. With the other group of strategies, diversity sampling, we seek to find gaps in our model's knowledge. Some popular approaches are [9]:

- **Model-based outlier sampling**. Looks for examples that are the most dissimilar to the model's understanding of the current training examples.
- **Cluster-based sampling**. A model-independent technique that aims to find diverse sets of examples through clustering.
- **Representative sampling**. Tries to find samples that accurately represent the broader underlying distribution of the data.
- **Real-world diversity sampling**. Focuses on sampling data to reduce the model's bias towards real-world demographics such as spoken language, nationality, age, gender or ethnicity.

4 Methodological contribution

In this project, we adapted several active learning sampling strategies to speaker diarization and compared them under a setup similar to a real-world use case.

4.1 Random baseline

Random sampling is not really an active learning technique since it just treats all examples equally, without any notion of informativeness or diversity. We use it as a baseline score to have a reference for comparing the performance of the different active learning sampling strategies, similarly to how it is done in most of the existing literature [1, 11, 12].
While it is the least computationally intensive and quickest sampling method, it is also expected to perform worse than or equal to the rest of the sampling strategies.

4.2 Uncertainty sampling

Even though uncertainty-based sampling is a classic active learning technique, a priori based on quite a simple idea - finding which examples our current model is less certain about - it has shown to be very competitive to this day, making it hard to be beaten by more novel and innovative active learning approaches.
Despite the simplicity of this strategy's premise, adapting it to our setup gave rise to two challenges.

4.2.1 Estimating predictions' confidence in deep models

To compute a score that would reflect the uncertainty of a model, we would need to somehow estimate the actual confidence in its predictions. Deep models, however, are not usually very good at representing it [13]. The softmax function, for instance, can be biased towards extreme values, outputting saturated probabilities close to 0 or 1. Training a model to maximize the probability of the expected class can lead, in general, to overconfident predictions.
One possible option to estimate confidence is to actually assume that the difference in the probabilities is still meaningful (even if saturated) and valid for ranking. So, we could directly use them to compute a confidence score. Nevertheless, there exist other more accurate estimations.
In [12], the authors integrate some of the recent advances in Bayesian deep learning to tackle the limitations of classical active learning: learning from small amounts of data and representing a model's uncertainty over unseen data. They present an active learning framework for high dimensional image data, which uses Bayesian Convolutional Neural Networks (BCNNs) [14]. These networks are CNNs with prior probability distributions over model parameters (θ), trained with dropout regularisation techniques so as to perform approximate inference. This allows to define a likelihood model (see Equation 2) suitable for sampling based on the network's uncertainty.

$$p(y|x,\theta) = softmax(f^{\theta}(x)) \tag{2}$$

In [13], the authors modify the architecture of a classifier to include a confidence estimation branch. Their network receives an input x and produces two outputs: softmax probabilities p and a confidence estimate c. The predictions p can get modified at training time based on c so as to push them closer to y and the classifier can learn to produce confidence estimates of its predictions.
In our work, what we do to estimate the confidence in a model's predictions is generate different versions of every audio example via data augmentation (i.e. adding background noise or music, passing the audio through filters). This allows us to obtain several inference outputs originally coming from one same example and by averaging these outputs we can produce an accumulated hypothesis. The reason to do this is because we consider that estimating the confidence of the model by measuring the level of disagreement on altered versions of one same example is more precise than just taking the raw predictions as estimates, similarly to the variational approximation done in [14] through dropout. Later in the Results section, we will discuss the outcomes of the uncertainty strategy both when using data augmentation estimates and without them, so we can have a better idea of what is their practical impact, and whether they are worth using or not.

4.2.2 Aggregation of confidence scores

The other challenge presented by the uncertainty strategy was to aggregate the uncertainty scores across speakers and across frames in a way that would represent properly the uncertainty of the model over the whole example.
The steps we followed for calculating the confidence scores with the augmentation technique are:

1. Generate the augmented versions of an audio file.
2. Run inference on all of the augmented audios and permutate the outputs to tackle the label ambiguity problem mentioned before.
3. Obtain an accumulated hypothesis by averaging the augmented versions as follows:

$$acc\_hypothesis = \frac{1}{n\_aug} \cdot \sum^{n\_aug} inf\_output_i \tag{3}$$

   where inf_output_i is the inference output for the i-th augmented audio version and n_aug the number of augmented versions.

4. Compute confidence score based on the distance to the 0.5 prediction with the formula below:

$$confidence = |acc\_hypothesis - 0.5| \cdot 2 \tag{4}$$

5. Aggregate the confidence score.
   5.1. Aggregate the confidence score across speakers.
   5.2. Aggregate the confidence score across frames.

In case we do not make use of augmentations, we would compute the confidence scores by performing steps 4 and 5 replacing *acc_hypothesis* by the model's inference output obtained from the original unaffected audio input.

We assumed speaker and temporal independence in the predictions, meaning that a given speaker s_1 could be active or not at timeframe t_1 with independence of what speaker s_2 is doing and also that the speaker activity of speaker s_1 at timeframe t_2 is not dependent of its speaker activity at timeframe t_1. These assumptions allow us to calculate the confidence of the predictions for each speaker s_i at every timeframe t_j independently and then aggregate them. We calculate confidence of a prediction in Equation 4 as the distance to 0.5 since we consider it to be the least confident prediction our model can give. As the prediction approaches 0, we know that the model is very certain that the speaker is not active and an output close to 1 tells us that the model is very certain that the speaker is active.
In terms of score aggregation, we first aggregate the scores across speakers to obtain a per-frame confidence score and finally we aggregate this scores to obtain

an overall instance-level confidence score. Any statistical aggregation method could work here, but we limited our experiments to *min*, *max* and *mean* aggregation.

4.3 Expected Gradient Length

The premise of Expected Model Change strategies is to find the examples that would make the largest impact to the current model parameters if their labels were available [1]. In the context of training with gradient descent, we have Expected Gradient Length (EGL) [15], which selects the samples that would impart the largest changes in the gradient of the objective function.
Since we lack the labels of our data, to compute the gradient length of an example we can only calculate an expectation over all its possible labels, as in Equation 5.

$$EGL(x) = \sum_{y \in Y} p(y|x, \theta) \, ||\nabla_\theta \mathcal{L}(x, y; \theta)||^2 \tag{5}$$

being x an audio example of T frames, y a sequence of labels of shape $T \times S$ where T is the number of frames and S the number of speakers and θ the model's parameters.
The marginalization of this score over all possible labels is usually a trivial task. For instance, in a multi-class classification setting like the one from Figure 2a, we would estimate the gradient length of every image as an expectation over the four possible labels (Dog, Cat, Hat, Frog). However, in speaker diarization, the number of possible labels grows as the number of speakers multiplied by the number of frames:

$$p(y|x, \theta) = \prod_{t=1}^{T} \prod_{s=0}^{S} p(y_{s,t}|x, \theta), \tag{6}$$

where $y_{s,t}$ is the label for speaker s at time t. This makes it very computationally expensive to try estimating the score considering all possible labels.
In [11], EGL is applied to a rather similar task, speech recognition. To alleviate this problem, they marginalize the gradient length over the K most probable sequences of labels. These are obtained by beam search decoding, like in [16]. We follow this approach of estimating the K most probable labels for a sequence through beam search while adapting it to multi-label classification.
Beam search (see Figure 5) estimates the K most likely sequence of labels for a given input by making use of class probabilities. It works by iteratively generating candidate sequences and keeping only the top-k best candidates or beams at every time step.

In speaker diarization, assuming voice activities of every speaker are independent from each other at time frame t, we have only two possible labels for each speaker s, which are active or non active. To compute the candidate probabilities at every step, we must consider all combinations of speaker activity, thus 2^S labelings, as illustrated in Figure 6. The probability of each labeling $p(y_t|x, \theta)$ is simply calculated as the product of the individual speaker probabilities.

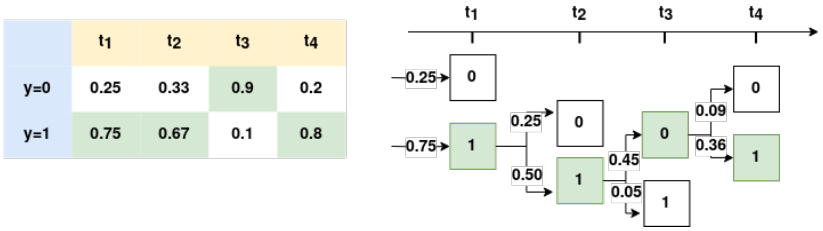


Figure 5: Example of beam search decoding for a binary classification output with $k = 1$.

To build a sequence, we take the top-k best candidates from $t - 1$, combine them with all possible labelings at t by multiplying their probabilities and filter the best k sequences again. It is worth noting that, under the assumption of independence between time steps, the result of beam decoding is actually the set of sequences with highest probability (not true if the steps are not independent like in speech recognition, the case of [11]).

5 Experimental setup

5.1 Dataset

For the sake of reproducibility, we chose to use open-source data. More specifically, we used the AliMeeting dataset [17], a corpus composed of around 120 hours of Mandarin meeting recordings including both far-field and near-field data, released along with the M2MeT challenge in 2022. Each meeting session consists of a 15 to 30-minute discussion by groups of 2-4 speakers with balanced gender coverage, different speaker overlap ratio and recorded in rooms of different size. The data is divided into three sets: 104.75 hours for training, 4 hours for validation and 10 hours for the test set.
We divide the meeting recordings into smaller segments of audio, as working with their original lengths poses a couple of constraints.
On the one hand, informativeness is not the same throughout a conversation. In a half an hour example there might be very few portions of it that are very informative and valuable but the rest of it may be useless so, using the full-length examples would either dilute the information present in each example or make us select a whole conversation when only a small part of it is useful for us.
On the other hand, annotating 30 seconds of audio is a lot easier in practice than annotating a 30 minute audio, where we have to match speakers across the whole conversation and we may have speakers not taking part in it for 10 or 15 minutes.
As mentioned before, a very common starting point for active learning setups is to divide the data into a small initial training set on which the model is trained

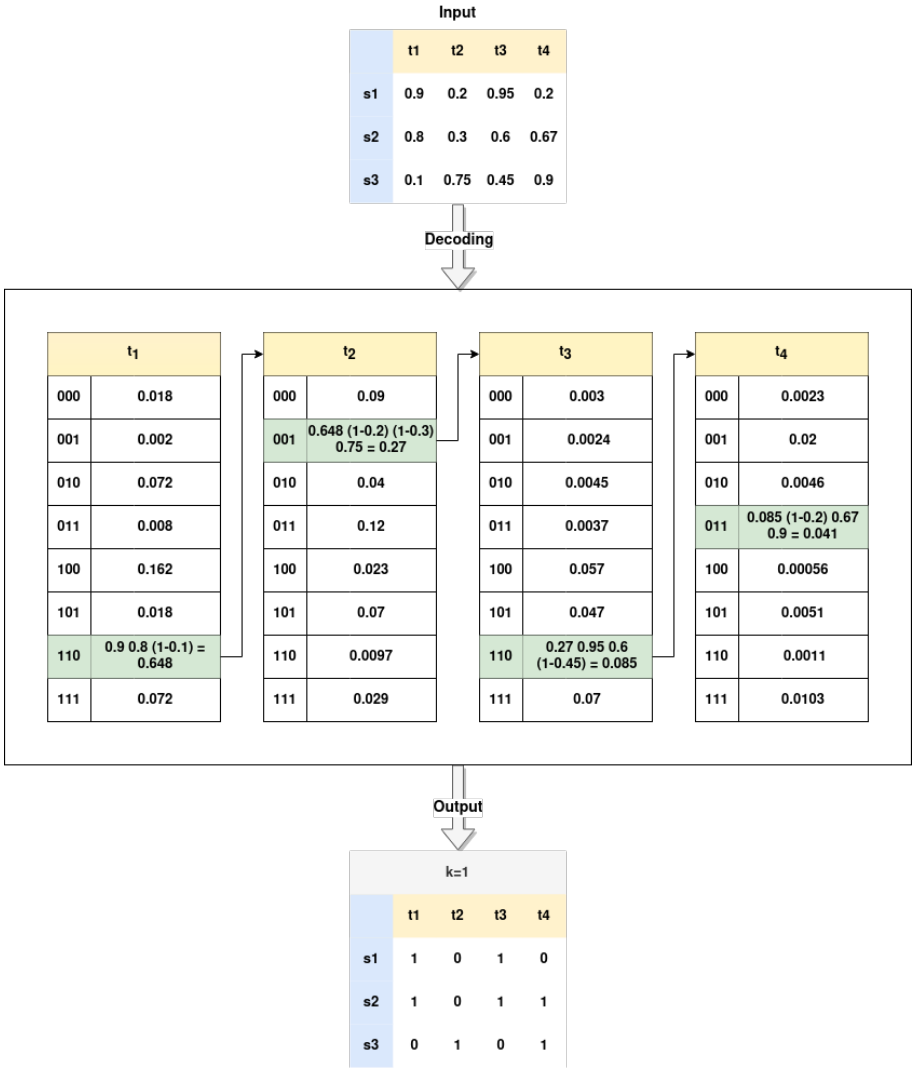


Figure 6: Example of how to generate candidate sequences with beam search for a speaker diarization hypothesis that shows active speaker ($y = 1$) probabilities. Green color corresponds to top-1 labels at time t.

so as to have a reference from where to begin sampling from the rest of the data, which remains unlabeled. This is particularly relevant for those sampling strategies that make use of the model's predictions. In our case, we just consider all of the AliMeeting training data as part of the initial unlabeled pool since we already have a starting point in the use of a pretrained model.

5.2 Model

Pyannote [18, 19] is an open-source kit of Python libraries for speaker diarization with neural networks. Pyannote's 2.1 default speaker diarization pipeline is divided intro three main stages or submodules:

1. Speaker segmentation applied to a short sliding window with a model based on the SincNet architecture [20].

2. Generation of local speaker embeddings with a neural network based on the X-Vector architecture [21], modified to receive SincNet features as input.

3. Hierarchical agglomerative clustering (HAC) of local speaker embeddings with centroid linkage to group them and retrieve a global speaker diarization output.

Pyannote also provides tools to easily fine-tune this pipeline, and having the speaker segmentation and speaker embedding extraction tasks divided into two different neural networks makes it a lot more convenient to fine-tune either of them.
The base model used for our experiments is a pretrained segmentation model[3] [18, 19], which has been previously trained on the AMI [22, 23, 24], AISHELL [25], DIHARD [26], REPERE [27] and VoxConverse [28] datasets.
The retraining phase of our active learning loop (see Figure 4a) implies the fine-tuning of this pretrained segmentation model, leaving the embedding extraction network unchanged.

5.3 Evaluation metrics

A very common metric to evaluate speaker diarization is Diarization Error Rate (DER), which measures three kinds of errors [29]:

- **False alarm**. Duration of non-speech incorrectly classified as speech (see Figure 7a).

- **Missed speech**. Duration of speech incorrectly classified as non-speech (see Figure 7b).

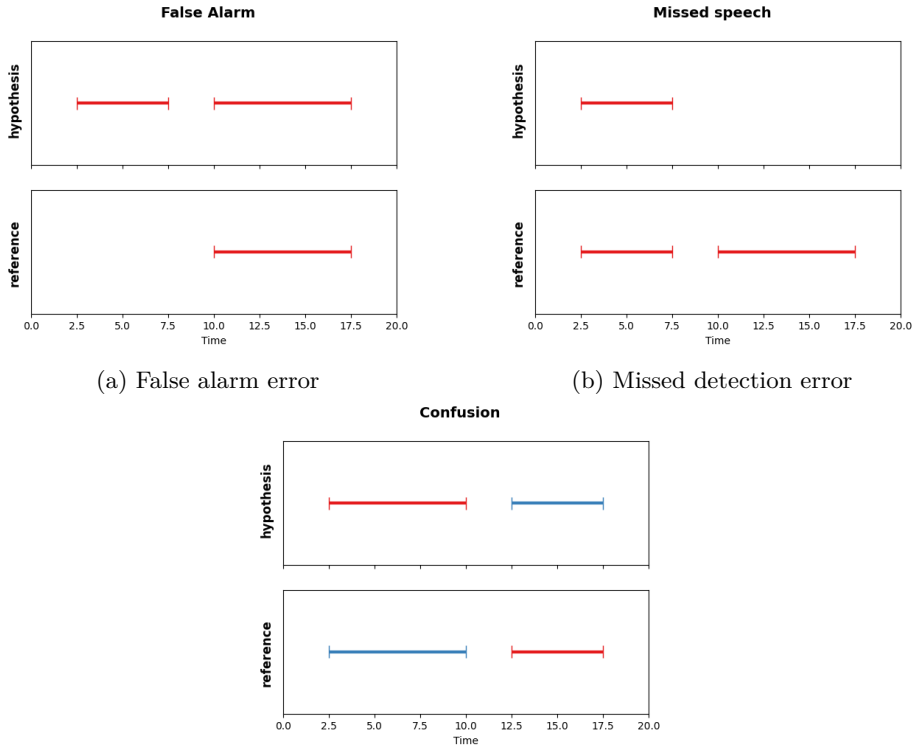
- **Confusion**. Duration of speech that is assigned to the wrong speaker (see Figure 7c).

To obtain the DER for a hypothesized diarization output, these three components are added up and divided by the total duration of reference speech, as shown in Equation 7.

$$DER = \frac{False\ alarm + Missed\ speech + Confusion}{Total\ speech} \tag{7}$$

[3] huggingface.co/pyannote/segmentation

The overlapping is counted as error once per overlapping speaker. Also, note that the DER can go above 1.
For mapping reference speakers to hypothesized speakers, the Hungarian algorithm [30] is used in order to minimize the confusion component.



(a) False alarm error

(b) Missed detection error

(c) Speaker confusion error

Figure 7: Illustration of the three DER components as hypothesis errors.

In the scenarios illustrated in Figure 7, we can observe that the false alarm error in Figure 7a would be equal to 5 seconds and the missed speech in 7b would be 7.5 seconds. In the confusion example from Figure 7c, the blue reference speech would be matched to the red hypothesis speech by the Hungarian algorithm, and that would entail a confusion error of 12.5 seconds after summing up the duration of the red hypothesis speech (7.5 seconds) and the blue hypothesis speech (5 seconds).
To evaluate the performance of our segmentation model, we can compute a local version of the DER, to which we will refer as local DER or chunk DER. We calculate this chunk DER by permutating the local speaker hypothesis to maximize their similarity to the reference.
Another possible metric to evaluate would be the global DER of the whole di-

arization pipeline, to assess the impact of fine-tuning the segmentation model on the full pipeline's performance. However, we will use chunk DER here since we are focusing on improving the quality of the segmentation model.

6 Results

For evaluating an active learning sampling strategy we perform the experiments given two parameters: *n_queries* and *batch_size*.
The first one, *n_queries*, is equal to the number of iterations to perform in our active learning loop. At every iteration of the process, we sample *batch_size* hours of data from the unlabeled pool - which actually corresponds to AliMeeting's training set - and move[4] it from there to the active learner's labeled training set, which contains all of the data that has been sampled up to that moment. Then, we fine-tune the pretrained segmentation model on all of the accumulated training data and evaluate its performance on AliMeeting's test set, that remains fixed throughout the experiment.
In addition, we use the model we just fine-tuned to re-rank (if needed)[5] the remaining unlabeled examples for the next iteration. As discussed earlier, in pool-based sampling we evaluate the whole collection of unlabeled data before greedily sampling. Thus, when using an active learning strategy based on the predictions of our model, we can use its most recent state so as to obtain more accurate estimations of the informativeness.

We ran a preliminary experiment to choose a combination of speaker-time aggregation methods for the uncertainty strategy. We compared three different options for each dimension, namely minimum, maximum and average. The results can be observed in Figure 8.
Our first choice was combining minimum aggregation for speakers and mean aggregation over time. Intuitively, this seemed like a good option since we are interested in high uncertainty (minimum confidence) over at least one speaker and then aggregate that over time because we want the example to be useful overall.
To ascertain this, we first tried varying the speaker aggregation method with a fixed aggregation method over time (average). As no significant difference was observed, we then fixed the speaker aggregation to minimum (our first choice) and varied the time aggregation. We could not run more than 1 experiment per combination due to time and computational limitations.
Even though these results are subject to some randomness, all aggregation strategies demonstrated fairly similar results. Following our initial intuition, we decided to use minimum aggregation for speakers and mean aggregation for

[4]We move the samples from the unlabeled pool to the training set because we are using an academic dataset where labels are available. In a real scenario, this step would imply manually annotating the data.

[5]We may not need to re-rank examples if we are using random sampling or if we are in the last iteration of the loop.

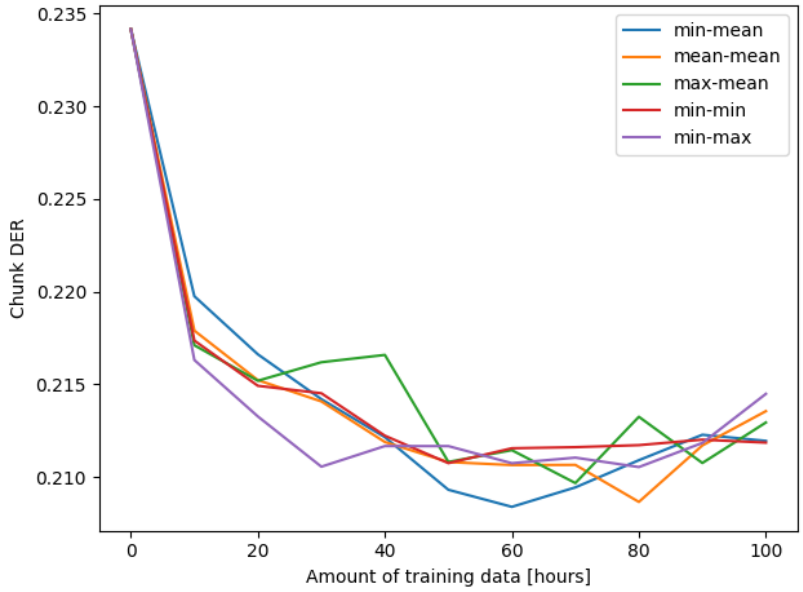


Figure 8: Evolution of Chunk DER as a function of the amount of training data used at each iteration. The curves correspond to the uncertainty sampling strategy when using different speaker-time aggregation methods.

time in all of our following experiments.

6.1 Random vs. active learning-based sampling

Our first experiment involved comparing the impact of the random, uncertainty-based (with augmentations to help estimate confidence) and EGL (marginalized over the top-10 most likely labelings) strategies on our active learning loop with $n\_queries = 10$ and $batch\_size = 10$ hours. To alleviate the randomness inherent to the training process, we performed 5 independent runs for each strategy and then average the results.

In Figure 9, we show the mean evolution of the chunk DER for these three strategies, including a range of the standard deviation over the different runs. The exact values are reported in tables 2, 3, 4 in the Appendix.

The starting point for the chunk DER at iteration 0 (before any sampling) is the same for all strategies and is obtained as a result of evaluating the pretrained model on AliMeeting's test set.

We can observe that both active learning strategies outperform random sampling. EGL is upper bounded by random sampling and lower bounded by uncertainty sampling in most cases, and uncertainty sampling with augmentations is clearly the best strategy.

Furthermore, the best result achieved by random sampling, which is 0.2130 with

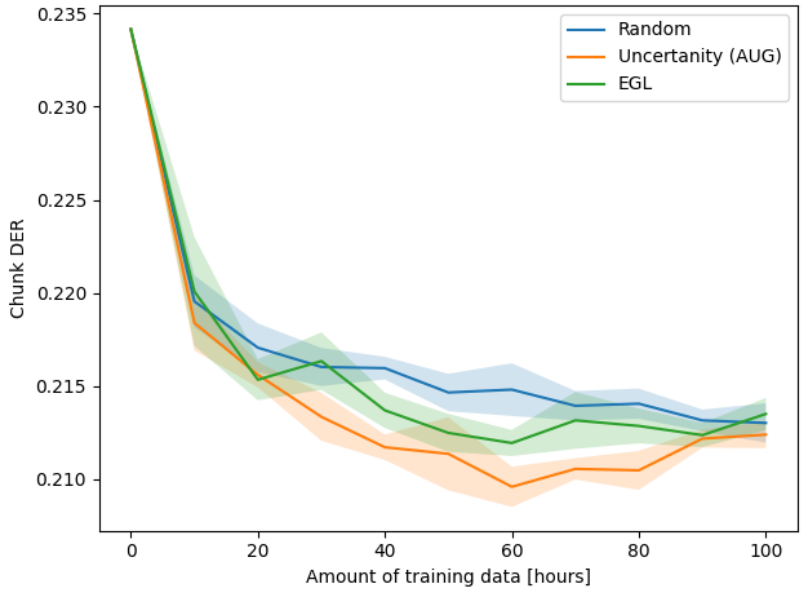


Figure 9: Evolution of Chunk DER as a function of the amount of training data used at each iteration. The curves for every strategy are obtained from averaging the results of 5 different runs/experiments.

100 hours of data is beaten by EGL with 50% less data and by uncertainty sampling with 60% less.
Another interesting point is the fact that with random sampling the chunk DER holds a decreasing tendency until the last iteration, whereas with uncertainty and EGL it reaches its minima both at 60 hours and then it becomes stationary or even gets worse.
One initial consideration may be that our model is overfitting the training data. In our experiments, the pretrained model is fine-tuned from scratch at every iteration with the current training set, instead of taking the fine-tuned model from the previous iteration and updating it on the new data. This way, we try to alleviate overfitting and catastrophic forgetting, which is a problem consisting on the model's performance degradation on previously seen data when trained on new data [31]. Neural networks are specially prone to suffer from this, since they share knowledge across many layers and parameters, that go through a lot of updates during gradient descent.
Another possible explanation may be that the active learning strategy is saturating. Prediction-based strategies like uncertainty sampling and EGL are designed to look for informative samples but one of their main limitations is that they are not good at finding diverse samples. So, these strategies may initially succeed at sampling informative instances but, at the same time, they might be misrepresenting the overall data distribution, biased towards certain

areas of the input space.

6.2 Impact of data augmentation in model confidence estimation

In a second experiment, we evaluated the quality of the confidence estimations of the segmentation model's predictions. Particularly, we wanted to compare the estimations when using data augmentations with the *dummy* estimations, obtained directly from the raw softmax scores.

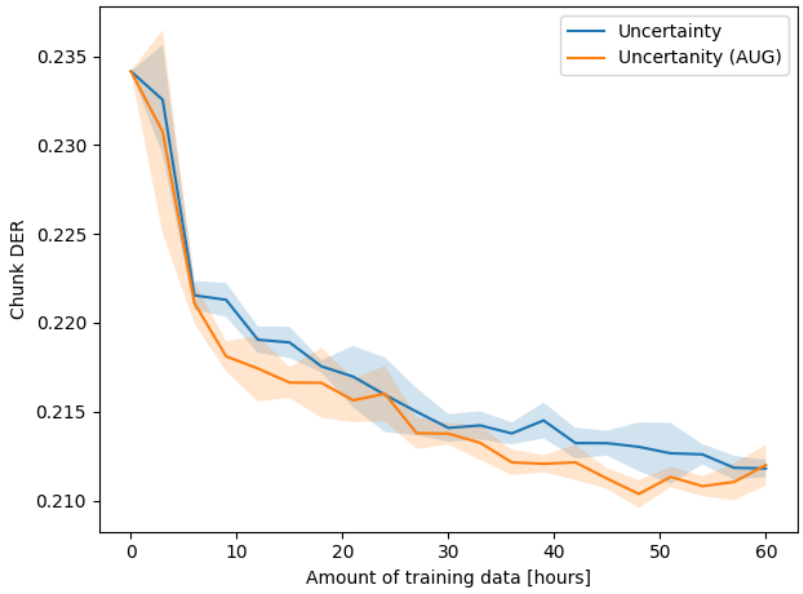


Figure 10: Evolution of Chunk DER as a function of the amount of training data used at each iteration. The curves for every strategy are obtained from averaging the results of 5 different runs/experiments.

We report in Figure 10 the chunk DER evolution for this experiment, with the hyperparameter values $n\_queries = 20$ and $batch\_size = 3$ hours. The results are again an average of 5 independent runs for each strategy and the exact numerical results are presented in tables 5 and 6.
As expected, uncertainty with augmentations provides better confidence estimations and outperforms plain uncertainty. Nevertheless, the gap between them is narrower than the one between uncertainty with augmentations and EGL or random.
Considering that computing augmentations over a large dataset and performing inference on all of the augmented data can be very time and resource consuming, this comparison provides us the insights to decide whether we want to use

them in case we need high quality confidence estimations without time and/or computational constraints; or if we prefer to avoid them in the opposite case.

7 Conclusion and perspectives

The initial goals for this project were:

1. Adapt a few existing active learning strategies to speaker diarization and
2. Evaluate their performance

As presented in Section 6, the adapted strategies showed encouraging results when compared to random sampling.
Even though the performance improvement after fine-tuning the segmentation model may not seem huge (2.46 points of improvement for the best case scenario), it should be noted that the goal of this project was never to achieve state-of-the-art performance for the AliMeeting dataset.
The actual objective was to assess if active learning could beat a random sampling baseline in an speaker diarization context. The adapted strategies outperformed the baseline while needing 50% and even 60% less training data to do so.

This promising initial results open up a lot of possibilities for future work. Firstly, a new active learning strategy based on diversity should be adapted to diarization, given the saturation and lack of diversity problems observed in prediction-based strategies. Cluster-based sampling rises as an interesting option, based on its model independent nature and thus absence of any model-related bias.
Actually, diversity sampling techniques are rarely found on their own. In an advanced active learning setup, uncertainty and diversity sampling strategies are usually combined. Intuitively, we can use cluster-based sampling to divide our data into groups that provide a diverse representation of it and then use uncertainty-sampling to find the most informative examples within each cluster.
Another diversity sampling strategy that could be very interesting in the context of Ava is real-world diversity sampling, that would allow us to sample data representative of Ava real users based on language, country, gender or age metadata.
Finally, for further evaluating and confirming the potential of the adapted strategies, these should be tested on Ava's internal data to see whether or not the good results translate from an academic to a real-world dataset and active learning still outperforms the random baseline in that situation.

References

[1] Burr Settles. Active Learning Literature Survey. Technical Report, University of Wisconsin-Madison Department of Computer Sciences, 2009. Accepted: 2012-03-15T17:23:56Z.

[2] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning, June 2022. arXiv:2206.14486 [cs, stat] version: 1.

[3] Tae Jin Park, Naoyuki Kanda, Dimitrios Dimitriadis, Kyu J. Han, Shinji Watanabe, and Shrikanth Narayanan. A Review of Speaker Diarization: Recent Advances with Deep Learning, November 2021. arXiv:2101.09624 [cs, eess].

[4] Xavier Anguera, Simon Bozonnet, Nicholas Evans, Corinne Fredouille, Gerald Friedland, and Oriol Vinyals. Speaker Diarization: A Review of Recent Research. *IEEE Transactions on Audio, Speech & Language Processing*, 20:356–370, February 2012.

[5] Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-End Neural Speaker Diarization with Self-attention, September 2019. arXiv:1909.06247 [cs, eess].

[6] Keisuke Kinoshita, Marc Delcroix, and Naohiro Tawara. Integrating end-to-end neural and clustering-based diarization: Getting the best of both worlds, February 2021. arXiv:2010.13366 [cs, eess, stat].

[7] Keisuke Kinoshita, Marc Delcroix, and Naohiro Tawara. Advances in integration of end-to-end neural and clustering-based diarization for real conversational speech, August 2021. arXiv:2105.09040 [cs, eess].

[8] Juan M. Coria, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. Overlap-aware low-latency online speaker diarization based on end-to-end local segmentation, September 2021. arXiv:2109.06483 [cs, eess].

[9] R. Munro, R. Monarch, and C.D. Manning. *Human-in-the-Loop Machine Learning: Active Learning and Annotation for Human-centered AI*. Manning, 2021.

[10] C E Shannon. A Mathematical Theory of Communication.

[11] Jiaji Huang, Rewon Child, Vinay Rao, Hairong Liu, Sanjeev Satheesh, and Adam Coates. Active Learning for Speech Recognition: the Power of Gradients, December 2016. arXiv:1612.03226 [cs, stat].

[12] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data, March 2017. arXiv:1703.02910 [cs, stat].

[13] Terrance DeVries and Graham W. Taylor. Learning Confidence for Out-of-Distribution Detection in Neural Networks, February 2018. arXiv:1802.04865 [cs, stat].

[14] Yarin Gal and Zoubin Ghahramani. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference, January 2016. arXiv:1506.02158 [cs, stat].

[15] Burr Settles, Mark Craven, and Soumya Ray. Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[16] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, page 1070, Honolulu, Hawaii, 2008. Association for Computational Linguistics.

[17] Fan Yu, Shiliang Zhang, Yihui Fu, Lei Xie, Siqi Zheng, Zhihao Du, Weilong Huang, Pengcheng Guo, Zhijie Yan, Bin Ma, Xin Xu, and Hui Bu. M2MeT: The ICASSP 2022 Multi-Channel Multi-Party Meeting Transcription Challenge, February 2022. arXiv:2110.07393 [cs, eess].

[18] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. pyannote.audio: neural building blocks for speaker diarization. In *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[19] Hervé Bredin and Antoine Laurent. End-to-end speaker segmentation for overlap-aware resegmentation. In *Proc. Interspeech 2021*, 2021.

[20] Mirco Ravanelli and Yoshua Bengio. Speaker Recognition from Raw Waveform with SincNet, August 2019. arXiv:1808.00158 [cs, eess].

[21] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, Calgary, AB, April 2018. IEEE.

[22] Iain Mccowan, J Carletta, Wessel Kraaij, Simone Ashby, S Bourban, M Flynn, M Guillemot, Thomas Hain, J Kadlec, V Karaiskos, M Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska Masson, Wilfried Post, Dennis Reidsma, and P Wellner. The AMI meeting corpus. *Int'l. Conf. on Methods and Techniques in Behavioral Research*, January 2005.

[23] Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska,

Iain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. The AMI Meeting Corpus: A Pre-announcement. In Steve Renals and Samy Bengio, editors, *Machine Learning for Multimodal Interaction*, Lecture Notes in Computer Science, pages 28–39, Berlin, Heidelberg, 2006. Springer.

[24] Jean Carletta. Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus. *Language Resources and Evaluation*, 41(2):181–190, May 2007.

[25] Yihui Fu, Luyao Cheng, Shubo Lv, Yukai Jv, Yuxiang Kong, Zhuo Chen, Yanxin Hu, Lei Xie, Jian Wu, Hui Bu, Xin Xu, Jun Du, and Jingdong Chen. AISHELL-4: An Open Source Dataset for Speech Enhancement, Separation, Recognition and Speaker Diarization in Conference Scenario, August 2021. arXiv:2104.03603 [cs, eess].

[26] Neville Ryant, Prachi Singh, Venkat Krishnamohan, Rajat Varma, Kenneth Church, Christopher Cieri, Jun Du, Sriram Ganapathy, and Mark Liberman. The Third DIHARD Diarization Challenge, April 2021. arXiv:2012.01477 [cs, eess].

[27] Aude Giraudel, Matthieu Carre, Valerie Mapelli, Juliette Kahn, Olivier Galibert, and Ludovic Quintard. The REPERE Corpus : a multimodal corpus for person recognition.

[28] Joon Son Chung, Jaesung Huh, Arsha Nagrani, Triantafyllos Afouras, and Andrew Zisserman. Spot the conversation: speaker diarisation in the wild. In *Interspeech 2020*, pages 299–303, October 2020. arXiv:2007.01216 [cs, eess].

[29] Hervé Bredin. pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, Stockholm, Sweden, August 2017.

[30] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[31] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, April 1999.

A Extended results

Strategy	Iteration	# hours	Chunk DER
Random	0	0	0.2342
Random	1	10	0.2196
Random	2	20	0.2171
Random	3	30	0.2160
Random	4	40	0.2160
Random	5	50	0.2147
Random	6	60	0.2148
Random	7	70	0.2139
Random	8	80	0.2141
Random	9	90	0.2132
Random	10	100	**0.2130**

Table 2: Numerical results from experiment 6.1 of random strategy. **Bold** shows best iteration.

Strategy	Iteration	# hours	Chunk DER
EGL	0	0	0.2342
EGL	1	10	0.2201
EGL	2	20	0.2153
EGL	3	30	0.2163
EGL	4	40	0.2137
EGL	5	50	0.2125*
EGL	6	60	**0.2119***
EGL	7	70	0.2132
EGL	8	80	0.2129*
EGL	9	90	0.2124*
EGL	10	100	0.2135

Table 3: Numerical results from experiment 6.1 of EGL strategy. **Bold** shows best iteration. * shows outperformance of random strategy.

Strategy	Iteration	# hours	Chunk DER
Uncertainty (AUG)	0	0	0.2342
Uncertainty (AUG)	1	10	0.2184
Uncertainty (AUG)	2	20	0.2156
Uncertainty (AUG)	3	30	0.2134
Uncertainty (AUG)	4	40	0.2117*
Uncertainty (AUG)	5	50	0.2114*
Uncertainty (AUG)	6	60	**0.2096***
Uncertainty (AUG)	7	70	0.2106*
Uncertainty (AUG)	8	80	0.2105*
Uncertainty (AUG)	9	90	0.2122*
Uncertainty (AUG)	10	100	0.2124*

Table 4: Numerical results from experiment 6.1 of uncertainty with augmentations strategy. **Bold** shows best iteration. * shows outperformance of random strategy.

Strategy	Iteration	# hours	Chunk DER
Uncertainty (AUG)	0	0	0.2342
Uncertainty (AUG)	1	3	0.2308
Uncertainty (AUG)	2	6	0.2211
Uncertainty (AUG)	3	9	0.2181
Uncertainty (AUG)	4	12	0.2174
Uncertainty (AUG)	5	15	0.2166
Uncertainty (AUG)	6	18	0.2166
Uncertainty (AUG)	7	21	0.2156
Uncertainty (AUG)	8	24	0.2160
Uncertainty (AUG)	9	27	0.2138
Uncertainty (AUG)	10	30	0.2138
Uncertainty (AUG)	11	33	0.2132
Uncertainty (AUG)	12	36	0.2121
Uncertainty (AUG)	13	39	0.2121
Uncertainty (AUG)	14	42	0.2121
Uncertainty (AUG)	15	45	0.2112
Uncertainty (AUG)	16	48	**0.2104**
Uncertainty (AUG)	17	51	0.2113
Uncertainty (AUG)	18	54	0.2108
Uncertainty (AUG)	19	57	0.2110
Uncertainty (AUG)	20	60	0.2120

Table 5: Numerical results from experiment 6.2 of uncertainty with augmentations strategy. **Bold** shows best iteration.

Strategy	Iteration	# hours	Chunk DER
Uncertainty	0	0	0.2342
Uncertainty	1	3	0.2326
Uncertainty	2	6	0.2215
Uncertainty	3	9	0.2213
Uncertainty	4	12	0.2191
Uncertainty	5	15	0.2189
Uncertainty	6	18	0.2176
Uncertainty	7	21	0.2170
Uncertainty	8	24	0.2159
Uncertainty	9	27	0.2150
Uncertainty	10	30	0.2141
Uncertainty	11	33	0.2142
Uncertainty	12	36	0.2138
Uncertainty	13	39	0.2145
Uncertainty	14	42	0.2132
Uncertainty	15	45	0.2132
Uncertainty	16	48	0.2130
Uncertainty	17	51	0.2127
Uncertainty	18	54	0.2126
Uncertainty	19	57	**0.2118**
Uncertainty	20	60	**0.2118**

Table 6: Numerical results from experiment 6.2 of base uncertainy strategy. **Bold** shows best iteration.