Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Multi-Task Learning for Segmentation of Clinical EEG Datasets

Master's Thesis

Jonas Lauer
jlauer@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Ard Kastrati
Prof. Dr. Roger Wattenhofer

April 29, 2025

Acknowledgements

I would like to use this opportunity to thank Ard for his great support.

How often did I come to your office thinking some sword of Damocles hangs over my head, only for you to tell me that it is just customary for Ms. Science to refuse to cooperate?

I don't know, I didn't count. But I'm sure she holds some grudge against me.

Abstract

Epilepsy, seizures, sleep disorders and other brain-related diseases can be diagnosed with the help of electroencephalography (EEG). Research into the automation of the laborious visual inspection needed to make such a diagnosis using machine learning techniques is still ongoing and has focused mainly on classifying a given EEG sample as one of several predefined classes. However, a given EEG signal can often be more concisely described by a set of events taking place in bounded intervals. Hence, segmenting the EEG sample, into multiple intervals of certain classes, requires the model to extract more information from it.

We ask whether this segmentation task structure gives the model the capability to improve its performance on a particular task with the help of knowledge it has gathered from other tasks. To that end, we train a model on eight different (semi-)publicly available datasets, both separately on each dataset and jointly with all datasets together. We use the concept of queries to tell the model which types of events it should try to detect in a given sample. Internally, our model uses the LaBraM EEG foundation model and parts of the Segment Anything Model as building blocks.

While mostly matching the performances of the separately trained models, our jointly trained model does not exceed these performances measurably.

Contents

Chapter 1

Introduction

This thesis is about predicting electroencephalography (EEG) events with deep learning methods. EEG has been applied for medical diagnosis since the early 20th century and, more recently, for brain computer interfaces (BCI, cf. e.g. [2]). An EEG signal is produced by placing multiple electrodes on the head of a human and measuring voltage fluctuations between these electrodes, which contain information about the activity of the human's brain. Each individual resulting signal is called a *channel* and any number of channels compose an *EEG recording*. The higher the number of electrodes, the higher becomes the spatial resolution of a recording, which can make it easier to trace patterns found in the signal through specific regions of the brain. A standard method of placing and naming these electrodes on the head is given by the so-called 10-20 system, a depiction of which is given in figure 1.1.

The appeal of EEG for research, medicine and BCI comes from the fact that it is non-invasive and that taking a measurement is relatively cheap, compared with, for example, MRI or PET scans [3]. Therefore, in the medical area, on which we will focus here, EEG is frequently used to detect epilepsy, seizures, sleep disorders and other brain-related disorders. To that end, an EEG recording is reviewed by an expert (e.g. a physician), who searches for specific patterns that are known to indicate certain diseases or events.

However, this process of manually searching for patterns can be cumbersome, especially since EEG recordings can be hours long (for example in the case of sleep disorders). Hence, full or partial automation of such tasks is highly beneficial, both for the physicians, who can spend more time on the consequences of the diagnosis results, and for the patients, as the quality of a diagnosis becomes independent of the tightness of a physician's time schedule and the diagnosis itself becomes cheaper.

Researchers have proposed a number of approaches, ranging from expert systems [4, 5] to deep learning models [6], to address this task. It should be noted that, because the brains of different humans can work quite diversely, the generalizability of a method to new patients can be a major issue. Most approaches that we know about represent the problem at hand as a classification task: Given a slice of EEG data, map it to one of several known classes. A to date less used approach [7] is to formulate the problem instead as a *segmentation task*: Given a

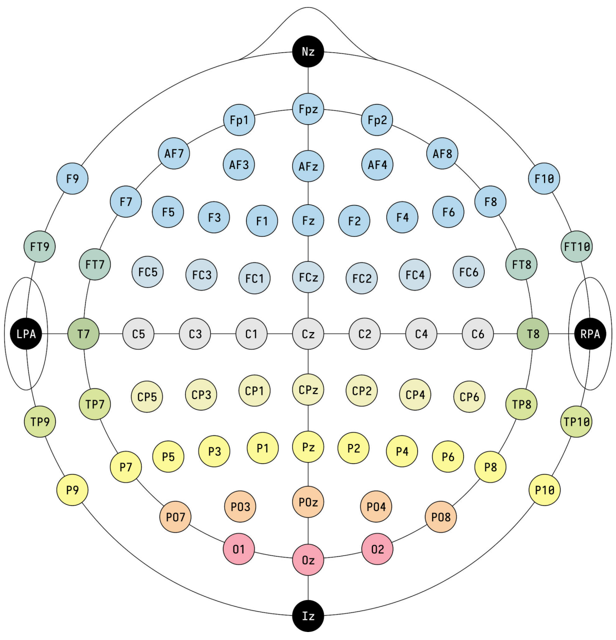


Figure 1.1: An extended version of the 10-20 system. Graphic (cropped) by Laurens R. Krol [1].

slice of EEG data, segment it time- (and potentially channel-)wise into intervals of known classes. These intervals should correspond to events taking place during that time in the EEG. In this work, we refer to a *segment* as the maximal interval (in the time-dimension), in which all timesteps have the same class.

Note that generally, the structure of the results of both approaches is the same: A class is assigned to every EEG timestep. However, there are two important differences. First, the classification approach produces segments that are equal to the length of the given EEG slice. Thus usually, the length of classification-induced segments is constant, while the length of segmentation-induced segments can vary. Second, the classification-approach, by definition, only takes into account the given EEG slice for a single class prediction. In contrast, the segmentation-approach also takes into account EEG signals that lie outside an event, thus providing the model with additional information of what happens around it, which might help improve the model's performance.

A direct consequence of this for the classification approach is that all samples must be completely covered by a specific class and have to be selected accordingly from the source dataset. While this sampling technique makes it easier for the model to distinguish different classes, it incorporates additional information in the training and evaluation data that may not be available in a real-world application scenario, where the boundaries of events are typically unknown. In contrast, the segmentation approach can process samples drawn from arbitrary locations in the source dataset.

Overall, it can therefore be said that the segmentation-approach provides more flexibility than the classification-approach regarding the granularity of predictions. This makes it of course also potentially a more difficult task. We think, however, that this increased granularity might be beneficial in medical application areas. For instance, suppose that, in the end, a physician is only interested in answering the question whether a patient has suffered from a seizure or not. This question can be directly answered by both, classification and segmentation models. But even then, a segmentation-model additionally tells the physician where exactly it thinks a seizure happened. Hence, the physician can manually check if the model was correct. Performing this manual check should take significantly less time than manually scanning the entire EEG and maintains a certain degree of human control in the otherwise automated EEG analysis process.

Switching from a medical to a machine learning perspective, with the segmentation approach only recently being applied to EEG [7], many fundamental questions remain open so far. For example, it would be important to understand whether the segmentation approach is able to generalize well enough to simultaneously detect many different event types, as this generalization could provide means to boost the model's performance on individual tasks, by letting different tasks learn from each other. A positive answer to this question might subsequently open the path towards models that understand EEG on a much more general level and across multiple application areas, similar to what has been achieved in natural language processing with large language models [8].

Our attempt to make some progress on this question involves training a deep learning segmentation model on multiple datasets, both separately and for all datasets together. A similar approach in this direction has already recently been taken for the classification approach, but with limited success [9], as the multi-task model's performance on individual tasks did not exceed that of single-task models.

We will see that our approach is *not* a lot more effective and exceeds single-task performance only marginally and only for some datasets, while performing worse in others.

The remainder of this text is structured as follows: In the next chapter 2, we will take a look at the datasets we will use, along with their internal structure and the types of events they distinguish. Then (chapter 3), we move on to formally define the segmentation task that was prosaically but not rigorously introduced above. In the same chapter, we will also discuss the consequences of this definition for our multi-dataset setting, from which the concept of "queries" emerges. When the problem to solve and the inputs to our model are understood, we continue in chapter 4 with the construction of the model itself. Lastly, chapter 5 covers our experiments and the ensuing conclusions in greater detail, while chapter 6 summarizes and explores the next steps that could be taken by future research.

Chapter 2
Datasets

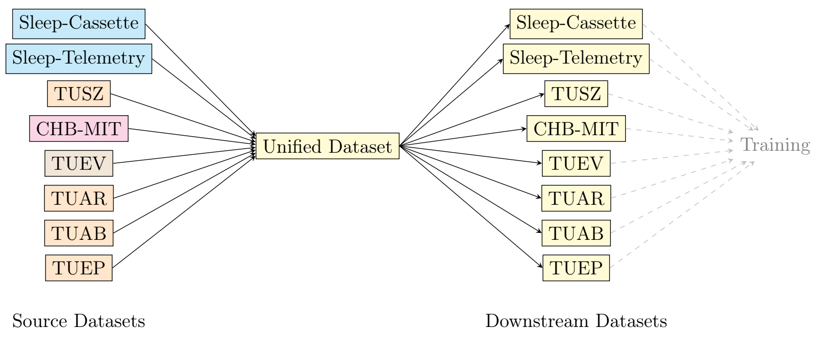


Figure 2.1

Now that we have a rough understanding of what we wish to achieve, let's take a look at the datasets we will work with, so that we understand a bit better the variety of the data we are dealing with (section 2.1). After that, we will see how to generalize from these individual datasets, in order to merge them into a uniform structure (section 2.2) that provides a simpler interface to derive the downstream datasets that we will use for training (section 2.3).

2.1 Source Datasets

But before diving into the different source datasets, it is necessary to clarify a bit the vocabulary that is used in these contexts.

EEG data is recorded from usually many different human beings. While in clinical settings, these are often referred to as *patients*, there are also lab experiments in which they are referred to as *participants*. Since future work might apply parts of this work also to non-clinical settings, we opted to call them *participants* in the following.

Next, there are different names or units specifying the time-wise relation of EEG data. The smallest such unit is the *trial*, that is mostly used in BCI datasets and indicates the start and end of an experiment that is repeated often and usually takes only a few seconds. A classic example would be a participant imagining to lift one of their hands for several seconds. However, trials are not really used in clinical datasets.

The maximal segment containing a continuously recorded EEG signal is called a *run*. Within a run, the EEG electrodes remain on their positions on the participant's head all the time (and are not taken off or turned off). Typically, in the source datasets a run is equivalent to an individual file. We sometimes also refer to these files as *records* or *recordings*.

The last and largest unit is called a *session*. A session may consist of multiple runs that are recorded with short gaps in-between that are not longer than a few hours.

Now that we have defined these terms, we can start to look at the datasets we use. In this work, we consider datasets about sleep stages, seizures, eye movements, muscle contractions and other artifacts, as well as epilepsy and general pathological conditions. For each dataset, we report in table 2.1 the different event types as well as their respective frequency of occurrence in the dataset.

2.1.1 Sleep Stages

Human sleep can be categorized into different phases, or "stages". Overall, one can distinguish rapid eye movement (REM) phases and non-REM phases, while the non-REM phases can be further differentiated by the "depth" of the sleep (i.e. how hard it is to wake someone up), into N1, N2, N3 and N4 phases (ordered by increasing depth).

In this work, we will use the Sleep-EDF database [10], which includes two datasets: Sleep-Cassette and Sleep-Telemetry. Sleep-Cassette collected data from 78 healthy participants for about 20 consecutive hours, during which the participants continued their normal activities (wearing a mobile EEG recorder). In Sleep-Telemetry, some of the 21 participants had received temazepam medication and the recordings were made in a hospital, with an average length of 9 hours. Both datasets include segments of the above sleep stages, as well as of wake phases and "movement time". Occasionally, the label creators did not know which sleep phase a segment should be assigned to. In that case, they labelled it as "sleep stage ?". The recordings contain three channels: Fpz-Cz, Pz-Oz, as well as a horizontal electrooculography (EOG) channel that can measure eye movements.

2.1.2 TUSZ and CHB-MIT

Seizures are pathological conditions of the brain, which can lead to a brief loss of consciousness or uncontrolled shaking of the concerned person's body. Different seizure types can be distinguished, based on the resulting behavior, as well as by the regions of the brain that are affected.

In this context, we will consider two datasets: TUSZ and CHB-MIT. TUSZ

Dataset Name	**Dataset Length**	**Event Type**	**Number of Segments**	**Average Segment Length**
Sleep-Cassette	3470.2h	Wake Phase	3698	2320s
		N1 Phase	5944	109s
		N2 Phase	6378	325s
		N3 Phase	3538	75s
		N4 Phase	1015	125s
		REM Phase	1560	497s
		Movement Time	123	31s
Sleep-Telemetry	361.6h	Wake Phase	744	177s
		N1 Phase	1311	82s
		N2 Phase	1718	325s
		N3 Phase	1218	80s
		N4 Phase	413	216s
		REM Phase	378	630s
		Movement Time	69	34s
TUSZ	1038h	Seizure	11	72s
		Focal Seizure	2313	63s
		Generalized Seizure	1272	77s
		Simple Partial Seizure	49	30s
		Complex Partial Seizure	363	93s
		Absence Seizure	114	8s
		Tonic Seizure	20	32s
		Tonic-Clonic Seizure	48	124s
		Myoclonic Seizure	2	646s
CHB-MIT	979.9h	Seizure	185	62s
TUEV	148.7h	Spike and/or Slow Wave	78	2s
		Generalized Periodic Epileptiform Discharge	491	3s
		Periodic Lateral Epileptiform Discharge	726	2s
		Artifact	362	5s
		Eye Movement	269	1s
TUAR	100h	Eye Movement	11189	5s
		Chewing	348	15s
		Shivers	36	7s
		Muscle Artifact	6815	12s
		Electrode Pop	8	1s
		Electrode Artifact	4492	11s
TUAB	1141h	Abnormal	1472	1402s
		Normal	1521	1343s
TUEP	631.8h	Epilepsy	1785	1070s
		No Epilepsy	513	712s

Table 2.1: An overview of the source datasets' size and event types.

[11], with 622 participants, is a subset of the TUEG corpus [12], a large and mostly unlabelled collection of EEG recordings from the Temple University Hospital. Similarly, CHB-MIT [13] was collected at the Children's Hospital Boston from 23 participants. Because this data stems from a real hospital, and was not recorded under lab conditions, it may contain a lot of noise and other irregularities. Perhaps, for instance, in different recordings the same electrodes could have been positioned at different positions due to an emergency not allowing for more careful placement. Additionally, the recordings were produced on many different devices under varying configurations concerning the sampling rate, the set of electrodes used and the type of reference channel used.

While CHB-MIT only classifies data as either "seizure" or not, TUSZ contains labels from 8 different seizure types.[1] Using [14], we filtered out EEG channels whose meaning is unknown.

As can be seen in table 2.1, both datasets are heavily imbalanced and contain much fewer seizures than unlabelled regions.

2.1.3 TUEV and TUAR

Like TUSZ, TUEV and TUAR are also subsets of the TUEG corpus, including data from 370 and 213 participants, respectively. They contain short events of few seconds length that can serve as indicators for epilepsy (in the case of TUEV) or contain artifacts that cause disturbances in the EEG and need to be filtered out. Both datasets are relatively small, compared to e.g. TUSZ, which might have an impact on the learning behavior of our model (see chapter 5).

An important difference between TUEV and the other TUEG-derived datasets is that here, the original TUEG-recordings were cut and only pieces containing activity (or clear non-activity), with respect to the events considered, were included. This has potentially important implications for our learning task, as we will further discuss in section 3.2.

2.1.4 TUAB and TUEP

Lastly, we also decided to include the TUEG-subsets TUAB and TUEP, with 2383 and 200 participants, respectively. Both can be viewed as datasets for binary classification – between "abnormal" and "normal" conditions, in the case of TUAB, and between participants with and without epilepsy in the case of TUEP. In contrast to all of the previous datasets, they do not include segments, where pathological conditions become clearly visible. Instead, each record is merely labelled as one of the two classes. To still be able to integrate these

[1]However, our evaluation set will only contain 6 different seizure types, because the simple partial and myoclonic seizures are too scarce in the dataset.

datasets into our pipeline, we create artificial “abnormal” or “epilepsy” segments spanning an entire record.

2.2 Common Dataset Structure

As an intermediate step, we will convert the different datasets into a common format which is described in this chapter. This unified dataset will then serve as the reference point to derive the different downstream datasets used during training. Hence, it is important to make the format as general as possible while including as much information as possible from the source datasets. To store this unified dataset and the downstream datasets, we will use the HDF5 file format [15], because, in our case, accessing all the data from a single file has benefits for the compute cluster architecture we rely on (for us, that is Slurm [16]).

As a first step, we would like to unify the various events and other dataset properties within a single structure, so that we are able to access data from various datasets in a similar, simple fashion, without having to consider the different peculiarities of each dataset.

It turns out that segments are a versatile enough structure to represent almost any information given by a source dataset. In many of the datasets, the events we care about are already given as segments and in the few where they are not (like in TUAB and TUEP), we can translate them into segments spanning an entire record.

Furthermore, we would also like to store information related to participants, sessions and runs, as this will later enable us to partition downstream datasets for cross-participant evaluation and to collect downstream dataset samples that do not cross record-boundaries. To that end, it makes sense to concatenate all of the records from a source dataset in the time-dimension, so that participant- and session-segments can be easily defined beyond record boundaries. This way, we can even encode attributes, that are valid in the entire dataset, as segments. This may concern, for instance, specific conditions in which the participants were during the recordings. For example, as mentioned in section 2.1.1, in the Sleep-Cassette dataset, participants were moving around with the electrodes on their head, which would not be the case for the TUEG-derived datasets that were, as far as we know, taken in a stationary setting.

We call the list containing all of these different segments `instances` in the following. Every element $e = (\text{type}, \text{start}, \text{stop})$ of this list is defined by its event type, as well as the start and end indices of its interval with respect to the time-dimension of our concatenated EEG signal. Additionally, for every event type, we define a sub-list containing the segments of this event type, since the search for events of a given type turned out to be a frequent use case and can be sped up this way.

EEG Matrix	**Included Datasets**	**Length (200 Hz)**	**Channels**
Sleep Stages	Sleep-Cassette, Sleep-Telemetry	2,758,888,000	Fpz-Cz, Pz-Oz, EOGh
TUEG-Tasks	TUSZ, TUAR, TUAB, TUEP	2,150,084,600	F7, Fz, O1, T4, Fp1, Sp2, T6, C3, Sp1, T2, F4, A2, Fp2, T1, P3, Pz, P4, F3, T5, O2, A1, Oz, C4, Cz, T3, F8
CHB-MIT	CHB-MIT	705,548,600	F3-C3, F8-T8, F4-C4, Pz-Oz, Fp1-F7, P4-O2, FT10-T8, P8-O2, T8-P8, Fz-Cz, P7-O1, T7-P7, Fp2-F8, T7-FT9, Cz-Pz, Fp1-F3, C3-P3, C4-P4, P3-O1, F7-T7, FT9-FT10, Fp2-F4
TUEV	TUEV	107,096,200	C3, P4, T6, F8, Oz, Sp1, T1, F4, A2, Sp2, F7, T5, F3, C4, T2, T3, Fp1, Pz, T4, Fz, O1, Fp2, A1, P3, Cz, O2

Table 2.2: The four EEG matrices in our unified dataset, along with the included datasets, length of the time-dimension (200 Hz) and the channels used.

As was noted in section 2.1, different datasets were recorded with different EEG (and occasionally also EOG) channels. From a storage perspective, it would thus produce a lot of overhead if we stored everything in a single matrix `eeg` $\in \mathbb{R}^{\text{time}\times\text{channels}}$. Instead, we will store everything in a single matrix *per dataset*, except for TUSZ, TUAR, TUAB and TUEP, as these datasets share the same set of channels and can thus be merged into a single matrix.[2] The same holds true for Sleep-Cassette and Sleep-Telemetry. Table 2.2 shows the four different EEG matrices we use, along with their channels and time-dimension size.

Because data from different sources was produced under varying conditions, we apply a few preprocessing steps to every recording from every dataset before concatenating them. First, we re-reference the EEG channels to an average reference. Then, we apply (if possible) a $0.1 - 75$ Hz bandpass filter, followed by a 50 Hz notch filter. Lastly, we resample the signal to 200 Hz. The reason for using 200 Hz as our sampling rate will become clear later, in section 4.2.

[2] We were not able to also merge TUEV into this matrix, because, as mentioned in section 2.1.3, there the original EEG records were cut into smaller pieces and could thus not be merged anymore with records from the other TUEG-datasets.

2.3 Downstream Datasets

Given the unified dataset, we can now discuss how the downstream datasets should look like. They are the datasets that are directly used for training the model, so they should provide a fast way to get the training data we want, without requiring much additional preprocessing.

Because we want to compare the model's performance when trained on each dataset individually with a model trained on all datasets simultaneously, we create one downstream dataset per source dataset. We then provide means to combine these datasets on the fly to train the model on multiple datasets, which provides greater flexibility, because this way we do not have to recompute the downstream datasets for the combined training.[3]

Each downstream dataset contains a list of EEG samples of fixed length l (to enable efficient batching) and associated with each EEG sample is a list of segments $s = (\text{type}, \text{start}, \text{stop})$, where $\text{start}, \text{stop} \in \{0, \dots, l\}$ are clamped to the EEG sample's length. This clamping means that we split events over multiple samples if they cross the sample borders.

In contrast to approaches in the literature [6], we do not use overlapping samples. Hence, our samples simply represent a partition of the original dataset. However, we did not apply padding and omitted remaining sampling-material shorter than l. Additionally, we did not let our samples cross record boundaries. Note that because we do not use overlapping samples, our downstream datasets are often distributed differently than some in the literature. In fact, this tends to make our datasets less balanced than their literature counterparts. We try to counteract this by using weighted sampling, with a weight defined for each sample. These weights are then used to draw a fixed number of samples (with replacement) for training, according to the weight-induced multinomial distribution (see section 4.6 for further details).

Notice that to find the samples we wish to include in a downstream dataset, we need to search the `instances` list in the unified dataset for the regions that contain the records of the source dataset. We can identify these regions with the help of auxiliary segments that were added during the creation of the unified dataset, e.g. "TUSZ training data" segments. However, there is no trivial ordering of segments in `instances`, because segments may overlap with and contain each other. If there are $n = |\texttt{instances}|$ segments and we make no further assumptions about the ordering of `instances`, it will thus take $\Omega(n)$ time to search through

[3]Notice that the fact that we use exactly one downstream dataset per source dataset makes the unified dataset a bit superfluous. But for us, it was helpful to have it, in order to better find the structure all downstream datasets should have in common and to be able to experiment faster with different downstream dataset structures (merging a source dataset into the unified dataset usually took much longer to run than deriving the downstream dataset from the unified one).

this list to identify a single sample. This procedure will need to be repeated n times, to derive all of the downstream datasets used for training and potentially even more often if we are looking for events with certain properties. For instance, we could be searching for all events inside a “run”-segment, because we want to ensure that we do not cross record boundaries within a sample. This approach to derive the downstream datasets would thus result in a computation complexity of at least $\Omega(n^2)$. In our case, $n \approx 100,000$, so we would not like to afford a quadratic runtime.

Instead, we will use a slightly different data structure that allows for a time-wise total ordering, by splitting start and end points of each segment and storing these points in time order. More precisely, we create a list of endpoints. Each entry $\text{endpoint} = (\text{IID}, b, \text{timestep})$ in this list consists of the index of the segment in `instances` this endpoint is from, a binary value $b \in \{0, 1\}$ indicating whether the endpoint is the start $(b = 0)$ or the end $(b = 1)$ endpoint of the segment, and the timestep that refers to the endpoint's time-dimension index in `eeg`. Creating and sorting this endpoint list takes $\Theta(n \log(n))$ time.

Now, to collect our downstream datasets, we can simply traverse this sorted endpoint list in time order and add some logic that keeps track of which segments are overlapping with the current time step. This allows us to select our downstream samples quasi on the fly, based on a large variety of conditions, yielding a $\Theta(n)$ runtime for sampling the downstream datasets after creating the sorted endpoint list.

Last but not least, in order to both train and evaluate our model, we need to split each downstream dataset into a training-part and an evaluation-part. While for TUSZ, TUEV and TUAB, the source datasets already provide such splits, this is not the case for the remaining five datasets. Hence, we need to split our downstream datasets manually. Because an application would probably have to apply our model to EEG recorded from people that are not in our datasets, we will first partition the participants of each datasets into training- and evaluation-participants. Then, we define the training set as the set of samples which were recorded from the training-participants, and analogously the evaluation set from the evaluation-participants' samples. This procedure is called “cross-participant” evaluation. We try to use 80% of a downstream dataset's samples for training and 20% for evaluation. Table 2.3 shows the main statistics of our downstream datasets.

Downstream Dataset	**Training Samples**	**Evaluation Samples**	**Event Type**	**Training Segments**	**Evaluation Segments**
Sleep-Cassette	617250	163158	Wake Phase	426160	112685
			N1 Phase	34584	11012
			N2 Phase	108600	26604
			N3 Phase	15946	3621
			N4 Phase	6292	2565
			REM Phase	39420	10352
			Movement Time	266	84
Sleep-Telemetry	62317	19021	Wake Phase	7201	1664
			N1 Phase	6209	1657
			N2 Phase	26699	9640
			N3 Phase	5471	1723
			N4 Phase	4857	1070
			REM Phase	12092	3138
			Movement Time	115	96
TUSZ	199766 (14518)	27784 (2484)	Focal Seizure	7552	1067
			Generalized Seizure	3653	683
			Simple Partial Seizure	171	0
			Complex Partial Seizure	2132	265
			Absence Seizure	80	64
			Tonic Seizure	44	4
			Tonic-Clonic Seizure	215	46
			Myoclonic Seizure	80	0
CHB-MIT	166884 (1275)	53556 (368)	Seizure	728 (656)	164 (236)
TUEV	22425 (664)	10797 (334)	Spike and/or Slow Wave	58	22
			Generalized Periodic Epileptiform Discharge	332	253
			Periodic Lateral Epileptiform Discharge	478	302
			Artifact	351	76
			Eye Movement	200	82
TUAR	17815 (11277)	4533 (2850)	Eye Movement	11811 (11957)	2992 (2846)
			Chewing	510 (603)	164 (71)
			Shivers	32	20
			Muscle Artifact	9046 (9108)	2620 (2558)
			Electrode Pop	6	2
			Electrode Artifact	5789 (5859)	1644 (1574)
TUAB	232334	23040	Abnormal	117752	10628
			Normal	114582	12412
TUEP	112698	28252	Epilepsy	97383	20989
			No Epilepsy	15315	7263

Table 2.3: The number of samples and segments in our downstream datasets. Observe that in our experiments, we will occasionally use "reduced" versions of the datasets (as explained in section 3.2). The samples and segments of the reduced dataset versions are written in brackets $(\cdot)$, if they differ from their unreduced counterparts. Note that, despite the fact that our reduced datasets contain the same segments as the unreduced ones, their number within the training and evaluation sets can vary, because the algorithm splitting a dataset into training- and evaluation-parts is randomized and is rerun for reduced datasets. The results in this table are computed with respect to a sample size of 16 seconds.

Chapter 3

Segmentation Tasks

In this chapter, we will formally define the "segmentation task" that was mentioned informally in the introduction and, given what we have learned about our datasets in chapter 2, we discuss some issues of this seemingly general definition (section 3.2). These issues will lead us to the concept of "queries" (section 3.3), which allow us to precisely state the tasks (table 3.1) we will run experiments on in chapter 5.

3.1 Problem Definition

Throughout this section and the rest of the report, we use the terms "event" and "segment" interchangeably, except when explicitly said otherwise. Both terms are used to describe an element of the `instances` list defined in the previous chapter.

Definition 3.1. Let us be given a set `classes` of the different event types we wish to distinguish. We assume (for this definition) that two events never overlap if the events have different types and both types are in `classes`. Let us also be given a (potentially long) EEG signal $\texttt{eeg} \in \mathbb{R}^{T \times \#\text{channels}}$ of (time-)length T and with #channels different channels, along with a list `instances` of events that take place within `eeg` and that all have a type that is in `classes`. Since, by our assumption, none of the events in `instances` overlap with each other, we can define the **segment representation** $\texttt{segments} \in \{0, \dots, |\texttt{classes}|\}^T$ of `instances` as

$$\texttt{segments}_i := \begin{cases} \text{type} & \text{if } \exists e = (\text{type}, \text{start}, \text{stop}) \in \texttt{instances} : i \in [\text{start}, \text{stop}) \\ 0 & \text{otherwise} \end{cases}, \qquad \forall i \in \{1, \dots, T\}$$

Thus, $\texttt{segments}_i = 0$ means that there is no event in `instances` overlapping with timestep i.

Then, a **segmentation task** is the problem of finding an integer $t \in \{1, \ldots, T\}$ and a function

$$\text{model} : \mathbb{R}^{t \times \#\text{channels}} \to \{0, \ldots, |\texttt{classes}|\}^t$$

such that

$$\text{model}(\texttt{eeg}_{a:a+t}) = \texttt{segments}_{a:a+t}, \qquad \forall a \in \{0, \ldots, T - t\}. \quad (3.1)$$

Here, the notation $x_{a:b}$ should mean the same as $(x_i)_{i \in [a,b]}$ and we also interpret $(\mathbb{R}^d)^c$ as $\mathbb{R}^{c \times d}$.

Put simply, a segmentation task is the problem of finding a model that, given an EEG sample, returns the corresponding segment representation.

Because we have the labelled EEG data from chapter 2 at our disposal, we will try to solve this task in a supervised manner. More formally, we partition our data into training and evaluation subsets and define (train) the model with knowledge of (`eeg`, `instances`) pairs from the training set, before evaluating its performance on the evaluation set.

The parameter t limits the length of the EEG signal, i.e. the *context*, the model is allowed to see for each of its predictions. The size of this context can have a large impact on the model's performance. If we make it too small, the short EEG sample may simply not provide enough information to make a correct segment prediction. On the other hand, we also don't want to make t too large, because that would limit the model's practical applicability[1] and cause problems during the training of the model (larger t means fewer training samples). We will later see that our model implementation is, in principle, able process EEG input of variable length, which will allow us to vary t between downstream datasets.

Note, however, that the assumption that events never overlap is too restrictive in general. For example, if we would want our model to learn to detect both "seizure" and "eye movement" events, but some participants move their eyes during a seizure, this causes an overlap between these two events. This means that the `instances`' segment representation is not unique anymore. Thus, errors may be introduced by the choice for one of the overlapping events over the other in the segment representation, if this choice is not known to the model. When such overlaps are small or scarce, the amount of error induced will be small compared to the overall size of the downstream dataset and we can most likely ignore it. If the overlaps are not scarce, we would define multiple segmentation task instances over the same EEG signal but for different event types, such that the different tasks' event types partition the original event types, i.e.

$$\biguplus_{j \in \text{tasks}} \texttt{classes}_j = \texttt{classes},$$

[1]For example, it would be inconvenient for a hospital to have to produce at least 1000 hours of EEG for each patient in order to be able to feed that EEG to our model for prediction.

but events don't overlap anymore *within* each of the `classes`$_j$.

We will find another argument for the necessity of defining multiple segmentation task instances in the following section and we will discuss the design incorporating these multiple instances into the model in section 3.3.

3.2 Unlabelled Data

Let us now briefly look at a more subtle issue that arises from the above definition of a segmentation task.

In all of the datasets we use, there are fragments in the EEG time series where the datasets provide no labels of any event type. To understand our modelling and loss definitions, it is important to ask what these 'empty' segments represent exactly. Does it mean that, within this segment, there are no events happening, at least among those types we consider? Or were the humans creating the labels simply not sure if there is any event happening in that segment, i.e. the labels may be incomplete? Let's call these different interpretations *type 1* empty segments and *type 2* empty segments, respectively.

Notice that the model cannot learn anything from type 2 empty segments, because we have no information about events within those segments. Thus, if possible, we would want to exclude them from training or at least not force the model to predict any specific class for them, i.e. remove those segments from equation 3.1 in definition 3.1. For the type 1 segments, on the other hand, we can safely let the model predict a dedicated "no-event" class. Now, we only need to find out which empty segments are type 1 and which are type 2.

After all, how we interpret these empty segments depends on the dataset. For example, as previously mentioned, in the sleep stage datasets, there is a dedicated "Sleep stage ?" label, which means the label editors were not sure which sleep/wake phase this segment belongs to (i.e. type 2). On the other hand, in the TUEG-derived datasets, there is a dedicated "background" label that stands for a segment where none of the other events considered occurred. For our goals, these TUEG "background" labels pose two problems: First, not every segment that is not labelled as any other event type is automatically assigned a "background" label. For example, in TUEV, the vast majority of EEG has neither an event nor a "background" label and "background" labels are not even always adjacent to other event labels but may occur somewhere in the middle of an EEG recording with no other labels nearby. This may not pose a problem in a classification task setting, where we can just classify EEG segments of a few seconds length that are fully labelled as one of the classes we consider (e.g. [6]). But in our segmentation task setting, we are interested in the boundaries of a segment. Hence, we need to be sure that the event really stops where the label stops and that if next to the label there is no other label, that this means that no other event takes place.

We cannot take for granted that this is indeed the case, if, for example, TUEV was labelled only with a classification task setting in mind.

To perhaps at least partially address this problem, we opt to define *reduced* variants of those datasets for which we are not sure what empty segments mean, so that in the reduced variant, we can be more certain that these are type 1 segments. We do this for the datasets TUSZ, TUEV, TUAR and CHB-MIT. Our assumption in deriving these reduced datasets is the following:

> The further the distance of an unlabelled timestep to the nearest label, the less likely it is that this timestep is of type 1.

The reasoning behind this is that we imagine the people who created the labels could have inspected more closely the regions in the EEG signal that are near their labels than those further away. Thus, in the reduced variants, we only select the samples that are close to (or contain) segments.

The second problem with the "background" labels, and type 1 empty segments in general, stems from the fact that they only ensure that they contain none of the event types the editors considered for this dataset. That does not say anything about event types that we may consider in different datasets. For example, TUSZ only considers seizure-related event types, but not eye movements as TUEV and TUAR do. Thus, a "background" label in a TUSZ recording does not tell us anything about eye movements within that segment. But if we train our model to predict both, seizures and eye movements, we would need to treat the empty segments from TUSZ simultaneously as type 1 w.r.t. the seizure classes and as type 2 w.r.t. the eye movement class.

Taking a broader perspective, it is clear that it does not make sense to ask the model if there are eye movements in a dataset that did not label eye movements. Thus, instead of letting the model distinguish all known event types for every dataset, we want to control in a more fine-grained way the classes the model is looking for when given an EEG sample, depending on the dataset the sample originated from. We will detail this way in the following section.

3.3 Queries

As we have seen in the above sections, the definition 3.1 of a segmentation task has two major issues:

1. The segmentation representation is only uniquely defined if none of the events considered overlap with each other.
2. Instead of segmenting all known event types all of the time, we want to segment only a select few per dataset.

An important observation is that, in practice, the `classes` in definition 3.1 do not necessarily need to have a one-to-one correspondence with event types. A simple example would be TUSZ, where the event types are 8 different types of seizures, but we may only be interested in whether in some segment there is *any* seizure or not – for example, to be able to compare the results to that of CHB-MIT, where we cannot distinguish different seizure types.

To that end, we introduce the concept of *queries*. Let `EventTypes` denote all the different event types we distinguish. Then we define a query q as a function $q : \texttt{EventTypes} \to \mathbb{N}$, that maps event types to classes. For any such a query q, we can thus instantiate a new segmentation task with $\texttt{classes} = \text{Range}(q)$.

As an example, if we are interested only in being able to distinguish seizures from no-seizure segments, the corresponding query would be defined as follows:

$$\text{seizure}(x) := \begin{cases} 1 & \text{if } x \in \{\text{Seizure, Focal Seizure, Generalized Seizure, } \dots\} \\ 0 & \text{if } x \text{ is a non-seizure related event} \end{cases}, \qquad \forall x \in \texttt{EventTypes}.$$

Another example would be sleep stages, where we would like to treat N3 and N4 phases the same and we want to ignore the movement time events (like [7]):

$$\text{sleep\_stages}(x) := \begin{cases} 1 & \text{if } x = \text{Wake} \\ 2 & \text{if } x = \text{N1} \\ 3 & \text{if } x = \text{N2} \\ 4 & \text{if } x \in \{\text{N3, N4}\} \\ 5 & \text{if } x = \text{REM} \\ 0 & \text{otherwise} \end{cases}, \qquad \forall x \in \texttt{EventTypes}.$$

For simplicity, we will assume that $\text{Range}(q) = \{0, 1, \dots, |\text{Range}(q)| - 1\}$ for all q.

We can now define queries for everything we want to know and with the classes we wish to distinguish. A list of all the queries we evaluate our model on is given in table 3.1, along with the datasets to which each query is applied and the event types belonging to each class. Note that for training, we used a few additional queries, which are described in appendix B.

Table 3.1: A list of the concrete queries we define, along with the datasets to which they are applied. For every query class, the event types assigned to this class are listed. Event types, that are not explicitly mentioned for a query, are implicitly assigned to the “no-event” class.

Query	**Datasets**	**Classes**	**Event Types**
Sleep Stages	Sleep-Cassette, Sleep-Telemetry	Wake Phase	Wake Phase
		N1 Phase	N1 Phase
		N2 Phase	N2 Phase
		N3/N4 Phase	N3 Phase, N4 Phase
		REM Phase	REM Phase
Seizure	TUSZ, CHB-MIT	Seizure	Seizure, Focal Seizure, Generalized Seizure, Simple Partial Seizure, Complex Partial Seizure, Absence Seizure, Tonic Seizure, Tonic-Clonic Seizure, Myoclonic Seizure
Multiclass Seizure	TUSZ	Focal Seizure	Focal Seizure
		Generalized Seizure	Generalized Seizure
		Simple Partial Seizure	Simple Partial Seizure
		Complex Partial Seizure	Complex Partial Seizure
		Absence Seizure	Absence Seizure
		Tonic Seizure	Tonic Seizure
		Tonic-Clonic Seizure	Tonic-Clonic Seizure
		Myoclonic Seizure	Myoclonic Seizure
Interictal Epileptiform Discharge (IED)	TUEV	IED	Spike and/or Slow Wave, Generalized Periodic Epileptiform Discharge, Periodic Lateral Epileptiform Discharge
Multiclass TUEV	TUEV	Spike and/or Slow Wave	Spike and/or Slow Wave
		Generalized Periodic Epileptiform Discharge	Generalized Periodic Epileptiform Discharge
		Periodic Lateral Epileptiform Discharge	Periodic Lateral Epileptiform Discharge
		Eye Movement	Eye Movement
		Artifact	Artifact

Table 3.1: (continued)

Binary TUAR	TUAR	Artifact	Eye Movement, Chewing, Shivers, Muscle Artifact, Electrode Pop, Electrode Artifact
Multiclass TUAR	TUAR	Eye Movement	Eye Movement
		Muscle Artifact	Muscle Artifact
		Electrode Artifact	Electrode Artifact
		Chewing	Chewing
		Shivers	Shivers
		Electrode Pop	Electrode Pop
Abnormal	TUAB	Abnormal	Abnormal
Epilepsy	TUEP	Epilepsy	Epilepsy

Further, we are now able to state exactly how the input to our model should look: For each EEG sample from a downstream dataset, we also give the model a set of queries that we would like to resolve for this sample.

CHAPTER 4

Model Architecture and Components

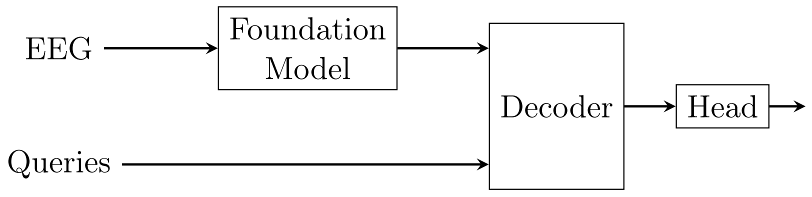


Figure 4.1

Until this point, we should have discussed everything that is not model-specific. We have defined the segmentation tasks the model should solve and we know the structure of the datasets that we will use to train the model.

Throughout this chapter, we will explain in detail and step-by-step the architecture of the model we will use, as well as many other model-related details. For reference, figure 4.1 already gives a compact overview of the final architecture. For notation, we will occasionally write $[n] = \{1, \ldots, n\}$ for $n \in \mathbb{N}_{>0}$ and $\langle x, y \rangle = \sum_{i=1}^{k} x_i \cdot y_i$ for $x, y \in \mathbb{R}^k$.

4.1 Interlude: Transformers

Before diving into describing the first part of our model, it makes sense to introduce here a central building block, called *Transformer*, that has been widely and successfully used in literature [8, 6, 7], as well as in several of the components we will use. For a thorough explanation of the inner structure of Transformers, we would refer you to the original paper introducing it [17]. Here, we would merely like to make you aware of a select few aspects of this architecture that will become important later on.

A Transformer, in the highly simplified way we will look at it, consists of a sequence of blocks that contain *attention* layers, among other things that are not relevant now. An attention layer receives 3 different input signals called *keys* $K \in \mathbb{R}^{n \times \mathrm{dim}}$, *values* $V \in \mathbb{R}^{n \times \mathrm{dim}}$ and *queries*[1] $Q \in \mathbb{R}^{\#\mathrm{queries} \times \mathrm{dim}}$ and produces

[1]Not to confuse with the queries we defined in the previous chapter. But they sometimes bear a similar meaning, as will become obvious later.

an output $Q' \in \mathbb{R}^{\#\text{queries}\times\text{dim}}$ that we can interpret as "responses" to each of the queries.

So we have #queries queries that will *attend to* n values, depending on how well a query and a key, associated with a value, match. This correlation is modelled by a dot-product over the dim-dimension. If the dot-product of a query and a key is large, we interpret this as the corresponding value being important for this query and we wish to include much information from this value in our query-response. On the other hand, if the dot-product is small, this value does not seem to be important for this query and we would prefer to ignore it. Mathematically, this is expressed as

$$Q' := \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{\text{dim}}}\right) \cdot V \in \mathbb{R}^{\#\text{queries}\times\text{dim}},$$

where the scaling factor $\frac{1}{\sqrt{\text{dim}}}$ is merely used to avoid small gradients in the softmax function for larger dim [17].

Now, where do these queries, keys and values come from? If we have a sequence n of data blocks $D \in \mathbb{R}^{n\times\text{block\_size}}$, we can let these blocks attend to themselves by defining linear maps $W^Q, W^K, W^V \in \mathbb{R}^{\text{block\_size}\times\text{dim}}$ and setting

$$Q := D \cdot W^Q \qquad K := D \cdot W^K \qquad V := D \cdot W^V.$$

This is called *self-attention*, because queries, keys and values are all derived from the same source.

Similarly, if we have two sequences of blocks $D_1 \in \mathbb{R}^{l\times\text{block\_size}_1}$ and $D_2 \in \mathbb{R}^{n\times\text{block\_size}_2}$, we can let D_1 attend to D_2 with $W^K, W^V \in \mathbb{R}^{\text{block\_size}_2\times\text{dim}}$ and $W^Q \in \mathbb{R}^{\text{block\_size}_1\times\text{dim}}$, as well as

$$Q := D_1 \cdot W^Q \qquad K := D_2 \cdot W^K \qquad V := D_2 \cdot W^V,$$

which is called *cross-attention*. A major advantage of Transformers is that the number of trainable parameters does not depend on the length of the input sequences, which helps against overfitting.

Other components of a Transformer block are a Multi-Layer Perceptron (MLP)[2] and layer normalization [18]. Additionally, if the attention layer should take the order of an input sequence into account, this needs to be done manually, using so-called "positional encodings" of each block that are added to the elements of the input sequence before the attention layer.

4.2 An EEG Foundation Model

At this point and with the knowledge of the previous sections, the first question becomes how we can extract from the EEG signal the information we need to

[2]an alternating sequence of linear layers and activation functions

answer our queries. This is not generally known to be an easy task, because EEG usually has a high signal-to-noise ratio [19], that may be even larger in our datasets that were mostly recorded in a clinical environment with diverse potential external factors like movements, pain or stress.

Moreover, ideally we would like the model to generalize from individual tasks to an *EEG representation* that allows to answer all kinds of queries. As this enables the model to pool information from multiple datasets and queries in this EEG representation, we hope it could ultimately improve the model's performance on the individual queries.

To that end, we would like to employ a foundation model for EEG. Foundation models are commonly pre-trained on a large amount of data in a self-supervised fashion (i.e. without task-specific labels) [20]. A recent example of a foundation model for EEG is LaBraM [19], which used many publicly available EEG datasets for training, including TUEG (from which many of our datasets are a subset), with promising results on several downstream tasks, including TUAB.

LaBraM takes an EEG sample of at most 16 seconds length[3], along with a list of the channels of that sample, and maps it to a 200-dimensional vector. Since the EEG sample is required to be sampled at 200 Hz, LaBraM effectively compresses the EEG signal by a factor of 16.

There is a list of standard EEG channels that LaBraM knows and was trained on. However, some of our datasets contain (exclusively) channels that are not in this list (e.g. sleep stages). Nonetheless, early experiments with permuted input channels showed no measurable impact on performance and hence, we treat channels that are unknown to LaBraM as an arbitrary subset of the LaBraM-known channels.

Internally, LaBraM uses a sequence of 2-dimensional convolution layers, followed by several Transformer blocks that perform self-attention on an EEG signal divided into blocks along the time- and channel-dimension. For our experiments, we will use the pre-trained LaBraM-Base checkpoint found in LaBraM's github repository [21].

As we have seen in section 2.1, a single event can be much longer than 16 seconds and hence, we would also like to make our samples longer than 16 seconds. We can thus not feed our entire EEG sample into LaBraM at once, but we need to slice it into 16-second-blocks that we can then let LaBraM compute on in parallel. Therefore, given an EEG sample of any length, the output of our foundation model will be a sequence of 200-dimensional vectors, one for each 16-second-block. We currently do not let these blocks overlap.

[3]Notice that this 16-second-limit is not due to LaBraM's architecture in general, but due to the limited size of the time-positional embedding in the published pre-trained version of LaBraM that we use.

4.3 Combining EEG and Queries

Where are we now? We know that we are given a set of queries and an EEG sample as input and we split this EEG sample into 16-second blocks before feeding it through LaBraM, hoping that the resulting EEG representation makes it easier to find out what is happening in the EEG. But what happens afterwards? The next goal must be to answer each query using LaBraM's EEG representation, hence, we need to combine queries and EEG representation in some way to segment the events a query asks for. Since we are now trying to extract information from the EEG representation (and our queries), we will call this combination-step "decoding" and refer to the corresponding component in our model as our *decoder*. Luckily, we are not the first ones to try to guide a model to a desired segmentation output using additional input specifying the object we are interested in.

In 2023, researchers from Facebook/Meta developed the Segment Anything Model (SAM) [22] that is capable of segmenting objects in images. Similar to us, their model does not try to identify all objects in an image simultaneously[4], but uses user-submitted "prompts" to decide for an object to segment. In contrast to our queries, these prompts often give clues about the location of the object in the image, like points or rectangles overlapping with the object in question, but free-form text prompts are also supported.

SAM's main building block to combine prompts and image is a modified version of a Transformer, which uses both self- and cross-attention between prompts and image. More precisely, a single SAM-Transformer block invokes self-attention of prompts, followed by cross-attention of prompts to the image, an MLP on the prompts and lastly cross-attention of the image to prompts. Moreover, as in the original Transformer paper [17], SAM adds a "residual connection" $\texttt{output} := \texttt{input} + \texttt{layer(input)}$ around each of these sublayers, which has been demonstrated to improve performance in deep neural networks [23]. Note also that each sublayer's output is used as an input by the next sublayer.

For our model's decoder, we adapt this SAM-Transformer to work on one-dimensional EEG, instead of two-dimensional images. Further, to be able to compute with queries, which we handled as an abstract object until now, we introduce a function mapping each query to a 200-dimensional embedding. Therefore, our decoder receives as input $\texttt{queries} \in \mathbb{R}^{\#\text{queries}\times 200}$ and $\texttt{eeg\_representation} \in \mathbb{R}^{\#\text{16s\_blocks}\times 200}$, and returns respective responses $\texttt{queries'} \in \mathbb{R}^{\#\text{queries}\times 200}$ and $\texttt{eeg\_representation'} \in \mathbb{R}^{\#\text{16s\_blocks}\times 200}$. As positional encoding for `eeg_representation` we use a fixed-length list of learnable parameters, similar to LaBraM [19]. Since the order of queries does not matter, we use no positional encoding for queries.

[4]which would be a very complex task, since there are around 100 objects per image on average in SAM's dataset

Via the SAM-Transformer, our model can process a long EEG input signal without requiring a proportional number of trainable parameters. Thus, the decoder's attention layers can collect information from the entire EEG length, which might help improve performance for long events.

After all Transformer blocks, SAM ends with an additional final attention layer. In our decoder, we replace it by a slightly different layer, whose main purpose is to expand the decoder's output dimension. Currently, the decoder's outputs are from $\mathbb{R}^{\#\text{queries}\times 200}$ or $\mathbb{R}^{\#\text{16s\_blocks}\times 200}$, so their shape depends either on the number of queries or on the length of the EEG, but not on both. However, having an output vector per query per 16-second block is desirable, because that will allow us to make "local" segment predictions based on a single vector, for each query and each 16-second block individually. So, in our final attention layer, we produce an output tensor `decoder_output` $\in \mathbb{R}^{\#\text{queries}\times\#\text{16s\_blocks}\times 200}$ in the following (slightly simplified) manner:

$$\begin{aligned}\texttt{decoder\_output}_{q,b} := \text{Attention}(&\text{queries} = \texttt{eeg\_representation'}_b,\\ &\text{keys} = \texttt{queries'}_q,\\ &\text{values} = \texttt{queries'}_q),\\ &\qquad \forall q \in [\#\text{queries}], b \in [\#\text{16s\_blocks}].\end{aligned}$$

Note that since, in this case, the attention layer's input sequences consist only of a single element, we might not even want to call this operation "attention".

4.4 Model Output Structure

We are not done yet, as `decoder_output` merely contains the information necessary to make our segment predictions, but not the predictions themselves. What is hence missing is a segment-prediction *head*, that takes as input one of the 200-dimensional vectors from `decoder_output` and outputs segments for the corresponding 16-second block and query.

While the task of our model has been well defined in chapter 3, this does not necessarily imply the same for our model's output. Consider a single query and a single 16-second block. Since the EEG was sampled at 200 Hz, our task will require us in the end to make a class-prediction for each of the 3200 time steps in that 16-second block. Even if we have only one class in that query (plus the "no-event" class), making task structure and model output identical would require our model to output 6400 values given a 200-value decoder output. We call this the *task-induced* approach, in the following.

The blowup from decoder output to model output is one problem. The other problem is the high-dimensional model output, which might be difficult to learn. A simple alternative would be to reduce the granularity of the model output, i.e.

to make not one class-prediction per timestep, but e.g. one prediction per 100 timesteps. While this approach helps with both of the above problems, it reduces the precision of the model's predictions, which introduces a structural limit on its performance that grows with decreasing model output size.

We will use a completely different approach that can reduce the model's output size while maintaining the same[5] precision as the task-induced approach. It has been introduced with DETR [24] for images and DETRtime [7], the adaptation of DETR to EEG. Here, the idea is to assume that not too many events happen within a fixed time frame. Let the maximum number of events that can take place within 16 seconds be denoted by $p \in \mathbb{N}$. Then, DETRtime outputs p segment predictions $(\texttt{class}, \texttt{center}, \texttt{width})$, where the predicted segment's center and width in the time-dimension are given by `center` and `width`. To be precise, for each prediction, DETRtime outputs these two values determining the segment's bounding-box, as well as a class-probabilities vector $\texttt{class\_prediction} \in [0,1]^{|\texttt{classes}|+1}$, with $\texttt{class} := \arg\max_{i \in \{0,\dots|\texttt{classes}|\}} \texttt{class\_prediction}_i$ used as the predicted class during evaluation. In total, the size of DETRtime's output per 16 seconds is thus $p \cdot (|\texttt{classes}| + 3)$, compared to $3200 \cdot (|\texttt{classes}| + 1)$ in the task-induced approach.

In practice, p varies between datasets and we set it differently for each query. We set it to 20 for all TUEV- and TUAR-related queries, because there, the events are very short and can occur frequently. We set $p = 5$ for seizure- and sleep-related queries, because in our data, the number of these events per 16 seconds never exceed this number. Lastly, we can set $p = 1$ for TUAB and TUEP, because, as discussed in section 2.1.4, there is always at most one segment covering the entire record (which is split across the 16-second blocks). Not counting the two bounding-box outputs, DETRtime thus reduces its output size by a factor between 160 and 3200 compared to the task-induced approach.

DETRtime's sparse outputs also allow us to significantly speed up evaluation, because we don't have to compare prediction and label for every timestep individually. Instead, we can compare multiple timesteps in constant time for each maximum overlap between prediction- and label-segments. The number of these maximum overlaps is bound by $\text{num\_predictions} + \text{num\_labels} \leq 2p$.

Notice that this type of output would even allow us to solve a more general version of the segmentation task, namely one where multiple segments (even of the same type) can overlap and the model is asked to predict individual events, instead of partitioning the time-dimension. However, it is not immediately clear how to best capture a model's performance in one or a few numbers for this kind of task, because such a number would probably need to make trade-offs between different types of errors made by the model. For example, the model could predict

[5]Strictly speaking, this depends on whether the resolution of the floating-point numbers used is high enough to represent any point in a given time frame with sufficient precision such that errors become negligible.

too few or more segments than those present in the labels, or predict segments that are shorter or longer than their peers in the labels. We would need to find cost functions for each of these types of error and weight them in some particular way, if we want to combine them. And because we anyway try to separate overlapping events through queries, we decided against such a generalization in our reported results and opted for the segmentation task definition from section 3.1, for which we can report standard measures like (macro) F1 score or balanced accuracy.

4.4.1 Multiple Heads

With the DETRtime-like outputs, our head will consist of two parts that work separately: the *class prediction head* and the *bounding-box prediction head*. Because we assume that most of the feature extraction has already happened in the foundation model and the decoder, these heads can be fairly simple. In fact, we will model them as MLPs.

Similar to SAM, DETR and DETRtime use a Transformer to perform cross-attention. Here, however, predictions take the role of prompts. Learnt embeddings for each of the p predictions are used, that together attend to the EEG blocks. But in contrast to DETR and DETRtime, our model needs to be able to answer multiple queries with a varying number of classes. Let max_classes denote the maximum number of classes among all queries. Then we can define the class prediction head as an MLP mapping a 200-dimensional input to a max_classes-dimensional output. But if we do that, we will use the same class prediction head for all queries, which implies that the head's output classes have different meanings, depending on the query. Since, in the way we have modelled it so far, the query is not explicitly given to the class prediction head, the model might start to confuse the meanings of its output classes when trained on multiple queries. Early experiments confirmed that, while the model was able to learn to predict seizures on TUSZ and sleep stages on Sleep-Cassette individually, it was not able to learn to predict both simultaneously.

In order to resolve this issue, we take inspiration CLIP's [25] contrastive learning approach, where images were assigned to corresponding textual descriptions via the cosine similarity of the respective embeddings. Similarly, we can define embeddings $\texttt{emb}_{q,c} \in \mathbb{R}^{200}$ for each class $c \in C_q$ of a particular query q and decide to which class a decoder output $\texttt{decoder\_output}_{q,b} \in \mathbb{R}^{200}$ belongs via the dot-product (unscaled cosine similarity)

$$\text{predicted\_class} := \arg\max_{c} \langle \texttt{decoder\_output}_{q,b}, \texttt{emb}_{q,c} \rangle .$$

Since this operation is equivalent to applying a linear layer (without bias) with

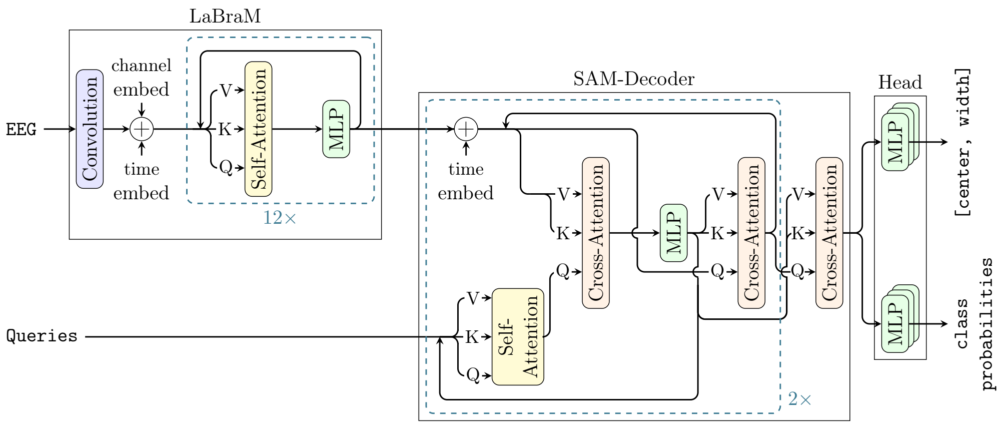


Figure 4.2: The complete architecture of our model, including the inner structure of its components.

weight matrix $W_q = (\texttt{emb}_{q,c,i})_{c\in[C_q],i\in[200]}$ and applying arg max, i.e.

$$\text{predicted\_class} := \arg\max_c (W_q \cdot \texttt{decoder\_output}_{q,b})_c,$$

we can view this as using multiple class prediction heads with parameters W_q varying between queries. In the same manner, and in order to allow the same amount of flexibility for the bounding-box predictions, we will also use a different bounding-box prediction head per query.

This completes the description of our model. A detailed view of its components and their interplay can be seen in figure 4.2.

4.5 Loss Function

Before we can start training our model, there is one open question left to answer: How do we define the loss function? We could, in theory, define a function "project" that maps the head's predictions to a partitioning of the EEG signal's time-dimension and apply a cross-entropy loss at each timestep. However, the outputs of this project-function do not explicitly distinguish anymore between class and position parameters of a segment, in contrast to the inputs. Hence, there may be a very complex relation between inputs and outputs, which might be difficult to learn for the model. Also, as we have already remarked in section 2.3, this timestep-wise comparison is costly since there are hundreds of millions of these timesteps in our datasets. In contrast, if we can limit the runtime by the number of predictions p and the number of events (which almost never exceed 20 per 16s), our loss computation becomes much faster.

To that end, let us consider a single 16-second block b and a single query q, that distinguishes C_q classes and suppose p again denotes the number of predictions. Let `class_predictions` $\in \mathbb{R}^{p \times C_q}$ be the corresponding output of our class prediction head and let `box_predictions` $\in \mathbb{R}^{p \times 2}$ denote the output of our bounding-box prediction head. Recall that the labels in our downstream datasets are given in a very similar format as these outputs of our head, namely as a list of events $e = (\text{type}, \text{start}, \text{stop}) \in \texttt{instances}_b$. Here, $\texttt{instances}_b$ shall denote only the events overlapping with the 16-second block b of the EEG sample we are currently looking at. Let $l := |\texttt{instances}_b|$ denote the number of labels for this block.

The central idea, that allows us to directly work on these p predictions and l labels, is to assign predictions to labels and, for each assigned prediction, compute a (local) loss based on how well it matches with the assigned label, in terms of location and type/class. Notice that, as long as this local loss is differentiable, we can backpropagate the error through our model. This fact is completely independent of the differentiability and complexity of the algorithm computing these assignments.

There are different ways to define this assignment algorithm. A perhaps straightforward way would be to assign each prediction to the label that fits it best, i.e. that produces the minimum local loss for the prediction. However, this could imply that some labels can be completely ignored throughout the entire training phase, because the model does not gain anything from predicting them, as long as it is good at predicting a subset of the labels. Suppose, for example, that in in our data, there is always a label of some type A, that is somehow easier to predict than a label of type B, which is also present in every 16-second block. Then our model might get stuck in the local (and global) minimum of only ever assigning predictions to events of type A and learns nothing about type B.

Even if we force every label to be assigned a prediction (if possible), our current strategy still has a major problem with blocks where there is no label. Because, naturally, we want to train our model to not predict anything in this case. To that end, we have to define a dedicated "no-prediction" class that our model's head can output. With this class, our model gains the ability to output any number of segments $p' \in \{0, \dots, p\}$, instead of just p. Yet, the importance of this "no-prediction" class for the model during training now entirely depends on the number of empty 16-second blocks in our data, which is problematic, since we did not make any assumption about the presence and frequency of such blocks. Them having such a huge influence on our model's behavior is thus not acceptable for us.

4.5.1 Simple Matching Assignment

While we decide to keep this "no-prediction" class, we try to make its effect on our model's predictions more dynamic. Instead of assigning every prediction to a class, causing potentially multiple predictions being assigned to the same label, we compute a bipartite matching between predictions and labels. Since the model's outputs are ordered and we can order the labels along the time-dimension (assuming they don't overlap), we could simply match the i-th prediction with the i-th label, for all $i \in \min\{p, l\}$. To better account for the "no-prediction" class, we can then tell the model that all of the *unmatched* predictions should predict the "no-prediction" class. Now, how often we ask the model to predict the "no-prediction" class, does not depend on the number of empty 16-second blocks anymore, where we had

$$\#\text{no\_predict} = \begin{cases} p & \text{if block empty} \\ 0 & \text{otherwise} \end{cases}.$$

Instead, it now depends on the number of labels per block, i.e.

$$\#\text{no\_predict} = p - l.$$

Despite the fact that this simple approach destroys the potentially useful feature of the model's invariance to the order of its outputs, we would still like to discuss two properties that may become relevant for future research into these types of loss functions.

First, with the above approach, in order to learn which label to predict, it is imperative that the i-th prediction has some information about the labels $1, \ldots, i-1$ to which its predecessors, predictions $1, \ldots, i-1$, will be assigned. Otherwise, it will be impossible to identify the i-th label among the others. While this may, at first, seem like an unnecessary and hard to achieve requirement of this simple approach, caused by the strict adhesion to a fixed order of the labels, it turns out that, in any matching-based assignment algorithm, some awareness of an individual prediction about the other labels and the predictions that get assigned to them, is always necessary. Because, if there was no such awareness, each prediction would always predict the same label. It would hence be important to find out to what degree this inherent complexity of matching-based assignments could be responsible for performance limits of today's segmentation models.

The other interesting property of the simple approach is that, by dropping the output-order invariance feature, it allows for a certain degree of specialization among different prediction outputs. In this case, for example, prediction 1 might learn to focus its attention on early parts of the EEG sample, while prediction p might instead focus more on patterns occurring at the end of the sample's time-dimension, simply because the labels they will get assigned to lie in those areas most often. This specialization could be extended by assigning particular predictions only to labels of a specific class. Since the generalization of the model should happen in the decoder and not in the head, specialization of different heads for specific classes might help to further improve our model's performance in the future, similar to the specialization of heads with respect to queries.

4.5.2 Hungarian Loss

Despite the potential advantages of a non-invariant assignment algorithm highlighted in the previous section, we decided to side with the better-studied "Hungarian loss" function that is also used in DETR and DETRtime [24, 7]. The idea is essentially to let the model learn from the most favorable interpretation of its outputs, namely by finding a matching that minimizes the sum of the local losses it induces. Formally, this is called a minimum weight perfect matching in a bipartite graph, which can be solved in $\mathcal{O}((p+l)^3)$ time using a variant [26] of the Hungarian method [27] (hence the name of the loss).

As mentioned previously, we want to define the local loss between a prediction and a label based on differences in location and size, as well as the predicted and actual class. Consider the i-th prediction and the j-th label. Let

$$\begin{aligned}
\text{class}_l &:= (\texttt{instances}_b)_{j,1} \in \{0, \ldots, C_q\} \\
\text{center}_l &:= (\texttt{instances}_b)_{j,2} \in \mathbb{R} \\
\text{width}_l &:= (\texttt{instances}_b)_{j,3} \in \mathbb{R}
\end{aligned}$$

denote the label's actual class, center and width. Similarly, let

$$\begin{aligned}
\text{center}_p &:= \texttt{box\_predictions}_{i,1} \in \mathbb{R} \\
\text{width}_p &:= \texttt{box\_predictions}_{i,2} \in \mathbb{R}
\end{aligned}$$

denote the prediction's center and width and recall that its class predictions are denoted by $\texttt{class\_predictions}_i \in \mathbb{R}^{C_q}$. We adopt the local loss definition from DETR, where the loss has three components:

1. The *class-loss*, defined as $-\texttt{class\_predictions}_{i,\text{class}_l}$
2. The *L1-loss*, defined as $|\text{center}_p - \text{center}_l| + |\text{width}_p - \text{width}_l|$
3. The Intersection over Union (or short: *IoU-loss*), defined as $\frac{|P \cap L|}{|P \cup L|}$, where $P, L \subseteq \mathbb{R}$ denote the intervals covered by prediction i and label j, respectively. Note that DETR actually uses a "generalized" IoU-variant [28] that, however, turns out to be equivalent to IoU for one-dimensional data. In our implementation, we compute the loss as

$$\frac{|P \cap L|}{|P \cup L|} = \frac{\min\{\text{ReLU}(y_p - x_l), \text{ReLU}(y_l - x_p)\}}{\max\{\text{ReLU}(y_p - x_l), \text{ReLU}(y_l - x_p)\}},$$

where $x_h := \text{center}_h - \text{width}_h/2$ and $y_h := \text{center}_h + \text{width}_h/2, \quad h \in \{p, l\}$ denote the borders of the intervals P and L, respectively, and $\text{ReLU}(a) := \max\{0, a\}$.

We hence use the sum of these three losses (with individual scaling factors $\lambda_{\text{class}} = 1, \lambda_{L1} = 10, \lambda_{IoU} = 2$, adopted from [7]) as weights to compute our minimum weight perfect matching. Once we have computed this matching, we add the weight of each matched prediction-label pair to our final loss, except that for the final loss, the class-loss is replaced by a cross-entropy loss:

$$-\log \frac{\exp(\texttt{class\_predictions}_{i,\text{class}_l})}{\sum_{c=0}^{C_q} \exp(\texttt{class\_predictions}_{i,c})}.$$

Additionally, as we had discussed in section 4.5.1, for the unmatched predictions, we also add a class-loss to our final loss, but with $\text{class}_l := 0$, where 0 stands for the "no-prediction" class. Recall that class 0 also stood for the "no-event" class in our segmentation task definition of section 3.1. Thus, in this work, we merge these two different auxiliary classes into a single implemented class 0. It is not necessary to merge the two classes. In fact, it might even improve the model's performance to treat them separately, as it may help the model not to confuse them.

4.6 Sampling and Loss Scaling

As we have seen in section 3.3, the frequency of occurrence of different event types in our datasets is highly imbalanced. Because of this, we observed our model to severely struggle at predicting rarely appearing event types, if we do not employ measures to counteract this imbalance.

Our first measure, which was already mentioned in section 2.3, is to use weighted sampling according to a multinomial distribution. To that end, we assign a weight to each sample in a dataset. Let $k \in \mathbb{N}$ be the number of samples of a dataset. We define the weight of a sample $i \in [k]$ as follows:

Let there be e different event types in this dataset. Let $n_1, \ldots, n_e \in \mathbb{N}$ denote the number of segments of each of these event types in the entire dataset. Further, let $N := \sum_{j=1}^{e} n_j$ and for the i-th sample, let $i_1, \ldots, i_e$ denote the number of segments of each event type within the i-th sample. Then, the i-th sample's weight is defined as

$$\text{weight}_i := \begin{cases} \sum_{j=1}^{e} i_j \cdot \frac{N}{n_j} & \text{if } \sum_{j=1}^{e} i_j > 0 \\ 1 & \text{otherwise} \end{cases}$$

Prior to using these weights as probabilities for the multinomial distribution, we of course normalize them, so that they sum up to 1. The factor $\frac{N}{n_j}$ ensures that we add the most weight for the least frequent event type. The factor i_j yields more weight to samples containing many events of rare types. Empirically, through these weights we obtained relatively balanced datasets, drawing $2 \cdot k$ training samples from this multinomial distribution.

In order to define the reduced variant[6] of several of our datasets, it now suffices to set the weights to 0 for all samples that we deem too far away from samples containing segments. In practice, we decide to add samples to the reduced dataset in increasing order of their distance to the nearest sample containing segments. We stop this process once the sum of the lengths of all empty segments becomes larger than the size of the region covered by the average event type.

Note, of course, that this weighted sampling is only applied to our training datasets and not to the evaluation datasets. Also, we did not apply weighted sampling on TUAB, because it is already almost perfectly balanced between abnormal and normal samples.

Because weighted sampling only helps in balancing the datasets, but is not capable of completely eliminating imbalances, we additionally employ class-weights in our cross-entropy class-loss. Specifically, for each class $c \in \{0, \ldots, C_q\}$ of a query, we define class-weights $w_c \in \mathbb{R}$, such that the class loss becomes

$$-w_{\text{class}_l} \cdot \log \frac{\exp(\texttt{class\_predictions}_{i,\text{class}_l})}{\sum_{c=0}^{C_q} \exp(\texttt{class\_predictions}_{i,c})}.$$

We determined these class weights empirically. The exact weights used for each class of each query can be found in table A.1 in the appendix.

[6]The notion of reduced variants of our datasets was introduced in section 3.2.

CHAPTER 5

Experiments and Results

5.1 Experiment Structure

Now that we know everything we need about the model, our datasets and the task definition, we are ready to run our experiments. As previously announced, we conduct two different experiments: On the one hand, the model will train on each of our eight datasets separately (*separate runs*). On the other, it will train on all eight datasets simultaneously (*combined run*). In total, we executed nine separate runs, because we evaluated the unreduced as well as the reduced version of TUSZ. In each run, the model was trained for 200 epochs, except for the unreduced TUSZ (100 epochs) and the combined run (150 epochs), because of time constraints. The initial learning rate was set to 10^{-6}, but after 150 epochs, it was reduced by a factor of 10 every 5 epochs (except for unreduced TUSZ, where this process already started after 85 epochs). For optimization, we used the AdamW optimizer [29] with $(\beta_1, \beta_2) = (0.9, 0.999), \epsilon = 10^{-8}$ and a 10^{-4} weight decay. During training, we also used a 0.25 dropout layer between decoder and head.

Because during earlier tests on the complete TUEV and TUAR and CHB-MIT datasets, the model struggled to learn anything, we thought that this might either be due to the high class imbalance in these datasets or due to the possibility that not all EEG regions in these datasets have been completely labelled. Thus, here we only ran experiments on the reduced versions of these datasets, where we are more confident about the completeness of the labels. It should be noted that in these reduced versions also the evaluation datasets have been subject to reduction and are therefore distributed differently than the original datasets. Hence, this should be taken into account when comparing our results with those of other works. In the combined run, we trained on the reduced versions of TUSZ, TUEV, TUAR and CHB-MIT.

For each class of each of the queries listed in table 3.1 of section 3.3, we count, on the timestep level, the true/false positives/negatives, respectively, of the model's predictions compared with the segment representation of the actual labels from the task definition 3.1. From these counters, we derive and report the F1 score and balanced accuracy for each query-class. For reference, these metrics are defined as follows:

$$\text{F1 Score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}, \qquad \text{Balanced Accuracy} = \frac{\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}}{2},$$

where TP, FN, FP and TN are the counters of true positive, false negative, false positive and true negative predictions, respectively. Additionally, we report the macro F1 score and the (multi-class) balanced accuracy for each query. The macro F1 score is computed as the average of the individual classes' F1 scores. The multi-class balanced accuracy is defined as

$$\frac{1}{C_q} \sum_{i=1}^{C_q} \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

where TP_i, FN_i are the true positives/false negatives of the i-th class of query q that defines C_q classes in total.

Because we use LaBraM for our initial EEG embeddings, we have a separation of a given sample into blocks of 16 seconds length throughout our model pipeline. Ultimately, as described in section 4.4.1, our heads make predictions for each of these blocks separately. In order for our model to also predict longer events, during evaluation we add a post-processing step of merging predictions from neighbouring 16s-blocks if they are of the same predicted class and the gap between them is shorter than 0.8 seconds.

TUAB and TUEP constitute an exception for our score calculations, because they are given as classification datasets. Thus, to allow comparison with other (classification) approaches evaluated on these datasets, we map our model's output for a given record to the "abnormal"/"epilepsy" class if the sum of the lengths of the "abnormal"/"epilepsy" segments predicted by the model is larger than the record length divided by two. Otherwise, we map its output to the "no-event" class. For these datasets, we hence report results based on the correct/incorrect predictions on the record level. The TUAB dataset contains 150 records labelled as normal and 126 labelled as abnormal. For TUEP, there exist 118 non-epileptic and 401 records with epilepsy.

Experiments were run on python version 3.11.6, along with python modules pytorch version 2.5.1, cuda-runtime version 11.8.0, scipy version 1.14.1, numpy version 1.26.4, timm version 1.0.11 and h5py version 3.9.0. We used seed 4321 to control the randomness in pytorch, numpy and the python random module. To speed up our training process, we distributed training among up to 6 Nvidia A100 GPUs using torchrun. In total, all training runs together took about 1680 GPU hours. Table 5.1 details the batch-size and sample length used for each dataset.

Dataset	**Batch Size**	**Sample Length**
Sleep-Cassette	128	19200
Sleep-Telemetry	64	19200
TUSZ	128	6400
CHB-MIT	32	6400
TUEV	256	3200
TUAR	256	3200
TUAB	64	19200
TUEP	64	19200

Table 5.1: Batch-sizes and sample lengths of each dataset used during training.

5.2 Results

The majority of our results is shown in table 5.2 that compares the model's performance on each query between the combined run and the corresponding separate run. If not explicitly states otherwise, the numbers in this section refer to the model's performance evaluated after the last epoch of its training.

Table 5.2: A comparison of the results from the separate runs with that from the combined run.

Dataset	**Query**	**Class**	**Separate Runs**		**Combined Run**	
			F1 Score	**Balanced Accuracy**	**F1 Score**	**Balanced Accuracy**
Sleep-Cassette	Sleep Stages	Wake Phase	0.962	0.933	0.969	0.956
		N1 Phase	0.213	0.568	0.299	0.614
		N2 Phase	0.756	0.897	0.760	0.875
		N3/N4 Phase	0.757	0.855	0.754	0.913
		REM Phase	0.529	0.705	0.614	0.796
		(macro) F1 & Balanced Accuracy	**0.644**	**0.628**	**0.679**	**0.697**
Sleep-Telemetry	Sleep Stages	Wake Phase	0.675	0.843	0.687	0.863
		N1 Phase	0.398	0.705	0.321	0.614
		N2 Phase	0.764	0.788	0.788	0.800
		N3/N4 Phase	0.685	0.883	0.703	0.887
		REM Phase	0.712	0.834	0.752	0.879
		(macro) F1 & Balanced Accuracy	**0.647**	**0.702**	**0.650**	**0.694**

Table 5.2: (continued)

TUSZ (reduced)	Seizure	No-event	0.479	0.635	0.038	0.500
		Seizure	0.806	0.635	0.800	0.500
		(macro) F1 & Balanced Accuracy	**0.642**	**0.635**	**0.419**	**0.500**
	Multiclass Seizure	No-event	0.499	0.633	0.450	0.578
		Focal Seizure	0.539	0.626	0.000	0.500
		Generalized Seizure	0.314	0.563	0.195	0.464
		Complex Partial Seizure	0.015	0.491	0.152	0.534
		Absence Seizure	0.000	0.500	0.015	0.504
		Tonic Seizure	0.000	0.500	0.000	0.489
		Tonic-Clonic Seizure	0.218	0.565	0.018	0.473
		(macro) F1 & Balanced Accuracy	**0.226**	**0.222**	**0.119**	**0.149**
CHB-MIT (reduced)	Seizure	No-event	0.648	0.640	0.470	0.639
		Seizure	0.629	0.640	0.756	0.639
		(macro) F1 & Balanced Accuracy	**0.638**	**0.640**	**0.613**	**0.639**
TUEV (reduced)	Interictal Epileptiform Discharge	No-event	0.839	0.581	0.720	0.559
		IED	0.320	0.581	0.316	0.559
		(macro) F1 & Balanced Accuracy	**0.580**	**0.581**	**0.518**	**0.559**
	Multiclass TUEV	No-event	0.739	0.546	0.630	0.542
		Spike and/or Slow Wave	0.016	0.508	0.015	0.517
		Generalized Periodic Epileptiform Discharge	0.208	0.562	0.000	0.500
		Periodic Lateral Epileptiform Discharge	0.187	0.552	0.109	0.508
		Eye Movement	0.107	0.568	0.064	0.549
		Artifact	0.222	0.594	0.056	0.474
		(macro) F1 & Balanced Accuracy	**0.246**	**0.259**	**0.146**	**0.186**

Table 5.2: (continued)

TUAR (reduced)	Binary TUAR	No-event	0.358	0.584	0.462	0.508
		Artifact	0.770	0.584	0.526	0.508
		(macro) F1 & Balanced Accuracy	**0.564**	**0.584**	**0.494**	**0.508**
	Multiclass TUAR	No-event	0.290	0.565	0.508	0.504
		Eye Movement	0.336	0.601	0.000	0.500
		Muscle Artifact	0.493	0.645	0.263	0.502
		Electrode Artifact	0.396	0.650	0.039	0.500
		Chewing	0.331	0.750	0.019	0.505
		Shivers	0.000	0.499	0.000	0.500
		Electrode Pop	0.000	0.500	0.000	0.500
		(macro) F1 & Balanced Accuracy	**0.264**	**0.315**	**0.118**	**0.146**
TUAB	Abnormal	No-event	0.868	0.853	0.828	0.837
		Abnormal	0.839	0.853	0.832	0.837
		(macro) F1 & Balanced Accuracy	**0.854**	**0.853**	**0.830**	**0.837**
TUEP	Epilepsy	No-event	0.545	0.696	0.610	0.738
		Epilepsy	0.893	0.696	0.899	0.738
		(macro) F1 & Balanced Accuracy	**0.719**	**0.696**	**0.755**	**0.738**

The main observation from this table is that the performance of the combined run does *not* significantly improve over that of the separate runs, except for slight increases in Sleep-Cassette and TUEP. Instead, the jointly trained model performed far worse for TUSZ, TUEV and TUAR.

In the case of TUEV and TUAR this might be explainable by the fact that these datasets are a lot smaller than for example Sleep-Cassette or TUAB. Hence, the model might find it easier to ignore these datasets and accept the (in dataset-dimensions) slightly increased loss, in order to focus more on bigger datasets.

However, if it is indeed the case that the model focuses on larger datasets, then CHB-MIT, with its rather stable performance, is an outlier, as its reduced version is almost as small as the reduced TUEV, with even fewer samples in CHB-MIT than in TUEV. It could be, though, that CHB-MIT still profited from the seizures learnt from TUSZ, despite the bad performance of the latter.

Another explanation for the low performance on all multiclass tasks might be the relatively small amount of event segments, which we have already seen in table 2.3. While insufficient data can be generally considered a major issue in the EEG deep learning domain, this issue could have been aggravated by the fact that we are attempting to solve a harder task than just classification. For example, as we have already discussed, in TUEV, we cannot restrict ourselves to only using

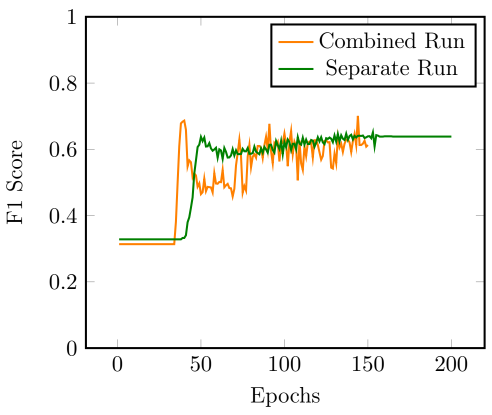


Figure 5.1: F1 score development of Seizure query for CHB-MIT in both, the combined and the separate run.

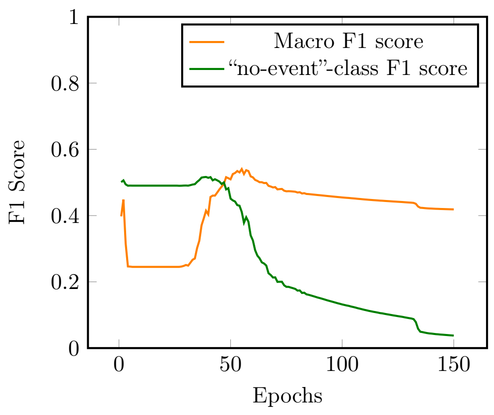


Figure 5.2: F1 score development of Seizure query for TUSZ in the combined run.

EEG data completely covered by labels, in stark contrast to previous approaches like [19], that hence achieve much higher performance. More importantly, since [19] also used LaBraM to achieve these higher scores, we should rethink our assumption that unlabelled data close to labels is probably event-free. For instance, it might not even be the case that the labels indicate the start and end points of an event, but actually rather mean something like: "there is an event of this type happening somewhere within this label-segment". In this case, our segmentation approach might not stand much of a chance, since the label boundaries would be very imprecise.

Unlike TUEV, it should be noted that our performance for TUAB is very comparable to (and slightly exceeds) that of 0.826 balanced accuracy in [19]. Similarly, we will see in table 5.3 that our separate run on the unreduced TUSZ dataset slightly exceeds the performance of [6]. When comparing these results, one should take into account that [6] use a 75% overlap of seizure samples in their evaluation dataset, which results in a much more balanced evaluation set in their case.

An interesting observation with respect to CHB-MIT is that the model starts predicting seizures only after around epoch 40, both in the separate and in the combined run (figure 5.1). It seems that now, in the combined run, TUSZ has inherited this issue, as its Seizure-query F1 score expansion in figure 5.2 shows. In contrast to CHB-MIT, however, the F1 score of TUSZ starts sinking after epoch 60, coinciding with a near-zero sinking F1 score of the "no-event" F1 score of that query, indicating that the model started predicting seizures all the time. This could be a sign of insufficiently calibrated loss class weights or of other queries starting to negatively influence the seizure query.

It is worth mentioning that our loss class weights cannot counter all class

Query	**Class**	**Reduced TUSZ** F1 Score	**Reduced TUSZ** Balanced Accuracy	**Unreduced TUSZ** F1 Score	**Unreduced TUSZ** Balanced Accuracy
Seizure	No-event	0.479	0.635	0.976	0.714
	Seizure	0.806	0.635	0.536	0.714
	(macro) F1 & Balanced Accuracy	**0.642**	**0.635**	**0.756**	**0.714**
Multiclass Seizure	No-event	0.499	0.633	0.975	0.825
	Focal Seizure	0.539	0.626	0.453	0.702
	Generalized Seizure	0.314	0.563	0.216	0.572
	Complex Partial Seizure	0.015	0.491	0.016	0.504
	Absence Seizure	0.000	0.500	0.000	0.500
	Tonic Seizure	0.000	0.500	0.000	0.500
	Tonic-Clonic Seizure	0.218	0.565	0.000	0.500
	(macro) F1 & Balanced Accuracy	**0.226**	**0.222**	**0.237**	**0.223**
	Number of Samples		17002		227550

Table 5.3: A comparison of the performances of the separate reduced vs. unreduced TUSZ runs.

imbalances if the same query is applied to multiple datasets with different class imbalances, since the weights are only defined once per query class. This might be an important limitation to extending this model to other datasets about similar event types.

In table 5.3, we compare the performances of the reduced and unreduced TUSZ-only runs. It should be clear that the F1 score on the "no-event" class can expected to be higher for the unreduced TUSZ, since this class occurs in unreduced TUSZ much more frequently than in reduced TUSZ, which pushes up its F1 score, even if one assumes equal recall of the model for both dataset versions. Nevertheless, the macro F1 score and balanced accuracy of the Seizure query show that the model trained on the unreduced TUSZ overall outperforms the model trained on reduced TUSZ. Hence, it would seem that working with a reduced version of TUSZ is unnecessary.

We would also like to add that the slight improvement of the combined run over the separate run for Sleep-Cassette must be taken with caution, as earlier tests on a slightly different model version showed better performance in all phases except the Wake phase,[1] coming close to the performance reported by [7]. It is

[1]The classes' F1 scores we achieved were the following. Wake Phase: 0.972, N1 Phase: 0.416, N2 Phase: 0.806, N3/N4 Phase: 0.832, REM Phase: 0.722.

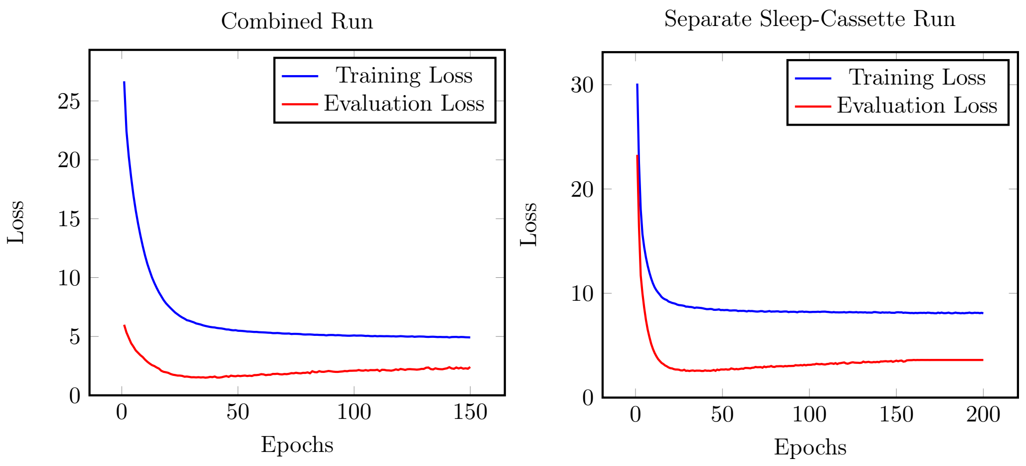


Figure 5.3

unclear whether the small modifications between the test model and the current model (e.g. swapping the direction of attention in the final decoder layer) contributed to this decrease in performance or whether it hints at a more general problem with the stability of our model.

5.3 Loss Discussion

A worrying trend in some of our results is the development of the evaluation loss. While expected to decrease over the epochs, it can be seen in figure 5.3 that at first, it decreases, but then after a few epochs, it starts increasing again. This trend was observed for the individual runs of Sleep-Cassette, unreduced TUSZ, TUAB and TUEP, as well as the combined run. Such behavior can be caused by a too high learning rate. However, we have set our learning rate already quite low, to 10^{-6} (compared to, e.g. 10^{-4} in [7]). It may be worth trying even smaller learning rates, but it is also possible that this problem is not caused by the learning rate.

Interestingly, the evaluation losses of reduced TUSZ and Sleep-Telemetry did not show such trends. Neither did an earlier experiment with Sleep-Cassette, with a slightly different architecture that, as mentioned, also showed better results. The fact that this loss-increase happened only for this Sleep-Cassette run, but not during the earlier one and neither in Sleep-Telemetry, might hint again at some instability of our approach, which could cause the model to perform very differently under small changes.

Notably, both TUAB and TUEP, the record-classification datasets, are affected. Hence, this could also point to the choice we made, to model the “epilepsy” and “abnormal” conditions as segments covering the entire record, as having a

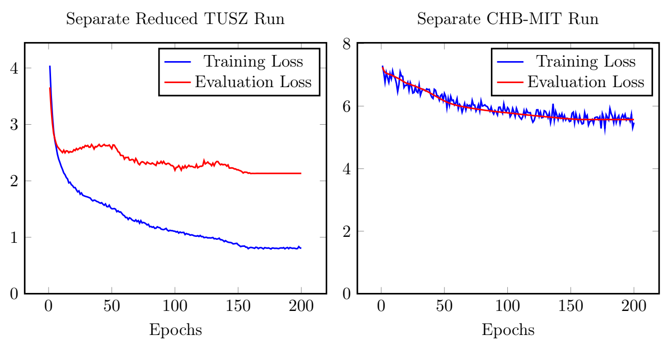


Figure 5.4

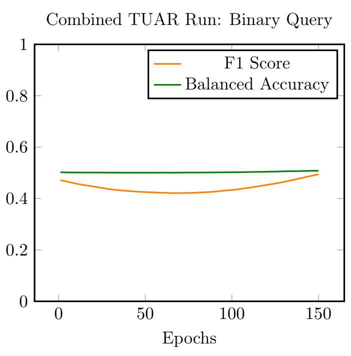


Figure 5.5

negative influence on the model's learning behavior. In this context, it could perhaps prove beneficial to take a little step back from the "everything is a segment"-paradigm and predict conditions like epilepsy as classes without temporal relations. One could adapt the Hungarian loss to reflect this, by ignoring the L1- and IoU-parts for particular queries, like "Epilepsy" and "Abnormal", both during the matching and for the final loss computation.

Despite this evaluation-loss behavior, we have already seen that the score of TUAB is still comparable with results found in [19]. Also, except for Sleep-Cassette, the increasing evaluation loss did not measurably decrease the macro F1 scores in any of our runs.

Lastly, it should be noted that the evaluation losses of CHB-MIT, TUEV and reduced TUSZ could maybe have further decreased, had we let the training continue for longer. This can be seen particularly well for reduced TUSZ in figure 5.4, where both training and evaluation loss bend around epoch 150 due to the starting decrease of the learning rate at this point. Similarly, the F1 scores of some queries on TUSZ, CHB-MIT, TUEV and TUAR were still increasing at epoch 150 in the combined run. Most clearly, this can be see in for the TUARBinary-query in figure 5.5. Hence, these datasets' F1 scores may have improved further, if the model had been trained for longer.

Plots of the losses of all runs can be found in appendix C, along with the F1 score and balanced accuracy expansion of every run (appendix D) and extended performance tables also showing precision and recall for all queries and all datasets (appendix B).

Chapter 6

Future Work

Overall, it must be said that our approach did not work as well as we expected and appears to have substantial problems in terms of generalization capabilities and even with respect to the single-task performance on some datasets.

The reasons for the performance issues and the lack of improvement over single-task performances may certainly relate to some yet unfound bug in our code or be due to our model architecture not yet fitting the task well enough. For example, it could be that the 12 LaBraM transformer layers give the model too many parameters, so that it learns each task separately instead of having to generalize. On the other hand, it could be that the two Transformer layers in our decoder are not enough to handle nine different queries with in total 35 different classes.

Also, the varying size of the source datasets could be an important factor. Instead of passing through each dataset about 2 times per epoch,[1] it might help to counteract such dataset-size imbalances by training on an equal total length of samples per epoch, for all datasets.

But it could likewise be the case that we simply have not enough data for our model to meaningfully generalize something from it. This may be true both in terms of the size of the datasets for a specific query, as well as concerning the diversity of event types we can learn to distinguish. More work may also be done to better represent time-independent conditions like "epilepsy" in our segmentation task, as proposed in section 5.3.

One could also take a very different direction on the model architecture side. For instance, similar to [9], it might provide further generalization potential to the model if the EEG representations get aligned with text tokens describing the segments contained in them, in order to allow for more verbose instruction tuning in the sense of [30].

A recurring issue of the Hungarian loss function from section 4.5.2 is this artificial "no-prediction" class of unmatched predictions, since increasing the number of predictions p will cause more predictions to be unmatched and forced to pre-

[1]the factor of two being due to our weighted sampling

dict this class over the training process. This can result in an imbalance between this “no-prediction” class and the other “real” classes and, if not counteracted by appropriate class-loss weighting, can entail a model being stuck in the local minimum of always predicting the “no-prediction” class. Addressing this problem might have been impeded by our decision to merge this class with the “no-event” class from our segmentation task.

However, in order to completely resolve this issue, a very different loss function would be needed. For instance, in contrast to the Hungarian loss, the Segment Anything Model’s loss does not require it to predict all objects in an image simultaneously. Instead, the given prompts are assumed to be sufficiently restrictive for the network to only need to return three possible candidate objects.

The drawback of this approach is that these prompts, depending on the rigor with which the training set was created, can give the model already some prior information about the interesting regions of an image. While it may be natural to assume in the case of images that the person wanting to segment the image already has clues about the locations of the objects in that image, this is typically not the case in the EEG analysis application areas we address. There, the question of which parts of an EEG are interesting, is to be answered by the model, not its user.

Nevertheless, it could still be possible to integrate the SAM-loss approach into our domain, for instance by making the model’s information processing pipeline more recurrent. It might be able to learn to *produce* these prompts, i.e. guesses where in the EEG something interesting could happen, by itself and for itself, perhaps even by iteratively increasing the granularity of its predictions in the process.

Bibliography

[1] L. R. Krol, "Eeg electrode positions in the 10-10 system using modified combinatorial nomenclature, along with the fiducials and associated lobes of the brain." 2020. [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=96859272

[2] X. Ma, S. Qiu, C. Du, J. Xing, and H. He, "Improving eeg-based motor imagery classification via spatial and temporal recurrent neural networks," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 1903–1906.

[3] J. Dellabadia Jr, W. L. Bell, J. W. Keyes Jr, V. P. Mathews, and S. S. Glazier, "Assessment and cost comparison of sleep-deprived eeg, mri and pet in the prediction of surgical treatment for epilepsy," *Seizure*, vol. 11, no. 5, pp. 303–309, 2002.

[4] K. P. Ayodele, W. O. Ikezogwo, M. A. Komolafe, and P. Ogunbona, "Supervised domain generalization for integration of disparate scalp eeg datasets for automatic epileptic seizure detection," *Computers in Biology and Medicine*, vol. 120, p. 103757, 2020.

[5] Y. Li, Y. Liu, W.-G. Cui, Y.-Z. Guo, H. Huang, and Z.-Y. Hu, "Epileptic seizure detection in eeg signals using a unified temporal-spectral squeeze-and-excitation network," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 4, pp. 782–794, 2020.

[6] Y. Zhu and M. D. Wang, "Automated seizure detection using transformer models on multi-channel eegs," in *2023 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*. IEEE, 2023, pp. 1–6.

[7] L. Wolf, A. Kastrati, M. B. Plomecka, J.-M. Li, D. Klebe, A. Veicht, R. Wattenhofer, and N. Langer, "A deep learning approach for the segmentation of electroencephalography data in eye tracking applications," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 912–23 932.

[8] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[9] W.-B. Jiang, Y. Wang, B.-L. Lu, and D. Li, "Neurolm: A universal multi-task foundation model for bridging the gap between language and eeg signals," *arXiv preprint arXiv:2409.00101*, 2024.

[10] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the eeg," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[11] V. Shah, E. Von Weltin, S. Lopez, J. R. McHugh, L. Veloso, M. Golmohammadi, I. Obeid, and J. Picone, "The temple university hospital seizure detection corpus," *Frontiers in Neuroinformatics*, vol. 12, p. 83, 2018.

[12] I. Obeid and J. Picone, "The temple university hospital eeg data corpus," *Frontiers in Neuroscience*, vol. 10, p. 196, 2016.

[13] J. Guttag, "Chb-mit scalp eeg database," PhysioNet, 2010. [Online]. Available: https://doi.org/10.13026/C2K01R

[14] S. Ferrell, V. Mathew, M. Refford, V. Tchiong, T. Ahsan, I. Obeid, and J. Picone, "The temple university hospital eeg corpus: Electrode location and channel labels," 2022. [Online]. Available: https://isip.piconepress.com/publications/reports/2020/tuh_eeg/electrodes/electrodes_and_channels_v30.docx

[15] *HDF5 File Format Specification Version 3.0*, HDF Group.

[16] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on Job Scheduling Strategies for Parallel Processing.* Springer, 2003, pp. 44–60.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[19] W. Jiang, L. Zhao, and B.-l. Lu, "Large brain model for learning generic representations with tremendous eeg data in bci," in *The Twelfth International Conference on Learning Representations*, 2024.

[20] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[21] W. Jiang, L. Zhao, and B.-l. Lu, "Labram," Github Repository, 2024. [Online]. Available: https://github.com/935963004/LaBraM

[22] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *2023*

IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2023, pp. 3992–4003.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2016, pp. 770–778.

[24] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision.* Springer, 2020, pp. 213–229.

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning.* PMLR, 2021, pp. 8748–8763.

[26] N. Tomizawa, “On some techniques useful for solution of transportation network problems,” *Networks*, vol. 1, no. 2, pp. 173–194, 1971.

[27] H. W. Kuhn, “The hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[28] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2019, pp. 658–666.

[29] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[30] H. Liu, C. Li, Q. Wu, and Y. J. Lee, “Visual instruction tuning,” *Advances in Neural Information Processing Systems*, vol. 36, 2023.

Appendix A

Query Class Weights

Table A.1: A complete list of the queries our models were trained on, along with their classes and loss-class-weights (see section 4.6 for the definition and usage of these weights).

Query	Class	Weight
Sleep Stages	No-event	0.40
	Wake Phase	0.50
	N1 Phase	4.00
	N2 Phase	1.70
	N3/N4 Phase	6.31
	REM Phase	4.00
Seizure	No-event	1.00
	Seizure	1.50
Multiclass Seizure	No-event	1.00
	Focal Seizure	1.73
	Generalized Seizure	10.17
	Simple Partial Seizure	1.00
	Complex Partial Seizure	12.24
	Absence Seizure	6459.02
	Tonic Seizure	70.67
	Tonic-Clonic Seizure	40.58
	Myoclonic Seizure	1.00
Focal Seizure	No-event	1.00
	Focal Seizure	240.73
Generalized Seizure	No-event	1.00
	Generalized Seizure	970.17
Simple Partial Seizure	No-event	1.00
	Simple Partial Seizure	1.00
Complex Partial Seizure	No-event	1.00
	Complex Partial Seizure	422.24
Absence Seizure	No-event	1.00
	Absence Seizure	64590.02
Tonic Seizure	No-event	1.00
	Tonic Seizure	70200.67

Tonic-Clonic Seizure	No-event	1.00
	Tonic-Clonic Seizure	14009.58
Myoclonic Seizure	No-event	1.00
	Myoclonic Seizure	1.00
Interictal Epileptiform Discharge	No-event	1.00
	IED	200.00
Multiclass TUEV	No-event	1.00
	Spike and/or Slow Wave	20000.00
	Generalized Periodic Epileptiform Discharge	1200.66
	Periodic Lateral Epileptiform Discharge	1200.66
	Eye Movement	80.38
	Artifact	100.00
Spike and/or Slow Wave	No-event	1.00
	Spike and/or Slow Wave	2000.00
Generalized Periodic Epileptiform Discharge	No-event	1.00
	Generalized Periodic Epileptiform Discharge	1200.66
Periodic Lateral Epileptiform Discharge	No-event	1.00
	Periodic Lateral Epileptiform Discharge	1200.66
Artifact	No-event	1.00
	Artifact	500.00
Eye Movement	No-event	1.00
	Eye Movement	80.38
Binary TUAR	No-event	1.00
	Artifact	10.00
Multiclass TUAR	No-event	1.00
	Eye Movement	200.38
	Muscle Artifact	1000.43
	Electrode Artifact	1000.00
	Chewing	1000.00
	Shivers	1600.00
	Electrode Pop	1600.00
Shivers	No-event	1.00
	Shivers	10000.00
Muscle Artifact	No-event	1.00
	Muscle Artifact	400.43

Electrode Pop	No-event	1.00
	Electrode Pop	16000.00
Chewing	No-event	1.00
	Chewing	500.00
Electrode Artifact	No-event	1.00
	Electrode Artifact	200.00
Abnormal	No-event	1.00
	Abnormal	1.00
Epilepsy	No-event	1.00
	Epilepsy	1.00

Note that the weights for sleep staging were taken from [7].

Appendix B

Extended Query Results

Table B.1: A complete list of our separate runs' experimental results including all queries, as well as precision and recall.

Dataset	**Query**	**Class**	**Precision**	**Recall**	**F1 Score**	**Balanced Accuracy**
Sleep-Cassette	Sleep Stages	Wake Phase	0.958	0.966	0.962	0.933
		N1 Phase	0.344	0.155	0.213	0.568
		N2 Phase	0.664	0.877	0.756	0.897
		N3/N4 Phase	0.802	0.717	0.757	0.855
		REM Phase	0.707	0.422	0.529	0.705
		(macro) F1 & Balanced Accuracy			**0.644**	**0.628**
Sleep-Telemetry	Sleep Stages	Wake Phase	0.633	0.724	0.675	0.843
		N1 Phase	0.331	0.499	0.398	0.705
		N2 Phase	0.888	0.670	0.764	0.788
		N3/N4 Phase	0.560	0.882	0.685	0.883
		REM Phase	0.693	0.733	0.712	0.834
		(macro) F1 & Balanced Accuracy			**0.647**	**0.702**
TUSZ (reduced)	Seizure	No-event	0.597	0.400	0.479	0.635
		Seizure	0.751	0.870	0.806	0.635
		(macro) F1 & Balanced Accuracy			**0.642**	**0.635**
	Multiclass Seizure	No-event	0.521	0.478	0.499	0.633
		Focal Seizure	0.469	0.634	0.539	0.626
		Generalized Seizure	0.328	0.301	0.314	0.563
		Complex Partial Seizure	0.031	0.010	0.015	0.491
		Absence Seizure	0.000	0.000	0.000	0.500
		Tonic Seizure	0.000	0.000	0.000	0.500
		Tonic-Clonic Seizure	0.663	0.130	0.218	0.565
		(macro) F1 & Balanced Accuracy			**0.226**	**0.222**

Table B.1: (continued)

TUSZ (reduced)	Focal Seizure	No-event	0.788	0.427	0.554	0.606
		Focal Seizure	0.421	0.784	0.548	0.606
		(macro) F1 & Balanced Accuracy			**0.551**	**0.606**
	Generalized Seizure	No-event	0.806	0.836	0.821	0.562
		Generalized Seizure	0.333	0.289	0.309	0.562
		(macro) F1 & Balanced Accuracy			**0.565**	**0.562**
	Simple Partial Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**
	Complex Partial Seizure	No-event	0.915	0.969	0.941	0.494
		Complex Partial Seizure	0.054	0.019	0.028	0.494
		(macro) F1 & Balanced Accuracy			**0.485**	**0.494**
	Absence Seizure	No-event	0.993	1.000	0.997	0.500
		Absence Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.498**	**0.500**
	Tonic Seizure	No-event	0.999	1.000	1.000	0.500
		Tonic Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
	Tonic-Clonic Seizure	No-event	0.986	0.999	0.993	0.552
		Tonic-Clonic Seizure	0.582	0.105	0.178	0.552
		(macro) F1 & Balanced Accuracy			**0.585**	**0.552**
	Myoclonic Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**
TUSZ (unreduced)	Seizure	No-event	0.965	0.987	0.976	0.714
		Seizure	0.684	0.440	0.536	0.714
		(macro) F1 & Balanced Accuracy			**0.756**	**0.714**

Table B.1: (continued)

TUSZ (unreduced)	Multiclass Seizure	No-event	0.962	0.988	0.975	0.692
		Focal Seizure	0.496	0.418	0.453	0.702
		Generalized Seizure	0.393	0.149	0.216	0.572
		Complex Partial Seizure	0.042	0.010	0.016	0.504
		Absence Seizure	0.000	0.000	0.000	0.500
		Tonic Seizure	0.000	0.000	0.000	0.500
		Tonic-Clonic Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.237**	**0.223**
	Focal Seizure	No-event	0.983	0.983	0.983	0.716
		Focal Seizure	0.450	0.450	0.450	0.716
		(macro) F1 & Balanced Accuracy			**0.716**	**0.716**
	Generalized Seizure	No-event	0.983	0.995	0.989	0.572
		Generalized Seizure	0.373	0.148	0.212	0.572
		(macro) F1 & Balanced Accuracy			**0.601**	**0.572**
	Simple Partial Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**
	Complex Partial Seizure	No-event	0.993	0.998	0.995	0.509
		Complex Partial Seizure	0.080	0.019	0.031	0.509
		(macro) F1 & Balanced Accuracy			**0.513**	**0.509**
	Absence Seizure	No-event	0.999	1.000	1.000	0.500
		Absence Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
	Tonic Seizure	No-event	1.000	1.000	1.000	0.500
		Tonic Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
	Tonic-Clonic Seizure	No-event	0.999	1.000	0.999	0.500
		Tonic-Clonic Seizure	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
	Myoclonic Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**

Table B.1: (continued)

CHB-MIT (reduced)	Seizure	No-event	0.619	0.679	0.648	0.640
		Seizure	0.662	0.600	0.629	0.640
		(macro) F1 & Balanced Accuracy			**0.638**	**0.640**
TUEV (reduced)	Interictal Epileptiform Discharge	No-event	0.843	0.835	0.839	0.581
		IED	0.314	0.326	0.320	0.581
		(macro) F1 & Balanced Accuracy			**0.580**	**0.581**
	Multiclass TUEV	No-event	0.753	0.725	0.739	0.546
		Spike and/or Slow Wave	0.009	0.053	0.016	0.508
		Generalized Periodic Epileptiform Discharge	0.257	0.174	0.208	0.562
		Periodic Lateral Epileptiform Discharge	0.239	0.153	0.187	0.552
		Eye Movement	0.079	0.166	0.107	0.568
		Artifact	0.181	0.286	0.222	0.594
		(macro) F1 & Balanced Accuracy			**0.246**	**0.259**
	Eye Movement	No-event	0.986	0.976	0.981	0.540
		Eye Movement	0.064	0.104	0.079	0.540
		(macro) F1 & Balanced Accuracy			**0.530**	**0.540**
	Spike and/or Slow Wave	No-event	0.993	0.993	0.993	0.496
		Spike and/or Slow Wave	0.000	0.000	0.000	0.496
		(macro) F1 & Balanced Accuracy			**0.496**	**0.496**
	Generalized Periodic Epileptiform Discharge	No-event	0.925	0.922	0.923	0.584
		Generalized Periodic Epileptiform Discharge	0.239	0.245	0.242	0.584
		(macro) F1 & Balanced Accuracy			**0.583**	**0.584**
	Periodic Lateral Epileptiform Discharge	No-event	0.916	0.956	0.936	0.537
		Periodic Lateral Epileptiform Discharge	0.211	0.119	0.152	0.537
		(macro) F1 & Balanced Accuracy			**0.544**	**0.537**

Table B.1: (continued)

TUEV (reduced)	Artifact	No-event	0.939	0.848	0.891	0.570
		Artifact	0.130	0.292	0.180	0.570
		(macro) F1 & Balanced Accuracy			**0.536**	**0.570**
TUAR (reduced)	Binary TUAR	No-event	0.666	0.245	0.358	0.584
		Artifact	0.661	0.923	0.770	0.584
		(macro) F1 & Balanced Accuracy			**0.564**	**0.584**
	Multiclass TUAR	No-event	0.681	0.184	0.290	0.565
		Eye Movement	0.290	0.400	0.336	0.601
		Muscle Artifact	0.419	0.600	0.493	0.645
		Electrode Artifact	0.326	0.505	0.396	0.650
		Chewing	0.244	0.514	0.331	0.750
		Shivers	0.000	0.000	0.000	0.499
		Electrode Pop	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.264**	**0.315**
	Shivers	No-event	0.997	0.999	0.998	0.499
		Shivers	0.000	0.000	0.000	0.499
		(macro) F1 & Balanced Accuracy			**0.499**	**0.499**
	Muscle Artifact	No-event	0.840	0.515	0.638	0.671
		Muscle Artifact	0.489	0.826	0.615	0.671
		(macro) F1 & Balanced Accuracy			**0.627**	**0.671**
	Electrode Pop	No-event	1.000	1.000	1.000	0.500
		Electrode Pop	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
	Chewing	No-event	0.999	0.937	0.967	0.924
		Chewing	0.118	0.912	0.209	0.924
		(macro) F1 & Balanced Accuracy			**0.588**	**0.924**
	Eye Movement	No-event	0.947	0.494	0.649	0.698
		Eye Movement	0.333	0.901	0.486	0.698
		(macro) F1 & Balanced Accuracy			**0.568**	**0.698**
	Electrode Artifact	No-event	0.925	0.606	0.732	0.711
		Electrode Artifact	0.358	0.817	0.498	0.711
		(macro) F1 & Balanced Accuracy			**0.615**	**0.711**

Table B.1: (continued)

TUAB	Abnormal	No-event	0.857	0.880	0.868	0.853
		Abnormal	0.852	0.825	0.839	0.853
		(macro) F1 & Balanced Accuracy			**0.854**	**0.853**
TUEP	Epilepsy	No-event	0.675	0.458	0.545	0.696
		Epilepsy	0.854	0.935	0.893	0.696
		(macro) F1 & Balanced Accuracy			**0.719**	**0.696**

Table B.2: A complete list of our combined run's experimental results including all queries, as well as precision and recall.

Dataset	**Query**	**Class**	**Precision**	**Recall**	**F1 Score**	**Balanced Accuracy**
Sleep-Cassette	Sleep Stages	Wake Phase	0.977	0.961	0.969	0.956
		N1 Phase	0.355	0.258	0.299	0.614
		N2 Phase	0.716	0.809	0.760	0.875
		N3/N4 Phase	0.684	0.840	0.754	0.913
		REM Phase	0.609	0.619	0.614	0.796
		(macro) F1 & Balanced Accuracy			**0.679**	**0.697**
Sleep-Telemetry	Sleep Stages	Wake Phase	0.619	0.770	0.687	0.863
		N1 Phase	0.414	0.262	0.321	0.614
		N2 Phase	0.864	0.725	0.788	0.800
		N3/N4 Phase	0.589	0.873	0.703	0.887
		REM Phase	0.680	0.840	0.752	0.879
		(macro) F1 & Balanced Accuracy			**0.650**	**0.694**
TUSZ (reduced)	Seizure	No-event	0.328	0.020	0.038	0.500
		Seizure	0.675	0.980	0.800	0.500
		(macro) F1 & Balanced Accuracy			**0.419**	**0.500**
	Multiclass Seizure	No-event	0.412	0.497	0.450	0.578
		Focal Seizure	0.000	0.000	0.000	0.500
		Generalized Seizure	0.176	0.220	0.195	0.464
		Complex Partial Seizure	0.122	0.203	0.152	0.534
		Absence Seizure	0.012	0.019	0.015	0.504
		Tonic Seizure	0.000	0.000	0.000	0.489
		Tonic-Clonic Seizure	0.010	0.103	0.018	0.473
		(macro) F1 & Balanced Accuracy			**0.119**	**0.149**

Table B.2: (continued)

TUSZ (reduced)	Focal Seizure	No-event	0.798	0.558	0.657	0.646
		Focal Seizure	0.469	0.734	0.572	0.646
		(macro) F1 & Balanced Accuracy			**0.614**	**0.646**
	Generalized Seizure	No-event	0.786	0.668	0.722	0.513
		Generalized Seizure	0.234	0.358	0.283	0.513
		(macro) F1 & Balanced Accuracy			**0.503**	**0.513**
	Simple Partial Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**
	Complex Partial Seizure	No-event	0.921	0.726	0.812	0.523
		Complex Partial Seizure	0.097	0.320	0.149	0.523
		(macro) F1 & Balanced Accuracy			**0.480**	**0.523**
	Absence Seizure	No-event	0.993	0.999	0.996	0.499
		Absence Seizure	0.000	0.000	0.000	0.499
		(macro) F1 & Balanced Accuracy			**0.498**	**0.499**
	Tonic Seizure	No-event	0.999	0.995	0.997	0.498
		Tonic Seizure	0.000	0.000	0.000	0.498
		(macro) F1 & Balanced Accuracy			**0.499**	**0.498**
	Tonic-Clonic Seizure	No-event	0.984	0.884	0.931	0.468
		Tonic-Clonic Seizure	0.007	0.052	0.012	0.468
		(macro) F1 & Balanced Accuracy			**0.472**	**0.468**
	Myoclonic Seizure	No-event	1.000	1.000	1.000	1.000
		(macro) F1 & Balanced Accuracy			**1.000**	**1.000**
CHB-MIT (reduced)	Seizure	No-event	0.856	0.324	0.470	0.639
		Seizure	0.626	0.954	0.756	0.639
		(macro) F1 & Balanced Accuracy			**0.613**	**0.639**
TUEV (reduced)	Interictal Epileptiform Discharge	No-event	0.842	0.629	0.720	0.559
		IED	0.233	0.489	0.316	0.559
		(macro) F1 & Balanced Accuracy			**0.518**	**0.559**

Table B.2: (continued)

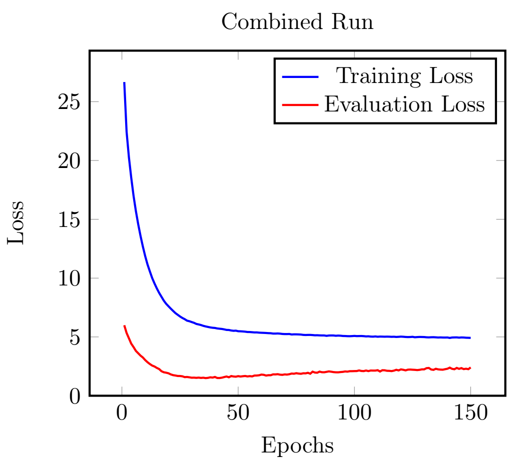
TUEV (reduced)	Multiclass TUEV	No-event	0.759	0.539	0.630	0.542
		Spike and/or Slow Wave	0.008	0.230	0.015	0.517
		Generalized Periodic Epileptiform Discharge	0.000	0.000	0.000	0.500
		Periodic Lateral Epileptiform Discharge	0.102	0.117	0.109	0.508
		Eye Movement	0.041	0.154	0.064	0.549
		Artifact	0.043	0.078	0.056	0.474
		(macro) F1 & Balanced Accuracy			**0.146**	**0.186**
	Eye Movement	No-event	0.985	0.980	0.983	0.510
		Eye Movement	0.031	0.040	0.035	0.510
		(macro) F1 & Balanced Accuracy			**0.509**	**0.510**
	Spike and/or Slow Wave	No-event	0.993	1.000	0.996	0.500
		Spike and/or Slow Wave	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.498**	**0.500**
	Generalized Periodic Epileptiform Discharge	No-event	0.909	0.991	0.948	0.495
		Generalized Periodic Epileptiform Discharge	0.000	0.000	0.000	0.495
		(macro) F1 & Balanced Accuracy			**0.474**	**0.495**
	Periodic Lateral Epileptiform Discharge	No-event	0.910	0.998	0.952	0.499
		Periodic Lateral Epileptiform Discharge	0.000	0.000	0.000	0.499
		(macro) F1 & Balanced Accuracy			**0.476**	**0.499**
	Artifact	No-event	0.929	0.985	0.956	0.512
		Artifact	0.167	0.038	0.062	0.512
		(macro) F1 & Balanced Accuracy			**0.509**	**0.512**
TUAR (reduced)	Binary TUAR	No-event	0.393	0.560	0.462	0.508
		Artifact	0.623	0.456	0.526	0.508
		(macro) F1 & Balanced Accuracy			**0.494**	**0.508**

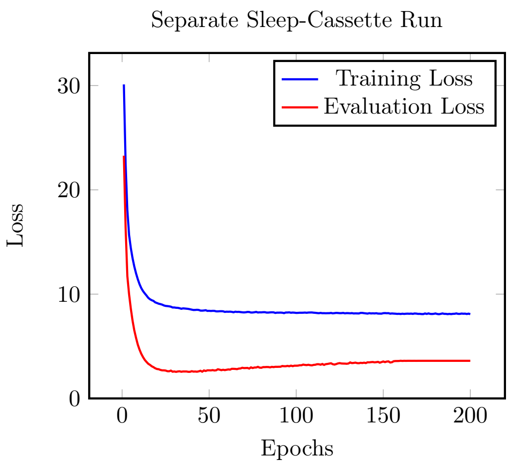
Table B.2: (continued)

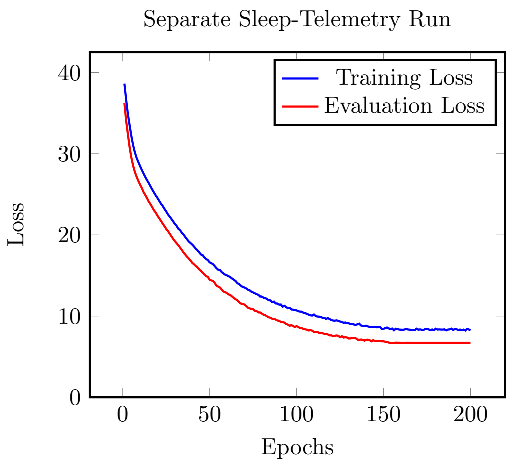
TUAR (reduced)	Multiclass TUAR	No-event	0.389	0.734	0.508	0.504
		Eye Movement	0.000	0.000	0.000	0.500
		Muscle Artifact	0.275	0.252	0.263	0.502
		Electrode Artifact	0.168	0.022	0.039	0.500
		Chewing	0.067	0.011	0.019	0.505
		Shivers	0.000	0.000	0.000	0.500
		Electrode Pop	0.000	0.000	0.000	0.500
		(macro) F1 & Balanced Accuracy			**0.118**	**0.146**
	Shivers	No-event	0.998	0.887	0.939	0.533
		Shivers	0.004	0.179	0.008	0.533
		(macro) F1 & Balanced Accuracy			**0.474**	**0.533**
	Muscle Artifact	No-event	0.643	0.720	0.679	0.505
		Muscle Artifact	0.368	0.290	0.324	0.505
		(macro) F1 & Balanced Accuracy			**0.502**	**0.505**
	Electrode Pop	No-event	1.000	0.967	0.983	0.483
		Electrode Pop	0.000	0.000	0.000	0.483
		(macro) F1 & Balanced Accuracy			**0.492**	**0.483**
	Chewing	No-event	0.992	0.634	0.774	0.557
		Chewing	0.012	0.481	0.023	0.557
		(macro) F1 & Balanced Accuracy			**0.399**	**0.557**
	Eye Movement	No-event	0.784	0.759	0.771	0.506
		Eye Movement	0.227	0.253	0.240	0.506
		(macro) F1 & Balanced Accuracy			**0.505**	**0.506**
	Electrode Artifact	No-event	0.793	0.703	0.745	0.510
		Electrode Artifact	0.224	0.318	0.263	0.510
		(macro) F1 & Balanced Accuracy			**0.504**	**0.510**
TUAB	Abnormal	No-event	0.500	0.500	0.500	0.500
		Abnormal	0.500	0.500	0.500	0.500
		(macro) F1 & Balanced Accuracy			**0.500**	**0.500**
TUEP	Epilepsy	No-event	0.724	0.588	0.649	0.748
		Epilepsy	0.842	0.908	0.874	0.748
		(macro) F1 & Balanced Accuracy			**0.761**	**0.748**

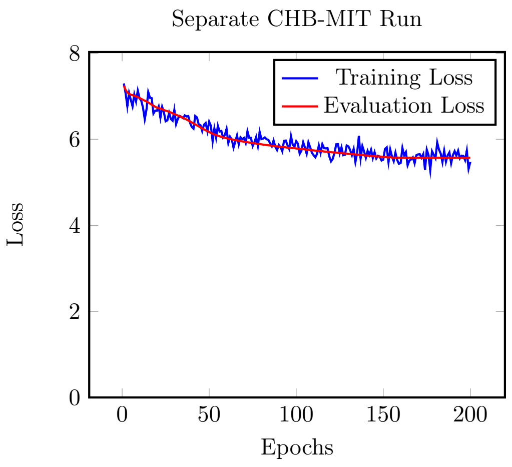
Appendix C

Dataset Loss Expansions

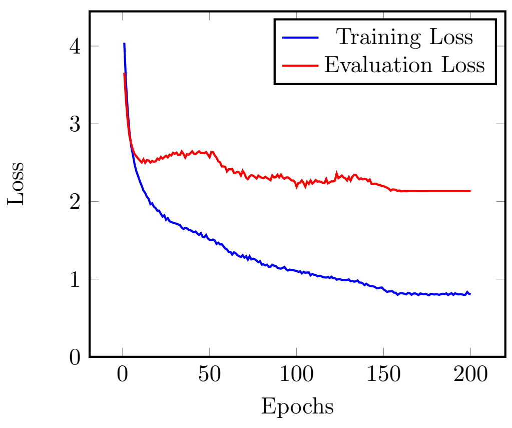




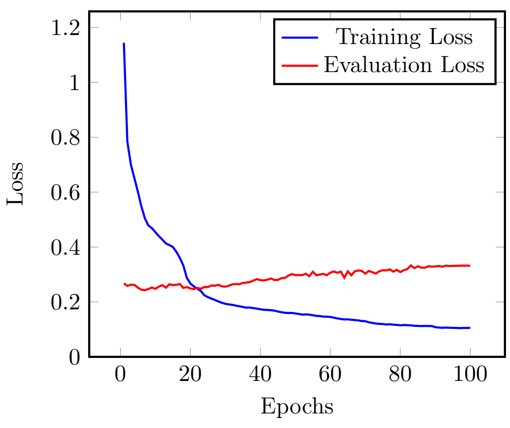




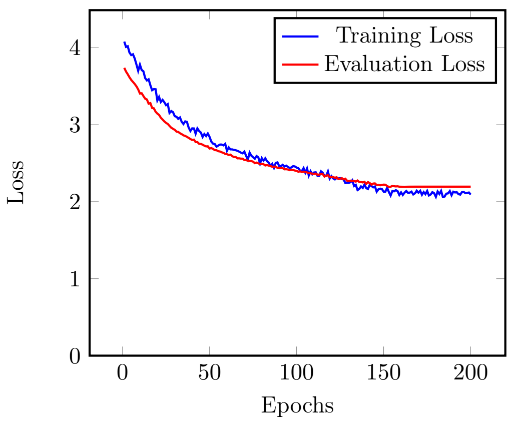
Separate Reduced TUSZ Run



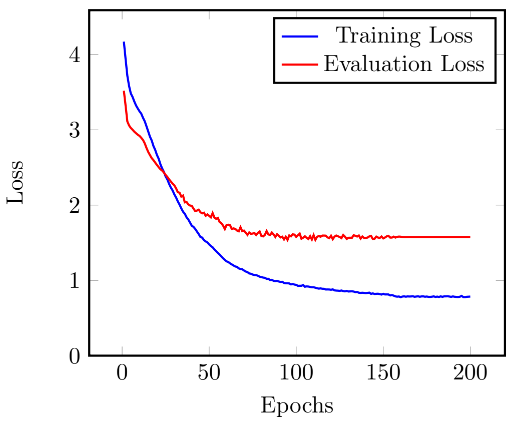
Separate Unreduced TUSZ Run



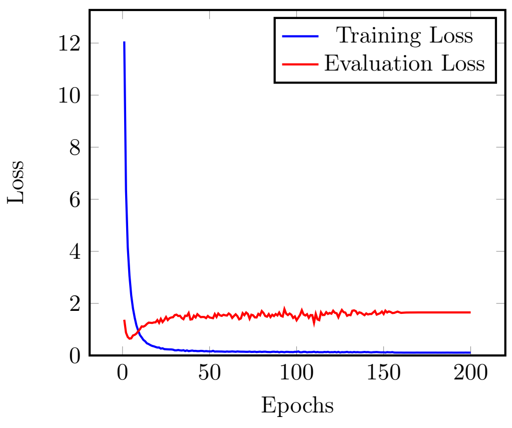
Separate TUEV Run



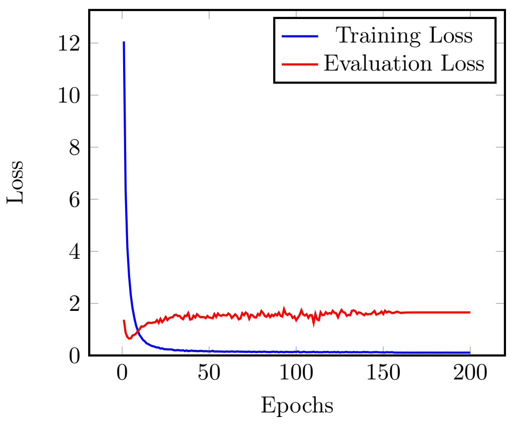
Separate TUAR Run



Separate TUAB Run

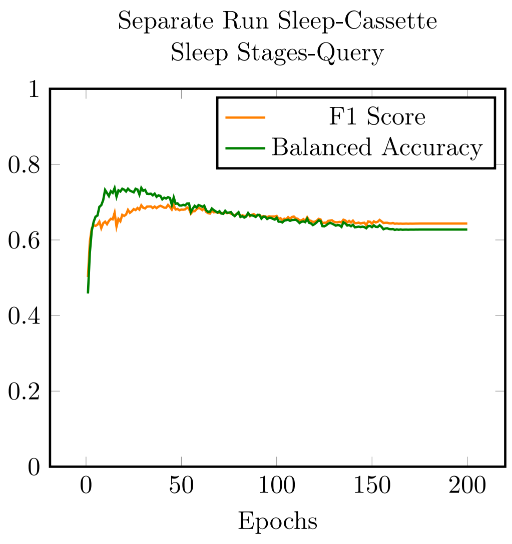


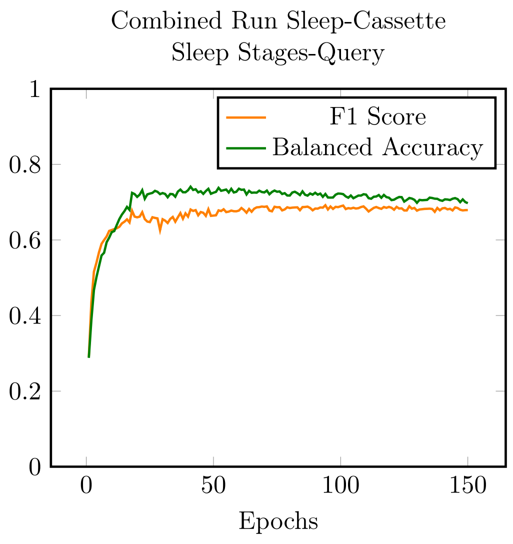
Separate TUEP Run

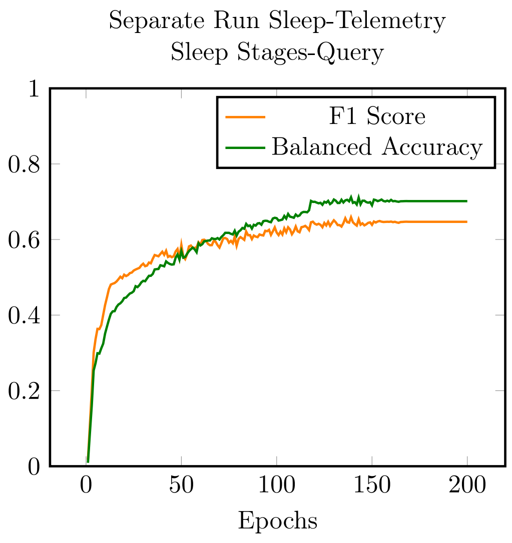


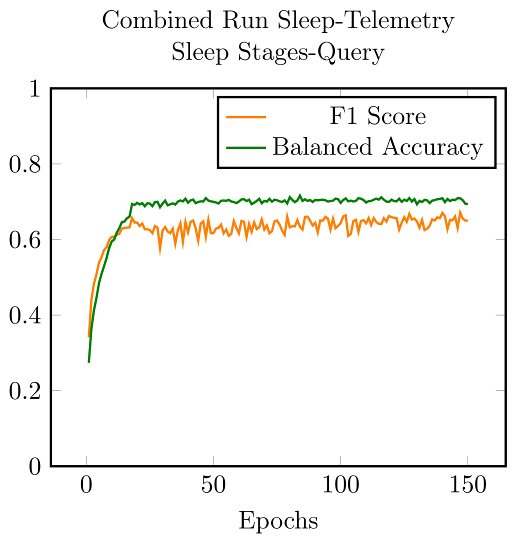
Appendix D

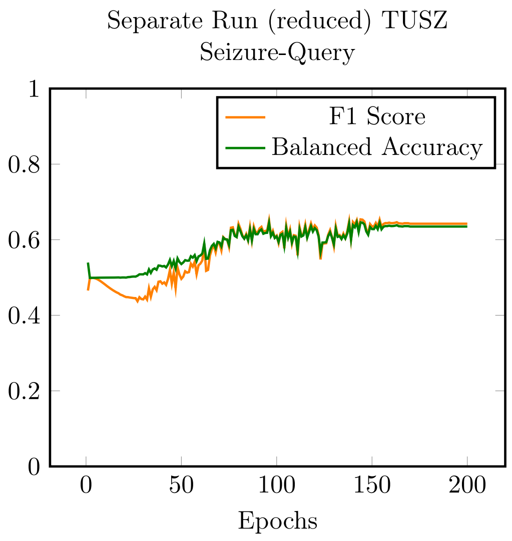
F1 Score and Balanced Accuracy Expansion Plots

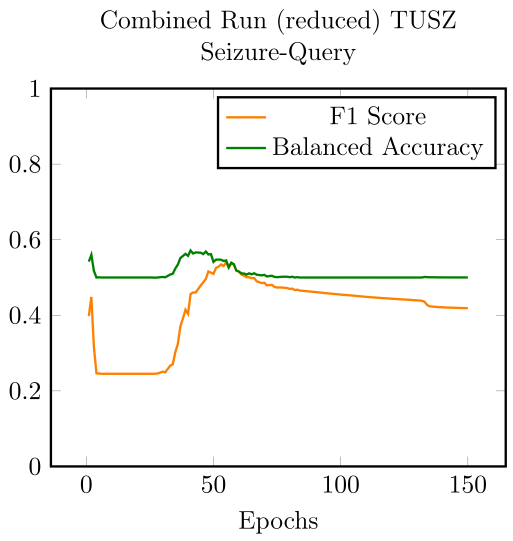


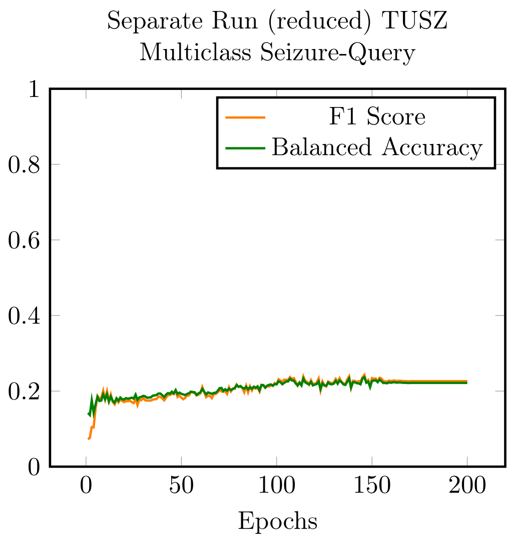


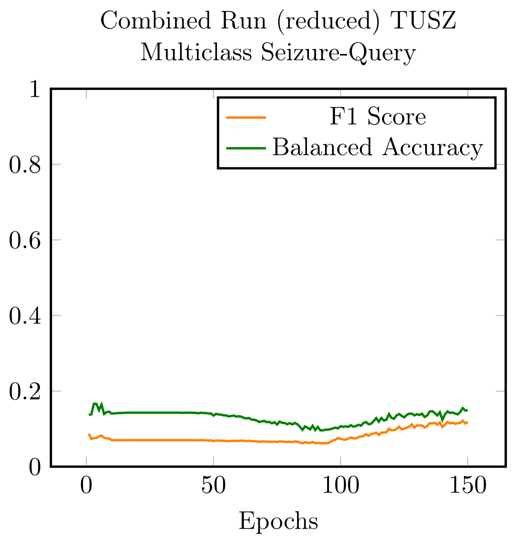


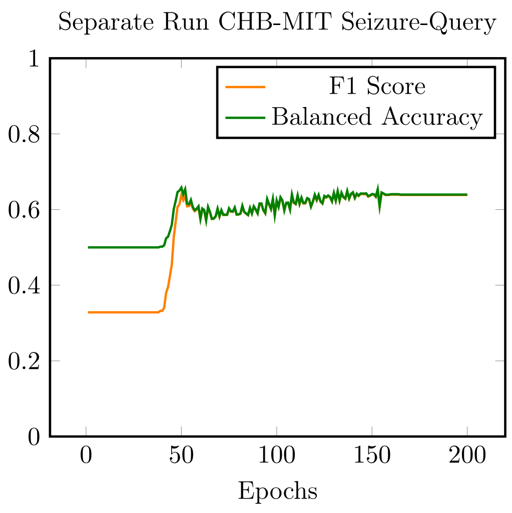


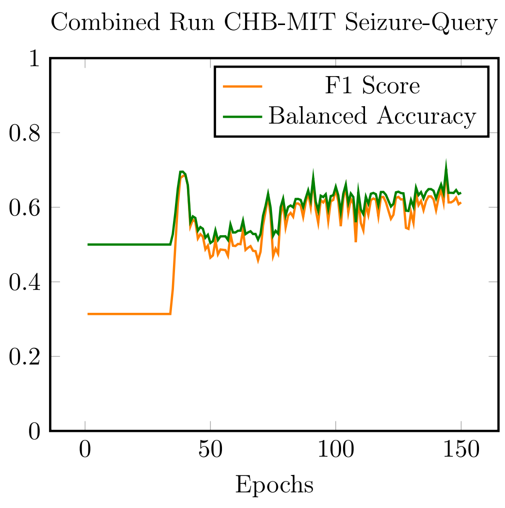

Separate Run (reduced) TUSZ
Seizure-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs

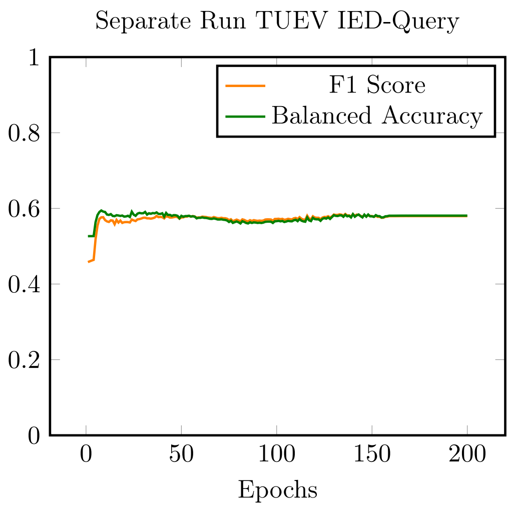

Combined Run (reduced) TUSZ
Seizure-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
Epochs

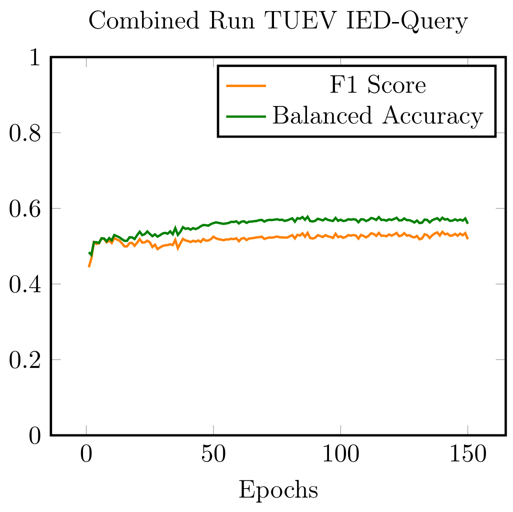

Separate Run (reduced) TUSZ
Multiclass Seizure-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs

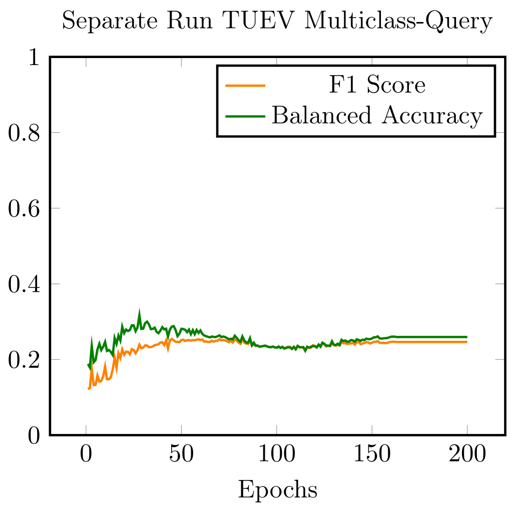

Combined Run (reduced) TUSZ
Multiclass Seizure-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
Epochs

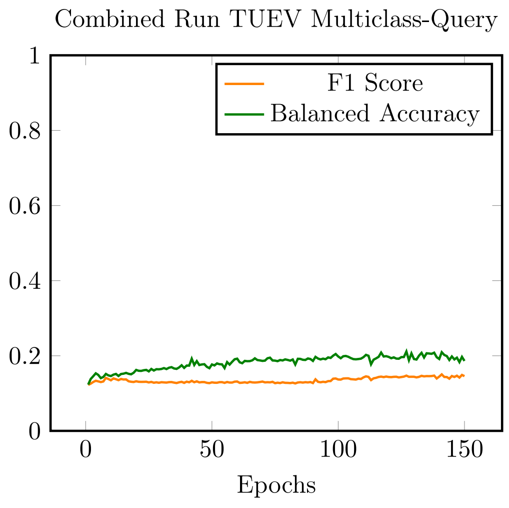

Separate Run CHB-MIT Seizure-Query
F1 Score
Balanced Accuracy
Epochs

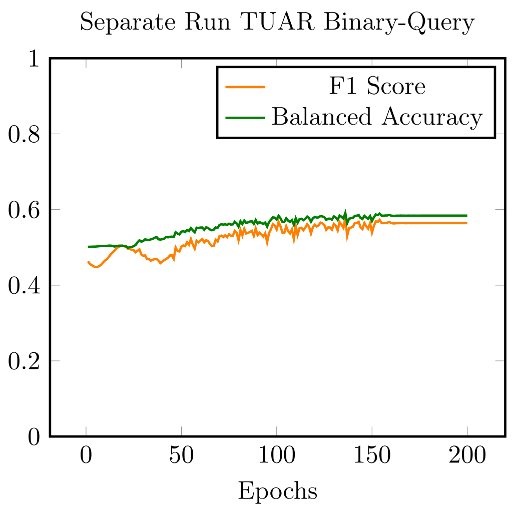

Combined Run CHB-MIT Seizure-Query
F1 Score
Balanced Accuracy
Epochs

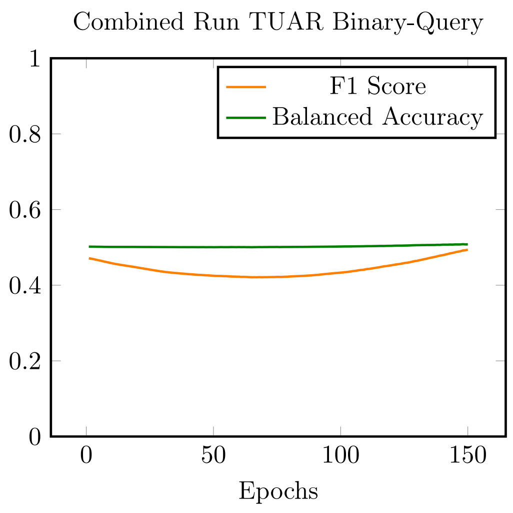

Separate Run TUEV IED-Query
F1 Score
Balanced Accuracy
Epochs

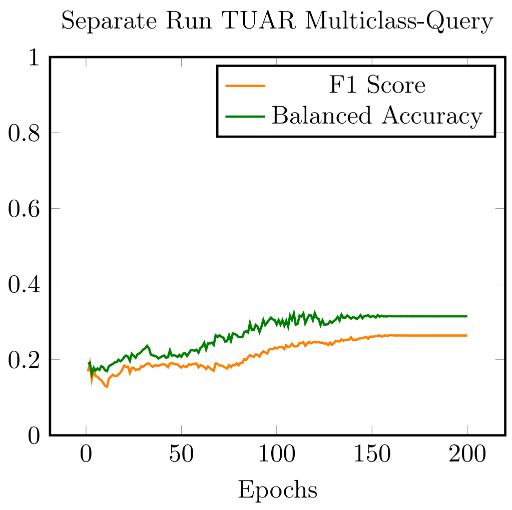

Combined Run TUEV IED-Query
F1 Score
Balanced Accuracy
Epochs

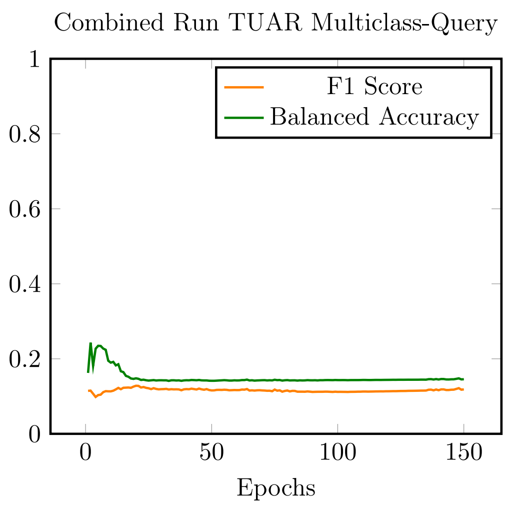

Separate Run TUEV Multiclass-Query
F1 Score
Balanced Accuracy
Epochs

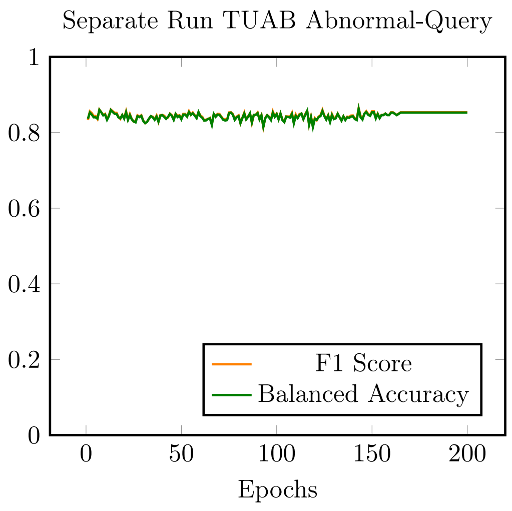

Combined Run TUEV Multiclass-Query
F1 Score
Balanced Accuracy
Epochs

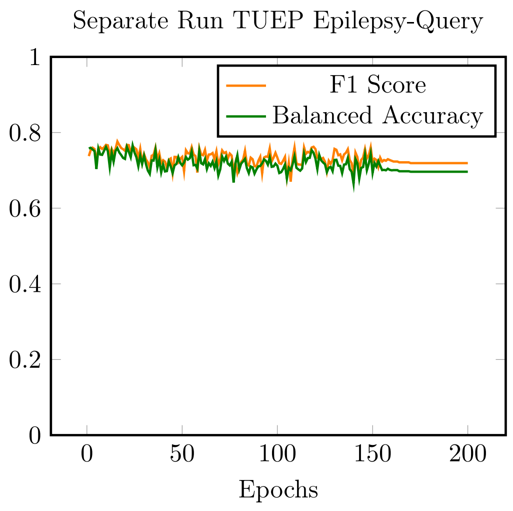

Separate Run TUAR Binary-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs


Combined Run TUAR Binary-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
Epochs


Separate Run TUAR Multiclass-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs


Combined Run TUAR Multiclass-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
Epochs


Separate Run TUAB Abnormal-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs


Separate Run TUEP Epilepsy-Query
F1 Score
Balanced Accuracy
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
Epochs