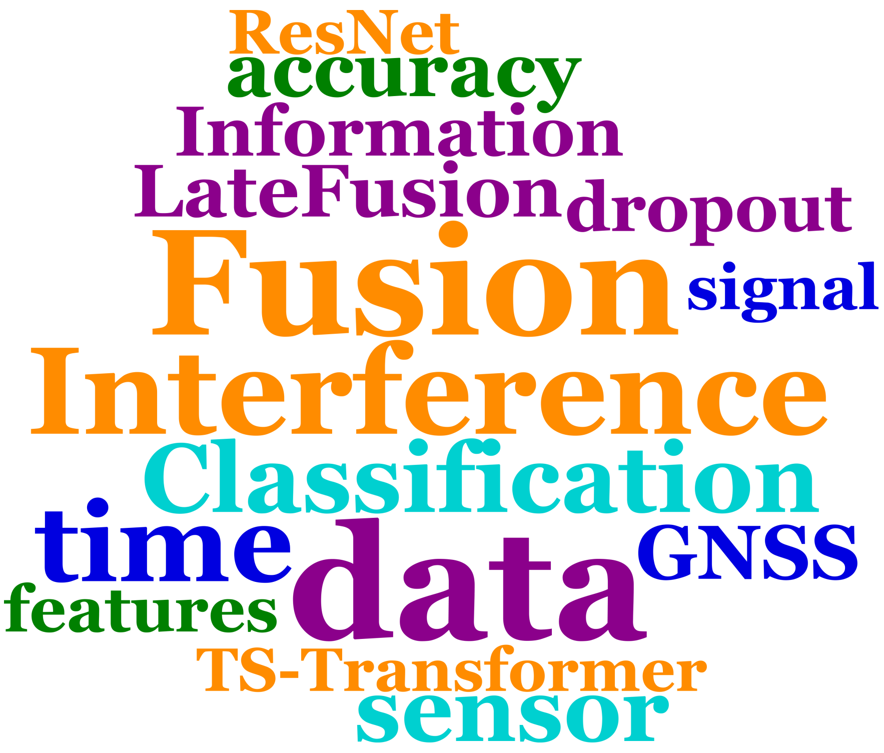

ResNet
accuracy
Information
LateFusion
dropout
Fusion
signal
Interference
Classification
time
data
GNSS
features
TS-Transformer
sensor

# Multimodal Learning for Reliable Interference Classification in GNSS Signals

Tobias Brieger [*†], Nisha Lakshmana Raichur[†], Dorsaf Jdidi [*†], Felix Ott[†], Tobias Feigl[†],
J. Rossouw van der Merwe[†], Alexander Rügamer[†], and Wolfgang Felber[†]
[*] *Friedrich-Alexander-University (FAU), Erlangen-Nuremberg, Germany*
[†] *Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany*

## BIOGRAPHY

**Tobias Brieger** received his B.Sc. degree in Computer Science from Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany in 2019. He is completing his master's degree in Computer Science at FAU. Since January 2022, he works as a graduate assistant at the Fraunhofer Institute for Integrated Circuits (IIS), Nuremberg, Germany, in the "Hybrid Positioning & Information Fusion" group. His research focus is on multimodal learning to improve digital signal processing and information fusion for GNSS systems.

**Nisha Lakshmana Raichur** received her M.Sc. degree in Computer Science from Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany in 2021. Since October 2021, she works as a research assistant in the "Hybrid Positioning & Information Fusion" group of the Fraunhofer Institute for Integrated Circuits (IIS), Nuremberg, Germany. Her research focus is on multimodal learning to improve digital signal processing and information fusion.

**Dorsaf Jdidi** received her B.Sc. degree in Computer Science from Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany in 2019. She is completing her master's degree in Computer Science at FAU. Since January 2022, she works as a graduate assistant at the Fraunhofer Institute for Integrated Circuits (IIS), Nuremberg, Germany, in the "Hybrid Positioning & Information Fusion" group. Her research focus is on unsupervised learning to improve digital signal processing and information fusion for GNSS systems.

**Felix Ott** received his M.Sc. degree in Computational Engineering from the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany in 2019. In 2019 he joined the "Hybrid Positioning & Information Fusion" group at the Fraunhofer Institute for Integrated Circuits (IIS) Nuremberg. In 2020 he started his Ph.D. at the Ludwig-Maximilians-University (LMU) in Munich in the group "Probabilistic Machine and Deep Learning". His research covers multimodal information fusion for self-location.

**Tobias Feigl** received his Ph.D. degree in Computer Science from the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) in 2021 and his Masters degree from the University of Applied Sciences Erlangen-Nuremberg, Germany, in 2017. In 2017 he joined the Fraunhofer Institute for Integrated Circuits (IIS) Nuremberg, Germany, where he worked as a student since 2009, with a strong focus on AI and information fusion. In parallel, since 2017 he is a lecturer at the Computer Science department at FAU, where he gives courses on machine and deep learning and supervises related qualification work.

**J. Rossouw van der Merwe** received his M.Eng. in Electronic Engineering from the University of Pretoria South Africa in 2016. He joined the Fraunhofer Institute for Integrated Circuits IIS the same year, where his research focus is on signal processing methods for robust and resilient GNSS receivers, including interference mitigation, spoofing detection, and array processing. Currently, he is pursuing his Doctorate at the Friedrich Alexander Universität, Erlangen, Germany.

**Alexander Rügamer** received his Dipl.-Ing. (FH) degree in Electrical Engineering from the University of Applied Sciences Würzburg-Schweinfurt, Germany, in 2007. Since then, he has been working at the Fraunhofer Institute for Integrated Circuits IIS in the Field of GNSS receiver development. He was promoted to Senior Engineer in February 2012. Since April 2013, he is head of a research group dealing with secure GNSS receivers and receivers for special applications. His main research interests focus on GNSS multi-band reception, integrated circuits, and immunity to interference.

**Wolfgang Felber** received his Dipl.-Ing. degree in Electrical Engineering in 2002 and his doctoral degree Dr.-Ing. in 2006 from Helmut-Schmidt-University of Federal Armed Forces Hamburg, Germany. Since 2014 he is head of the Satellite Based Positioning Systems department of Fraunhofer IIS, division Localisation and Networking in Nuremberg. The main topics in his department are hardware development of satellite navigation receivers for multiple or hybrid precise systems and secure applications. Additionally, since 2016 he is head of the business field localization at Nuremberg, which combines different localization technologies for industrial IoT applications.

## ABSTRACT

Interference signals degrade and disrupt Global Navigation Satellite System (GNSS) receivers, impacting their localization accuracy. Therefore, they need to be detected, classified, and located to ensure GNSS operation. State-of-the-art techniques employ supervised deep learning to detect and classify potential interference signals. Here, literature proposes ResNet18 and TS-Transformer as they provide the most accurate classification rates on quasi-realistic GNSS signals. However, employing these methods individually, they only focus on either spatial or temporal information and discard information during optimization, thereby degrading classification accuracy.

This paper proposes a deep learning framework that considers both the spatial and temporal relationships between samples when fusing ResNet18 and TS-Transformers with a joint loss function to compensate for the weaknesses of both methods considered individually. Our real-world experiments show that our novel fusion pipeline with an adapted late fusion technique and uncertainty measure significantly outperforms the state-of-the-art classifiers by 6.7% on average, even in complicated realistic scenarios with multipath propagation and environmental dynamics. This works even well (F-$\beta$=2 score about 80.1%), when we fuse both modalities only from a single bandwidth-limited low-cost sensor, instead of a fine-grained high-resolution sensor and coarse-grained low-resolution low-cost sensor. By using late fusion the classification accuracy of the classes *FreqHopper*, *Modulated*, and *Noise* increases while lowering the uncertainty of *Multitone*, *Noise*, and *Pulsed*. The improved classification capabilities allow for more reliable results even in challenging scenarios.

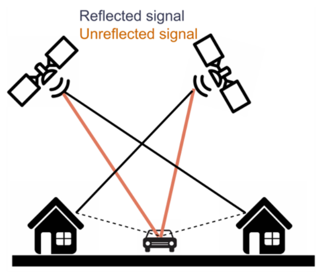
## KEYWORDS

Global navigation satellite system (GNSS), jamming, detection, classification, multimodal fusion, intermediate fusion, late fusion, Multimodal Transfer Module (MMTM), RestNet, TS-Transformer, machine learning, deep learning, time-series

## I. INTRODUCTION

Interference signals affect the Global Navigation Satellite System (GNSS) and so, degrade its localization accuracy (Dovis, 2015; Rügamer and Kowalewski, 2015). The problem is only getting worse as sources of interference such as Privacy Protection Devices (PPDs) (see Fig. 1a) become cheaper and easier to obtain (van der Merwe et al., 2022b). Such PPDs can be actively used on the highway to force collisions with self-driving cars like the Tesla Model 3 through jamming attacks (Zangvil Y, 2019). These jammers are also used in space, e.g., Finnair reported GPS jamming near Kaliningrad, where the collateral damage can be even worse (Reuters, 2022). Therefore, potential interference signals must be mitigated or a potential transmitter (i.e., jammer) eliminated. However, to successfully remove the interference transmitters physically, they must first be detected and then localized. In addition, the successful classification of the waveform of an interference signal helps to infer the signal's purpose, thus simplifying its localization. For instance, if the waveform matches a typical PPD, this suggests that the jammer is most likely inside a car, but if it matches a waveform used for aeronautic ranging, then it is likely mounted on a tower near an airport. In addition, by understanding which interference types exist and how often they occur, suitable interference mitigation methods can be developed and optimized. For instance, if mostly chirp-interference are observed, then interference mitigation algorithms that have superior chirp-mitigation are more efficient for receiver design (van der Merwe et al., 2022a). Lastly, a significant limitation of the current state-of-art interference detection and classification is that it either does not consider multipath or performs poorly in it (Xing and Gao, 2020), limiting real-world adaption. Therefore, real-world degradation should be overcome in detection and classification approaches. Signals that are subject to multipath interference look similar to interfered signals with the exception that the amplitude of a non-line-of-sight signal reflection is typically smaller than an interfered signal. For classic threshold-based methods, it is challenging to detect this small difference. Our MultiModal Learning (MML) approach learns to distinguish between different interference waveforms, while multipath effects are present.

**(a)** Mobile interference devices.



**(b)** Multipath between sensor, jammer, and satellites.

**Figure 1:** Different sources that affect the processing chain of GNSS and so, degrade its localization accuracy.

To address these challenges, classic approaches (Gross and Humphreys, 2017) apply Maximum-Likelihood to distinguish if a signal is clean or affected by multipath, spoofing, or jamming. However, they can only detect jamming but cannot classify the jammer type and suffer from real-world multipath effects. Recently, snapshot-based data-driven methods, such as Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) (Morales Ferre et al., 2019), outperformed classic model-based techniques, such as pattern recognition and mathematical formulations, as they enable high classification accuracy even under challenging scenarios. One of the reasons for this is that they can learn a mapping approximation directly from data that implicitly describe non-deterministic or nonlinear functions without any additional modeling effort. However, these classical learning-based methods either do not use time and context at all, e.g., Random Forest (RF) (Breiman, 2001), SVM (Noble, 2006), and CNN (Fukushima and Shouno, 2015), only consider time-dependent local phenomena (i.e., spatial features) in snapshots, e.g., ResNet (He et al., 2015) and Temporal CNN (TCN) (Wu et al., 2021), or only consider time-dependent global phenomena (i.e., time-sensitive features) in sequences but ignore local phenomena, e.g., Recurrent Neural Networks (RNN). Thus, when used individually, these methods always lack some potential information that, when used in combination, might increase classification performance.

To improve interference signal mitigation and integrate both spatial and time-sensitive features, we propose a novel MML system that improves classification accuracy beyond the state-of-the-art, accounts for the uncertainty of its estimates, and reduces computational and energy costs. To this end, our MML approach complements the prominent methods by Voigt (2021), who claims to provide the most robust and accurate classification to enable multimodal embedding of inputs. The key idea is that, on the one hand, low-frequency spectrograms embed spatial information and temporal relationships to enable a multipath-resistant architecture for classification; and on the other hand, coarse-grained high-frequency time series data provides information that allows lowering the sampling rates of the fine-grained informative spectrogram. This helps to reduce computational effort and improve efficiency. Our pipeline implicitly fuses different types of sensor information, namely spectrum and time domain, as it adapts and optimizes appropriate neural network types and combines their loss functions. Our experiments show that our MML framework with late fusion mechanics learns to implicitly weight between spatial (images of spectrum data) and time-sensitive (matrix of frequency bins) features, as it provides reliable interference classification. We acquire realistic, deterministic data in a large-scale real-world measurement campaign that covers 33 different waveforms across seven sources of interference signals. Our measurements cover multipath effects (caused by reflector and diffusor walls and a car), varying distances between sensors and jammer, and motion dynamics. We evaluate state-of-the-art methods and our MML framework on both synthetic and realistic datasets and mixtures thereof. We also show that using both, the time and spatial features, helps (only time: F2>59.5% and only spatial: F2>88.6%) to fuse different features from different sensors (F2>95.3%) but also to fuse different features from a single sensor (F2>80.1%).

To the best of our knowledge, we are the first to investigate data-driven multimodal fusion methods on real-world data to create energy-efficient multipath-resistant classification algorithms that may adapt to various types of input and artifacts thereof.

The rest of the paper is structured as follows. Sec. II discusses related work. Sec. III introduces the concept of our solution. Sec. IV describes our experimental setup. Sec. V reports and interprets results before Sec. VI concludes.

## II. RELATED WORK

This Section discusses relevant related work on jammer detection and classification as an important part of effective GNSS interference management. First, Section II.1 deals with classical (model-driven) techniques. Next, Section II.2 discusses modern (data-driven) techniques.

There are several ways to deal with interference signals (Dovis, 2015; Amin et al., 2017), such as: detection (Rügamer et al., 2013), mitigation (Broumandan et al., 2016; van der Merwe et al., 2018; Borio, 2018), localization (Bartl et al., 2017; Kim et al., 2012), or classification (van der Merwe et al., 2017; Mehr and Dovis, 2022; Elango et al., 2022). There are several main categories of interference signals (Dovis, 2015; van der Merwe et al., 2017), some examples are: (1) amplitude modulated, (2) chirp, (3) frequency modulated, (4) pulse or range finder-like devices, (5) narrowband, and (6) broadband jammers. And of course, each category has a variety of subclasses, signal properties, e.g., signal bandwidth or signal repetition rates (Rügamer et al., 2017), affected by environmental effects, e.g., multipath or signal attenuation (Rappaport, 1996), dynamic properties (i.e., the interference signal type and parameters change over time) (van der Merwe et al., 2022a), or even have multiple simultaneous interference signals (van der Merwe et al., 2020). Hence, all different categories are typically difficult to detect or to be classified, as the signal properties in the time and frequency domains may be similar in the presence and absence of jammers (Hashemi et al., 2019).

### 1. Classic (model-driven) techniques

In general, classical approaches apply threshold-based techniques to detect or separate interference in GNSS signals (Rügamer et al., 2013; van der Merwe et al., 2017; Bartl et al., 2017). However, those techniques only show a high detection accuracy on synthetic data and degrade significantly on realistic data. Instead, Gross and Humphreys (2017) propose a more sophisticated mechanism. They apply Maximum Likelihood to distinguish if a signal is clean or affected by multipath, spoofing, or jamming. However, they can only detect jammers (average detection accuracy of about 93% on synthetic data), but cannot classify the jammer type.

The main weakness of model-driven methods is that they cannot distinguish anomalies close to the noise floor, as a sensitive threshold method may result in false positive results (Yang et al., 2012). These methods also suffer from real multipath effects that introduce additional information or confusion into the signal pattern that they cannot resolve either (Pirsiavash et al., 2017).

## 2. Modern (data-driven) techniques

Unlike model-driven techniques, supervised Artificial Intelligence (AI) techniques (i.e., data-driven) can implicitly learn "thresholds" or informative patterns that return accurate detection and classification even close to the noise floor (Yang et al., 2012). And of course, AI-based methods can exploit exogenous information such as multipath components or other sources of interference in the signal to map even more specific patterns just as specific classes (Niitsoo et al., 2019; Feigl et al., 2020). In the case of multipath, AI-based (fingerprinting) methods (Niitsoo et al., 2019) learn to map the patterns introduced by multipath to a reference class (Feigl et al., 2021). In addition, suitable AI-based methods can even implicitly denoise effects to enable accurate and robust classification (Feigl et al., 2018b,a,c).

To adapt those techniques to the GNSS community, Morales Ferre et al. (2019) investigated an SVM with Radial Basis Function (RBF) kernel and a simple CNN to detect and classify interference on a dataset with five interference classes and a class with no interferences that represent real GNSS data from the GPS L1 frequency band, and added self-created synthetic interference signals. Their SVM classifies the six classes on average with an accuracy of about 94.9% whereas their CNN only results in 91.3% respectively. The authors conclude that their deeper neural network (i.e., CNN) is more difficult to train so SVM performs best. However, it is unclear whether they trained the CNN correctly and their dataset (with synthetic interference) is really comparable with real applications, as multipath effects are smeared along a radio propagation path. Similar to Morales Ferre et al. (2019), Swinney and Woods (2021) combines power spectral density (PSD) images, spectrograms, raw constellations, and histogram signal into a single image and feed it into an SVM and a Decision Tree (DT) classifier. Their mean classification accuracy is about 98% and 96.3% respectively. However, similar to Morales Ferre et al. (2019), they only evaluate their methods using synthetic GNSS and interference data. Thus, for both studies of Morales Ferre et al. (2019) and Swinney and Woods (2021), it remains unclear if and how their methods would perform on real-world data.

Voigt (2021) bases his research on Morales Ferre et al. (2019) and compares different neural network architectures for real-world data classification to show that deeper neural networks can outperform both model-driven and classical machine learning techniques. Unlike Morales Ferre et al. (2019) and Swinney and Woods (2021), Voigt (2021) is the first to compare different neural network architectures for real-world data classification. Voigt (2021) shows that ResNet18 (He et al., 2015) (average classification accuracy of 98.8%) and TS-Transformer (Zerveas et al., 2021) (average classification accuracy of 98.9%) outperform most modern methods on an unpublished dataset. However, signal properties in the time and frequency domains are similar in the presence and absence of jammers. Thus, we believe that employing these methods individually discards information along the optimization and so, degrades the classification accuracy. Furthermore, as Voigt (2021) only evaluated his methods on 306 interference samples of real-world data, he can only detect and not classify. It is unclear how ResNet18 and TS-Transform perform on a more realistic variety of data with multipath effects, variable distances and power levels, motion dynamics, and real-world noise.

To conclude, state-of-the-art snapshot-based data-driven methods, such as SVM (Morales Ferre et al., 2019) and CNN (Voigt, 2021), cannot exploit potential cross-snapshot time information but still outperform classic model-based techniques (van der Merwe et al., 2017), such as pattern recognition and mathematical formulations, as their nonlinear functions classify accurately even in challenging scenarios, e.g., dynamic environments with moving objects or multipath environments. One reason is that a snapshot only contains data from a few milliseconds without real temporal context and another reason is that they learn a mapping approximation directly from data that implicitly describes non-deterministic or nonlinear functions without any additional modeling effort. However, classic learning-based methods, (1) either do not employ time and context at all, e.g., RF (Breiman, 2001), SVM (Noble, 2006), and CNN (Fukushima and Shouno, 2015); (2) only consider time-dependent local phenomena (i.e., spatial features) in snapshots, e.g., ResNet (He et al., 2016) and Temporal CNN (TCN) (Wu et al., 2021); or (3) only consider time-dependent global phenomena (i.e., time-sensitive features) in sequences but may ignore local phenomena, e.g., Recurrent Neural Networks (RNNs). Thus, when used individually, state-of-the-art methods may not respect potential complementary information that, when used in combination (temporal and spatial), may increase classification accuracy and robustness. Although our framework is based on the fundamental methods of Voigt (2021), we claim that using both methods individually discards information along the optimization and degrades the classification accuracy. Hence, in contrast to Voigt (2021), we propose a pipeline that exploits both the spatial and the temporal relationships between samples when we fuse (joint loss function) ResNet18 (He et al., 2015) and TS-Transformer (Zerveas et al., 2021) to compensate for knowledge gaps of individual methods.

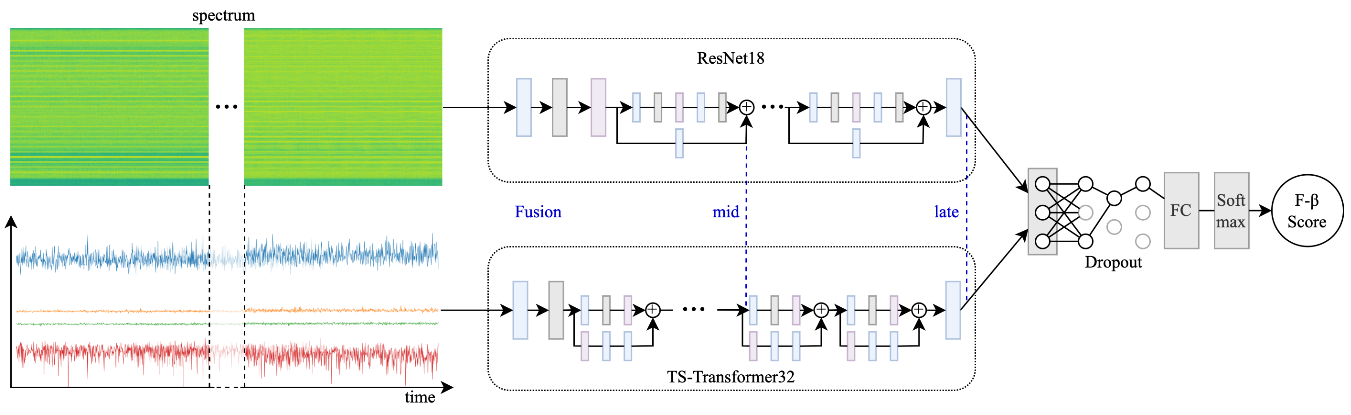


**Figure 2:** Pipeline of our MML architecture. Our method implicitly fuses spectrum (low-frequency 0.2 Hz) and time-domain (high-frequency 1 Hz) data as it employs prominent multimodal learning techniques, merges their losses, uses Monte Carlo dropout to measure reliability considering entropy, and classifies multipath-resistant GNSS interference.

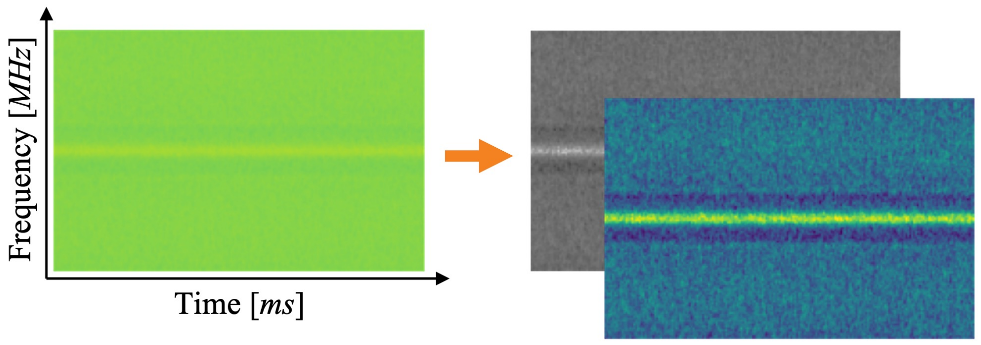


**Figure 3:** Pre-processing of the spectrograms. Images are converted to grayscale and a colormap (blue to green) increases the contrast.

## III. METHOD

This Section introduces the concept of our novel classification pipeline. First, Section III.1 introduces the key idea and the pipeline thereof. Next, Section III.2 explains how we pre-process both spectral and temporal input data. Then, Section III.3 introduces the fundamental neural network architectures and different prominent fusion techniques. Finally, Section III.4 provides insights on our uncertainty estimation and regularization component.

### 1. Idea and Pipeline

We propose a multi-stage framework (pre-processing of features, their fusion, classification, and uncertainty estimation). We first feed the low-rate (0.2 Hz) snapshots of spatial features to our MML estimator, a ResNet18 (He et al., 2016), as a spectrogram (i.e., images). Second, we feed the higher-rate (1 Hz) sampled or interpolated time series data to our MML estimator, a TS-Transformer (Voigt, 2021). We process time series and spectrogram data separately and perform a late or intermediate fusion of the features from ResNet and TS-Transformer to perform the final classification. Section III.3 describes the fusion techniques in detail. In a post-processing step, Monte Carlo dropout (Gal and Ghahramani, 2016) is applied to the fused layers to assess the uncertainty of each estimate, see Section III.4 for details. In a final step, we calculate the overall accuracy using a fully connected layer and the SoftMax function (Section III.4 provides details). We use renowned scores (e.g., $F - \beta$=2 score) to evaluate the performance and efficiency of our MML framework, for details see Section III.4.

### 2. Pre-processing Pipeline

Our pre-processing is divided into two steps according to the two input types of two different sensor platforms: 1) pre-processing of spectrograms and 2) pre-processing of time series data. Note that, we describe the sensors in Section IV in detail.

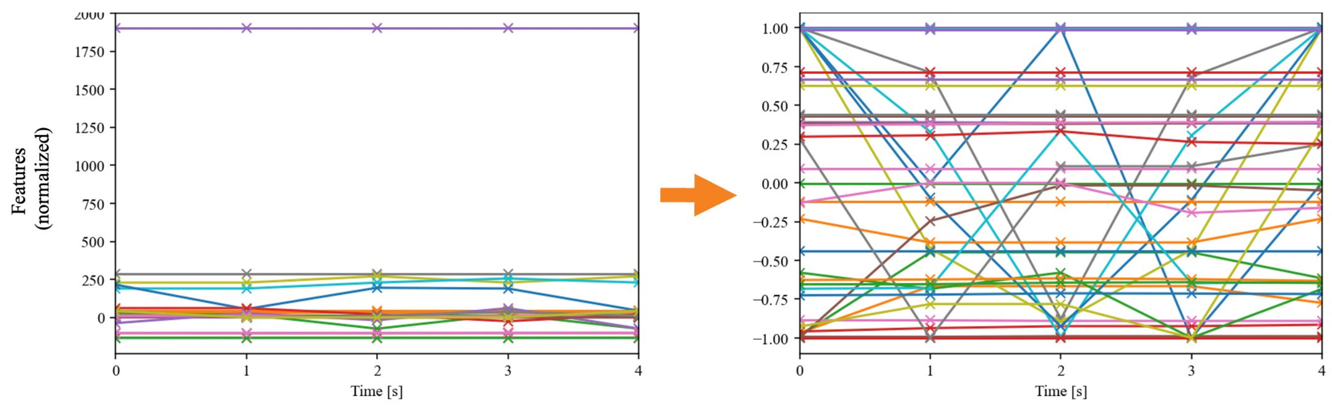


**Figure 4:** Pre-processing of the time series data.

*a) Pre-processing of the spectrograms*

A Short-Term Fourier Transform (STFT) using a Fast Fourier Transform of size 616, is calculated form a 20ms raw IQ snapshot, from a medium-end (ME) sensor. Snapshots of 20 ms raw IQ data from the medium-end (ME) sensor are Fourier transformed and stored as a 3D-(RGB)-image-matrix (dimensions: 775 × 616 in pixels). We scale down the images by 4 (dimensions: 194 × 154 in pixels) and convert them to grayscale, reducing the input feature size. There is a 50% chance they will be flipped horizontally to increase variation. Figure 3 provides some details of the pre-processing of spectrogram input data.

*b) Pre-processing of the time series data*

Using the Low Cost (LC) sensor's SDR to record 64 power spectral density (PSD) and 64 kurtosis values, this data is combined with mode, number of satellites, and carrier-to-noise density ratio ($C/N_0$) per second of the *u-blox* sensor, i.e., receiver. All these features are normalized as seen in equation 1, by each feature over the whole duration of the experiment and a sliding window of 5 s is applied. Figure 4 provides some details of the pre-processing of time series input data.

$$\text{Normalization: } x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

## 3. Main-processing Pipeline

By fusing the raw data of a typical medium-end (ME) sensor (at a low-sampling 0.2 Hz rate) with characteristics of a low-cost (LC) sensor (at a high-sampling 1 Hz rate), we claim to improve the accuracy and robustness of our MML classifier. Our MML aims to extract meaningful information from both modalities and exploits the combined feature representation to estimate accurate classification results. In general, fusion may be achieved at the decision level (i.e., late fusion) or at the intermediate level. Although experiments in neuroscience (Karpathy et al., 2014) and Machine Learning (ML) (Schroeder and Foxe, 2005) suggest that mid-level fusion may promote learning, late fusion is still the predominant method used for multimodal learning (Abavisani et al., 2019; Katsaggelos et al., 2015). Thus, we evaluate both fusion techniques in our MML method: (1) the late fusion method, which fuses high-level features from respective sources before we employ them for classification; and (2) the intermediate fusion, that fuses the features at intermediate levels (see Fig. 7). To achieve intermediate fusion, we apply the Multimodal Transfer Module (MMTM) (Joze et al., 2020), which can be added at different levels of the feature hierarchy to enable slow modality fusion. This may help to recalibrate the channel-wise features in each neural network stream. Note that we describe the different fusion techniques in detail in Section III.1.

Now that we know that there are different fusion techniques, let us describe in detail the neural network architectures on which our pipeline is based and to which we apply our fusion techniques.

*a) ResNet18*

We employ ResNet (the most prominent residual learning framework) to enable the training of very deep networks (He et al., 2015). The trick with residual networks is that the gradient is routed much deeper into the network and is therefore more durable, so it cannot vanish as quickly. Conversely, essential information remains deep in the network and can be used profitably by deeper layers. This means that residual networks are easier to optimize and can gain accuracy with a significantly greater

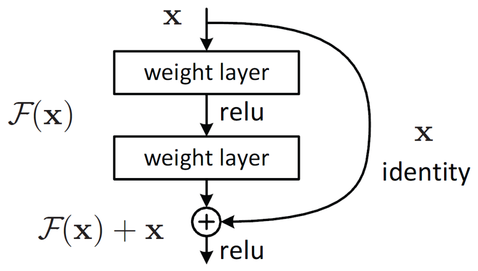


**Figure 5:** Semantic ResNet architecture.

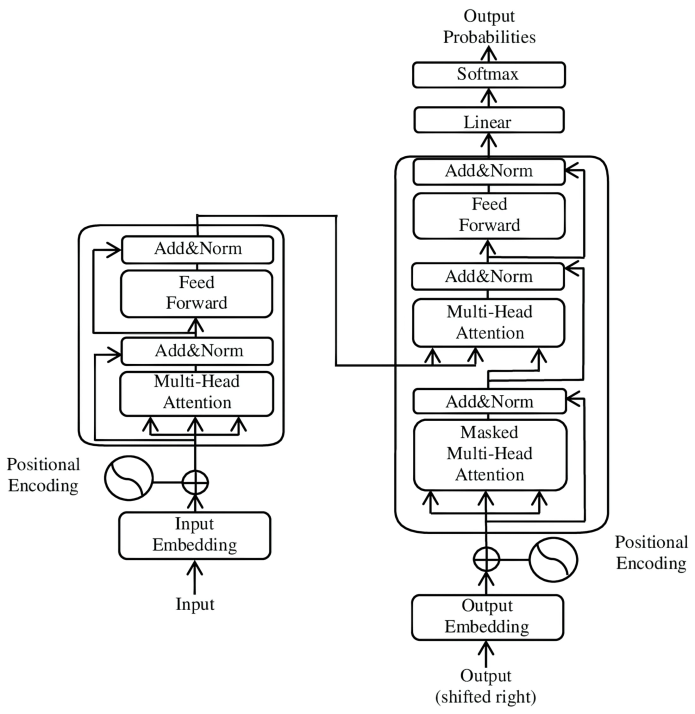


**Figure 6:** Semantic Transformer architecture.

depth. Interestingly, we employed a residual network that was pre-trained on the ImageNet dataset[1] with a depth of 18 layers and fine-tuned it on our GNSS-specific data.

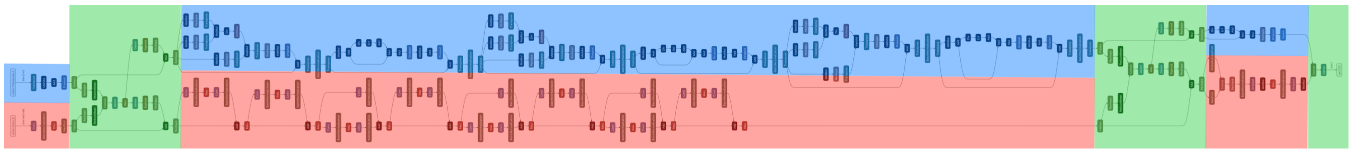
The formulation of $F(x) + x$ can be realized by feed-forward neural networks with shortcut connections. Shortcut connections are those skipping one or more layers shown in Figure 5. The shortcut connections perform identity mapping. Their outputs are added to the outputs of the stacked layers. Using residuals, the network can easily gain accuracy from greatly increased depth.

*b) TS-Transformer*

Transformer (Zerveas et al., 2021) employs a novel architecture based at its core on attention modules. Basically, there are two central concepts of attention: 1) self-attention, in which the attention mechanism pays attention to the input sequence and determines a new representation of each symbol of sequence based on the context of the entire sequence (Vaswani et al., 2017); and 2) multi-head attention, where we replace each attention operation with multiple parallel attentions on the same input (Vaswani et al., 2017), to allow the model to jointly attend to information from different representation sub-spaces at different positions. With a single attention head, averaging inhibits this. Today, the Transformer architecture is known to outperform all other time-sensitive methods in most research areas (Zerveas et al., 2021; Devlin et al., 2019).

The task of the encoder, on the left half of the Transformer architecture in Figure 6, is to map an input sequence to a sequence

[1]`https://www.image-net.org`


**(a)** MMTM architecture. 661,775,536 parameters in total.

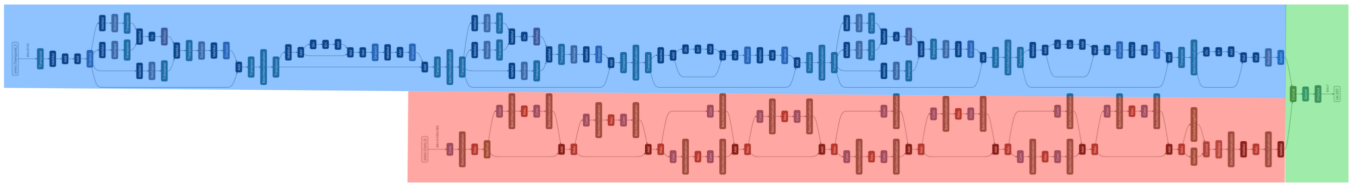
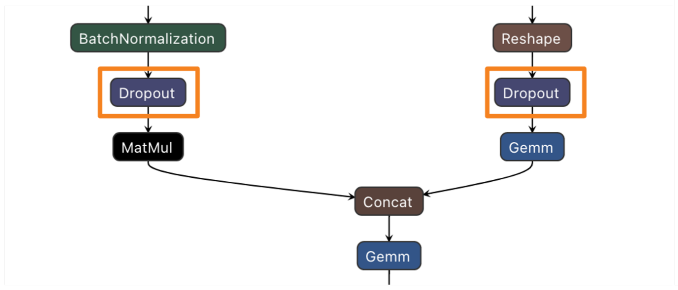
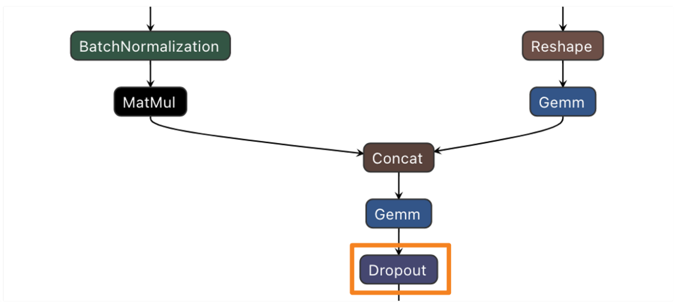

**(b)** Late fusion architecture. 12,106,544 parameters in total.

**Figure 7:** Overview of two different multimodal fusion architectures that we investigate and evaluate: (a) intermediate and (b) late fusion. Note that both architectures consist of ResNet (red color), TS-Transformer (blue color), and the respective fusion layers (green color).



**(a)** LateFusion with separated Dropout layer.



**(b)** LateFusion with combined Dropout layer.

**Figure 8:** LateFusion methods.

of continuous representations, which is then fed into a decoder. The decoder, on the right half of the architecture, receives the output of the encoder together with the decoder output at the previous time step, to generate an output sequence.

### *c) Fusion methods*

In this paper, we investigate one intermediate and two different late fusion methods. In the following, we will examine the late fusion methods LateFusion and SoftFusion and the intermediate fusion methods Multimodal Transfer Module (MMTM).

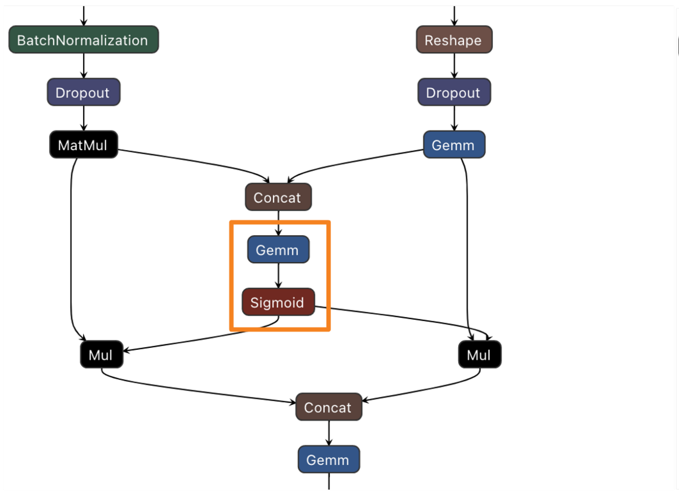
### *d) LateFusion*

LateFusion (Snoek et al., 2005) is the simplest and most commonly used fusion method. It merges data after a separate full processing in different unimodal streams. The structure of the LateFusion based on concatenation is shown in Figure 8. During fusion, we concatenate the temporal features and the spatial features each. We investigated two variations, (a) with a separated Dropout layer for each net and (b) with a shared Dropout layer after the concatenation, see Figure 8. After the concatenation layer, a fully connected and a SoftMax layer classify the target.

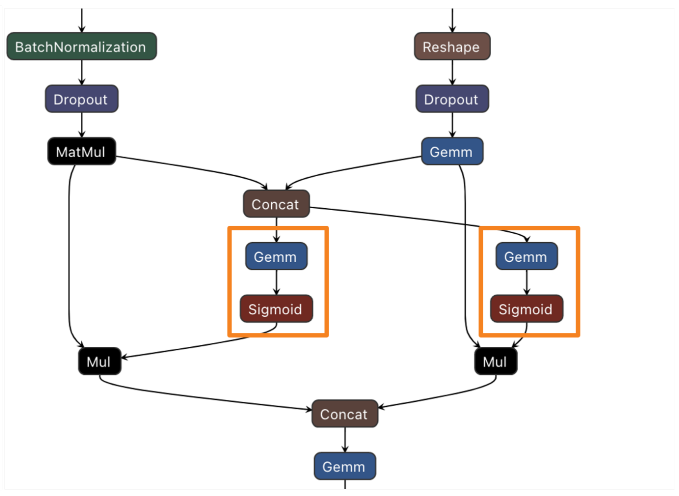
### *e) SoftFusion*

Figure 9 illustrates the structure of the SoftFusion module. We feed in the high-level features ($\mathbf{a}_V$ and $\mathbf{a}_I$) generated by the ResNet and TS-Transformer models. The input features are fused based on the attention mechanism introduced by Chen et al. (2019). A pair of continuous masks, $\mathcal{S}_{\text{ResNet}}$ and $\mathcal{S}_{\text{TS-Transformer}_\text{I}}$, are introduced by:

$$\begin{aligned}\mathcal{S}_{\text{ResNet}} &= \text{Sigmoid}_V([\mathbf{a}_V;\mathbf{a}_I])\\ \mathcal{S}_{\text{TS-Transformer}_\text{I}} &= \text{Sigmoid}_I([\mathbf{a}_V;\mathbf{a}_I]),\end{aligned} \tag{2}$$



**(a)** SoftFusion.



**(b)** SoftFusion 2.

**Figure 9:** SoftFusion methods.

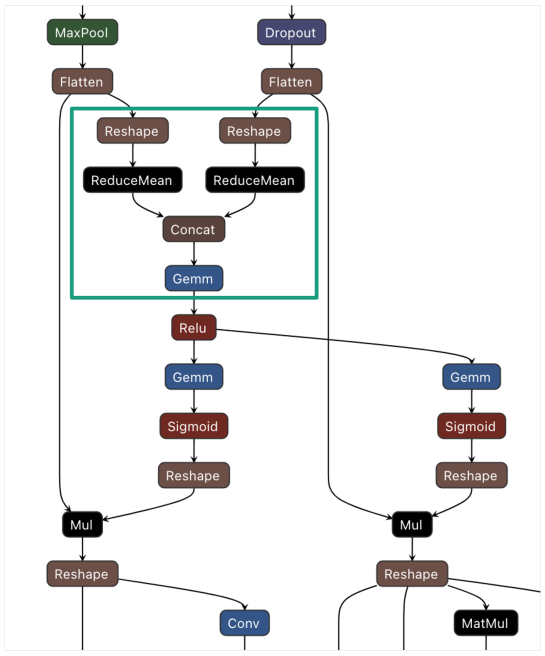


**Figure 10:** MMTM fusion method.

to perform SoftFusion. The masks $\mathcal{S}_{\text{ResNet}}$ and $\mathcal{S}_{\text{TS-Transformer}_\text{I}}$ are the masks applied (element-wise product $\odot$) to the features $\mathbf{a}_V$ and $\mathbf{a}_I$ learned by the neural network by conditioning on both features by:

$$g_{\text{soft}}(\mathbf{a}_V, \mathbf{a}_I) = [\mathbf{a}_V \odot \mathcal{S}_{\text{ResNet}}; \mathbf{a}_I \odot \mathcal{S}_{\text{TS-Transformer}_\text{I}}]. \tag{3}$$

The sigmoid function finally re-weights each feature vector and preserves the order of coefficient values in the range $[0, 1]$ to produce the new re-weighted vectors. After the SoftFusion, the features are forwarded to the next residual unit (ResNet) or attention module (TS-Transformer) and finally to fully connected and dropout layers to perform the classification.

### *f) MMTM Fusion*

Inspired by the squeeze-and-excitation (SE) (Hu et al., 2018) module for unimodal CNNs, Joze et al. (2020) proposed a Multimodal Transfer Module (MMTM) that allows the fusion of modalities with different spatial dimensions. The key idea is, that the squeeze operation compresses the spatial information into channel descriptors via a global average pooling operation over spatial dimensions of input features that enables fusion of modalities of arbitrarily feature dimensions. Instead, the excitation operation generates the excitation signals using a simple gating mechanism like a sigmoid function, that allows the suppression or excitation of different filters in each stream. Figure 10 illustrates the structure of the MMTM module (Joze et al., 2020).

The matrices $\mathbf{R} \in \mathbb{R}^{N_1 \times \cdots \times N_K \times C}$ and $\mathbf{T} \in \mathbb{R}^{M_1 \times \cdots \times M_L \times C'}$ represent the features at any given level of ResNet and TS-Transformer networks that are the inputs to the MMTM module. Here, $N_i$ and $M_i$ represent the spatial dimensions, and $C$ and $C'$ represent the number of channels of ResNet and TS-Transformer, respectively. MMTM learns the global multimodal embedding to re-calibrate the inputs $\mathbf{R}$ and $\mathbf{T}$ using a squeeze-and-excitation operation on the input tensors $\mathbf{R}$ and $\mathbf{T}$. The squeeze operation enables fusion between the modalities $S_{\mathbf{R}}$ and $S_{\mathbf{T}}$ that have arbitrary spatial dimensions:

$$\begin{aligned} S_{\mathbf{R}}(c) &= \frac{1}{\prod_{i=1}^{K} N_i} \sum_{n_1,\ldots,n_k} \mathbf{R}(n_1,\ldots,n_K,c) \\ S_{\mathbf{T}}(c) &= \frac{1}{\prod_{i=1}^{L} M_i} \sum_{m_1,\ldots,m_L} \mathbf{T}(m_1,\ldots,m_L,c), \end{aligned} \tag{4}$$

that are further mapped into a joint representation $Z$ using concatenation and FC layer.

Excitation signals, $E_{\mathbf{R}} \in \mathbb{R}_C$ and $E_{\mathbf{T}} \in \mathbb{R}_{C'}$ are generated using $Z$, which are used to re-calibrate the input features, $\mathbf{R}$ and $\mathbf{T}$, by a simple gating mechanism:

$$\begin{aligned} \tilde{\mathbf{R}} &= 2 \times \sigma(E_{\mathbf{R}}) \odot \mathbf{R} \\ \tilde{\mathbf{T}} &= 2 \times \sigma(E_{\mathbf{T}}) \odot \mathbf{T}, \end{aligned} \tag{5}$$

where $\sigma(.)$ is a sigmoid function and $\odot$ is a channel-wise product operation.

In Section V we evaluate the effect of the fusion techniques on the classification accuracy of GNSS interference in detail.

## 4. Post-processing Pipeline

Although, we process time series and spectrogram data separately to decouple from the need to have data from both modalities for each snapshot/point in time, we perform a late or intermediate fusion of the features from ResNet and TS-Transformer to perform the final classification. In our post-processing step, we apply Monte Carlo dropout to the fused layers to assess the uncertainty of each estimate. And in the final step, we calculate the overall accuracy using a fully connected layer (FC) and the SoftMax function. We use $F - \beta$=2 score to evaluate the performance and efficiency of our MML framework.

### *a) Dropout*

With sparse data and/or a complex (with many parameters) network, the model can memorize the training data (so-called overfitting to the data) and, as a result, work great with the data it saw during training, but gives poor results for unknown data (Srivastava et al., 2014). To address these problems we apply Dropout, as it is a well-understood regularization technique to prevent overfitting. Dropout turns off a different set of neurons (with a predefined number of neurons) at each training step. Hence, with dropout, any information can disappear at any time during training. Therefore, a neuron cannot rely on just a few inputs, it must distribute its weights and pay attention to all inputs. As a result, it becomes less sensitive to input changes, leading to a better generalization of the model.

In addition to regularization, Dropout also provides a mean with an uncertainty estimate. As on each training iteration, dropout randomly selects the neurons to fail in each layer, according to that layer's dropout rate, a different set of neurons is dropped each time. Thus, the architecture of the model is slightly different each time. In consequence, the result is an average ensemble of many different neural networks, each trained on just one batch of data. Hence, the results of our neural network reflect the mean of an ensemble of the networks. We can also derive the variance (uncertainty) if we actively dropout during interference.

The Monte Carlo Dropout (MCD) model estimation can be computed as the average of T predictions:

$$\begin{aligned} p &= \frac{1}{T} \sum_{i=0}^{T} f_{nn}^{d_i}(x) \\ c &= \frac{1}{T} \sum_{i=0}^{T} \left[ f_{nn}^{d_i}(x) - p \right]^2 \end{aligned} \tag{6}$$

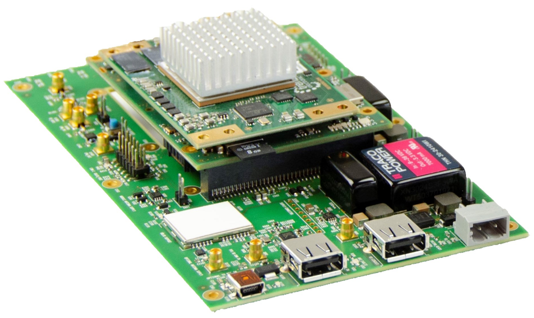


**Figure 11:** The hardware of the ME sensor (case removed).

However, we also apply Bayesian Active Learning by Disagreement (BALD) along with Dropout to estimate the uncertainty.

*b) Bayesian Active Learning by Disagreement (BALD)*

BALD is a state-of-the-art method for the uncertainty estimation of Bayesian models Subedar et al. (2019). It selects the instance that maximizes the mutual information between the predictions and the model parameters as seen in the following equation:

$$\begin{aligned} \text{BALD} : &= H(y^*|x^*, D) - \mathbb{E}_{p(w|D)}[H(y^*|x^*, w)] \\ H(y^*|x^*, D) &= -\sum_{i=0}^{K-1} p_{i\mu} \log p_{i\mu}, \end{aligned} \tag{7}$$

$H$ is the predictive entropy which captures a combination of aleatoric and epistemic uncertainty. The criterion is to select those data that maximize the parameter disagreements between the current training model and its following updates. For instance, if we sample many networks using MC Dropout, and they disagree about the output, this means that some of them are wrong.

Now that we know the concept of our multimodal fusion architecture and its parameters, let us explain our experimental setup and design to evaluate these methods (in Section IV) before we discuss the results in Section V.

## IV. EXPERIMENTAL SETUP

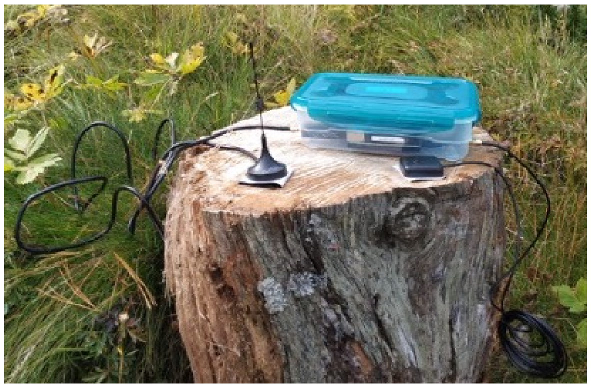
This Section introduces the concept of our experimental setup. First, Section IV.1 describes the hardware and software that we use across all of our experiments. Next, Section IV.2 reports the study design of our synthetic experiments, before Section IV.3 describes the study design of our real-world experiments. Finally, Section IV.4 provides details on the datasets that we use to validate and discuss our experiments and the results thereof (see Section V).

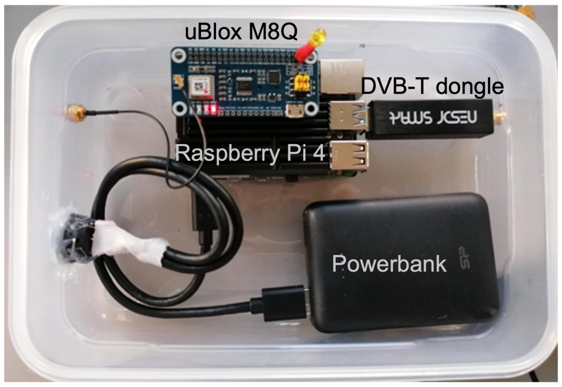
### 1. Hardware and Software Setup

*a) Hardware - Fraunhofer snapshot tool (ME)*

The hardware consists of the in-house developed Fraunhofer snapshot tool, that acquires short, wideband snapshots in both E1 and E6 GNSS bands simultaneously. The same hardware is used for snapshot-based real-time kinematic (RTK) positioning (O'Driscoll et al., 2022), and was originally developed for server-side GNSS positioning (Rubino et al., 2016). The Fraunhofer snapshot tool is setup to record 20 $\text{ms}$ raw IQ snapshots every second using a sample rate of 62.5 $\text{MHz}$ and an analog bandwidth of 50 $\text{MHz}$ and a bit-width of 8 bits. It embeds additional information in the snapshots, including receiver outputs from an onboard u-blox-M8 GNSS receiver, and interference mitigation status from an on-board HDDM (van der Merwe et al., 2021) (the snapshots for our application are derived from data before the HDDM). This additional information can be further evaluated in future. Figure 11 shows the sensor without the case. In the following, this sensor is referred to as ME. We always employed the ME sensor in our measurement campaign and generate one snapshot every 5 seconds.

Our pre-processing performs a Fast Fourier Transform (FFT) to transform the raw data of the ME sensor into the frequency domain. There is consequently an information gap of about 4 980 $\text{ms}$ between each data frame due to processing delays. To fill this gap we sample from the LC sensor: (high-rate) abstract temporal features, such as $C/N_0$ or automatic gain control (AGC) values (of the same sensor or even another sensor), or interpolate the IQ values (see temporal features in Figure 17).


(a) Exemplary scenario setup of our Low-Cost sensor.



(b) Top-view of our Low-Cost sensor system hardware.

**Figure 12:** Low-Cost sensor system hardware.

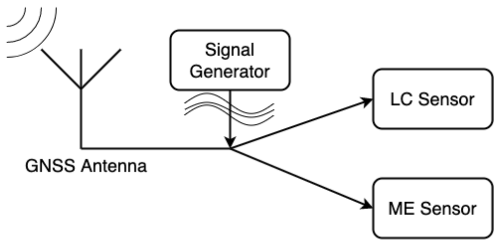


**Figure 13:** Schematic of the hardware setup in our laboratory. This setup provides real-world data with synthetic (inference). The real GNSS signal, from an outside receiver, is mixed with interference signals generated by a signal generator, connected via cables. The LC and ME sensors are recording this mixed signal simultaneously.

### *b) Hardware - Low-Cost Sensor (LC)*

Figure 12b shows the top-view of the prototypical version of our low-cost GNSS interference monitoring hardware (van der Merwe et al., 2022b). The central part of the system is a RaspberryPi 4B (4 GB RAM version) single-board computer (SBC), which does the primary processing, interfacing, and networking. GNSS processing is provided by a GPS RaspberryPi hat (pHat) containing a u-blox MAX-M8Q. This u-blox receiver processes GPS, Galileo, Beidou, GLONASS, QZSS, and SBAS signals in the L1 band, but only a maximum of three concurrent GNSSs. Therefore, it is configured to use GPS L1 C/A, Galileo E1 OS-B/C, and GLONASS G1 OS, to maximize the spectrum diversity.

The SBC connects to the pHat via a universal asynchronous receiver-transmitter (UART) bus over the general purpose input/output (GPIO) interface and decodes the National Marine Electronics Association (NMEA) messages at a 1 Hz rate. The GNSS receiver is primarily handled by the GPS service daemon (GPSD). The second sensor is a NeSDR SMArt v4 that connects over USB2.0, and contains an RTL2832U digital video broadcasting (DVB) radio-frequency front-end (RFFE). This sensor can be reconfigured as a software-defined radio (SDR) RFFE and delivers 8 bit complex I/Q samples at a maximum sample rate of 3.2 MHz. It is sufficient to receive narrow global navigation satellite system (GNSS) signals, such as GPS L1 C/A (1.023 MHz chipping rate), but insufficient to accurately process and localize most GNSS signals. van der Merwe et al. (2022b) provide details on the setup and specifications of the low-cost platform.

Despite its limitations of center frequency, sampling rate, number of simultaneously processable GNSS systems, and computational performance, van der Merwe et al. (2022b) showed that the system, especially the bandwidth suffices for detection, monitoring, and classification purposes. The received samples are further processed on the SBC. We employ the antennas that both systems u-blox and the NeSDR receivers provide (see Fig. 12a).

## 2. Data Acquisition: Synthetic Experiments

### *a) Labeling and Time Synchronization*

We recorded data from both sensors and stored them along with timestamps that are globally NTP-time-synchronized. We labeled each activity and measurement (LC, ME, and positions of the sensors and interference source) and stored separate files per subject. Thus, we can also use our data for localization purposes.

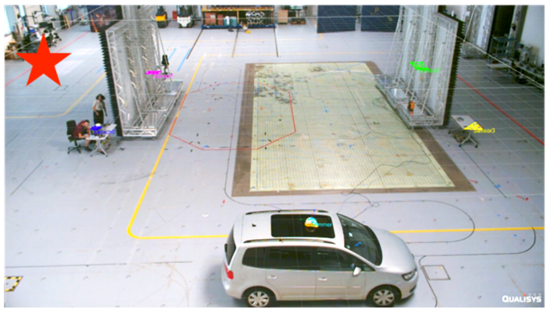
The real GNSS signal, from an outside receiver, is mixed with interference signals generated by an *N5182A MXG Vector Signal Generator*, connected via cables. This mixed signal is split and recorded by the LC and ME sensors simultaneously. While recording with the LC and ME Sensor the waveforms were played with different duration and -40 – 0 $\mathrm{dB}$ power levels. The generated interference signals had a bandwidth between 2.5 – 35 $\mathrm{MHz}$ and a period of 0.1 – 10 $\mathrm{ms}$. According to the scenario each waveform is repeated for 5 to 120 $\mathrm{s}$.

Note that, all experiments were performed using the GPS L1 band (1575.42 $\mathrm{MHz}$). No one was harmed during the experiments.

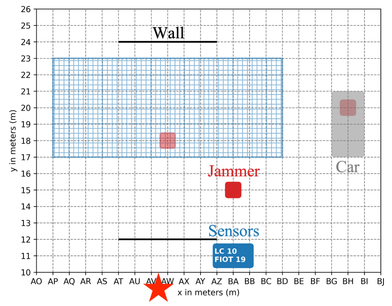
## 3. Data Acquisition: Real-world Experiments

A GNSS signal repeater (see Fig. 15) is mounted on the roof and is used to counter the attenuation caused by concrete, see the red star in Fig. 14 and Fig. 16.

Our intensive measurement campaign records for all selected waveforms, six different scenarios at the Fraunhofer test center – LINK Halle – in Nuremberg, Germany. Table 1 lists the characteristics of each scenario and describes the study design briefly.



**(a)** Exemplary scenario setup in the Fraunhofer test-center.



**(b)** Top-down view on the Fraunhofer test-center.

**Figure 14:** Real-world setup. In the Fraunhofer test-center, the GNSS signals are relayed via a repeater on the roof, see Figure 15 and the red star in the top left corner in (a) and the bottom in (b). We recorded different static scenarios (distances between jammer and the blue sensor station: 2 $\mathrm{m}$, 4 $\mathrm{m}$, 5 $\mathrm{m}$, 7 $\mathrm{m}$, 10 $\mathrm{m}$, 11 $\mathrm{m}$, 12 $\mathrm{m}$) including reflector- and diffuser walls and a car. Note that we attenuate the signals such that 1 $\mathrm{m}$ in our setup mimics 32 $\mathrm{m}$ in the real world.

**Table 1:** Description of all recording scenarios.

| **Date** | **Duration** [min] | **Scenario** | **Specification** | **Speed** [km/h] | **Distance** [m] |
|---|---|---|---|---|---|
| **07/01/22** | 23 | Static | Hall | 0 | 4 |
| **07/01/22** | 23 | Static | Hall | 0 | 5 |
| **07/01/22** | 23 | Static | Hall | 0 | 7 |
| **07/01/22** | 12 | Dynamic | Hall | 3 | 2 - 12 |
| **07/01/22** | 12 | Dynamic | Hall | 6 | 2 - 12 |
| **07/01/22** | 12 | Dynamic | Hall, Walls | 3 | 2 - 12 |
| **07/04/22** | 26 | Static | Hall, Walls | 0 | 2 |
| **07/04/22** | 26 | Dynamic | Hall, Walls | 6 | 2 - 12 |
| **07/04/22** | 26 | Static | Hall, Walls, Car, antenna on the dashboard | 0 | 10 |
| **07/04/22** | 26 | Static | Hall, Walls, Car, antenna inside glove box | 0 | 10 |
| **07/04/22** | 26 | Static | Hall, Walls | 0 | 11 |
| **07/25/22** | 54 | Static | Labor, -40 to 0 dB | - | - |
| **07/26/22** | 108 | Static | Labor, -40 to 0 dB | - | - |

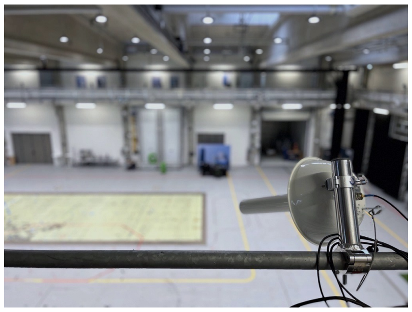
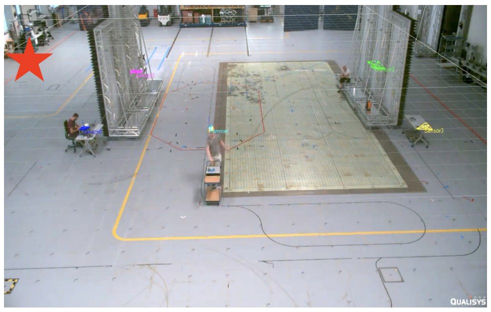
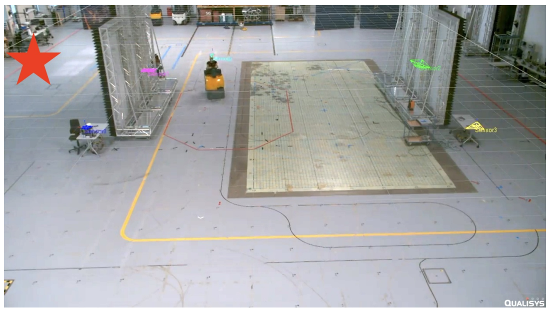


**Figure 15:** GNSS repeater in the Fraunhofer test-center.



**(a)** Walking with an interference signal.



**(b)** Driving with an interference signal.

**Figure 16:** Dynamic scenarios.

The first scenarios were recorded while the interference source and the sensors were statically mounted 4 $\mathrm{m}$, 5 $\mathrm{m}$, or 7 $\mathrm{m}$ apart. The dynamic scenarios were realized by walking or driving a forklift with the interference source in the hall 2 $\mathrm{m}$ to 12 $\mathrm{m}$ distant from the sensors. During 2 dynamic scenarios we also placed 6 (3×6 $\mathrm{m}$) walls with a reflector surface (white) on the one side and a diffuser surface (black) on the other side (see Fig. 16). We used these walls also for 4 static scenarios with 2 $\mathrm{m}$ and 11 $\mathrm{m}$ between interference source and sensors. In addition to the walls we also parked a car (VW Touran) inside the hall and placed the antenna of the interference source once on the dashboard and once in the glove box (see Fig. 14a). We attenuated the transmitter by 30 $\mathrm{dB}$, so, 1 $\mathrm{m}$ in this environment corresponds to 31.6 $\mathrm{m}$ in the real world (see equation 8) (van der Merwe et al., 2022b):

$$\frac{d'}{d} = \sqrt{\frac{P_\mathrm{r}}{P'_\mathrm{r}}} = 10^{\dfrac{-30\text{ dB}}{20}} = 0.0316 \tag{8}$$

## 4. Datasets

We use realistic data recorded in a deterministic test environment, to benchmark our methods. We use different sensors with different sampling rates, resolutions, and duration (see Fig. 17).

For classification, we use six main classes of interference, namely: Chirp, FreqHopper (Frequency Hopper), Modulated, Multitone, Pulsed, and Noise interference, which are typically challenging to detect or classify as their signal characteristics in the time and frequency domain with and without interference are similar. We selected 33 different, interesting waveforms as described in Tab. 2 from all six classes (see Fig. 18) and the GNSS signal without interference.

In general, we almost balanced the number of samples per class, see Figure 19. For the uncertainty estimation, we fully balanced the dataset. In total, we recorded about 309 $\mathrm{min}$ = 5.15 $\mathrm{h}$ of static and about 62 $\mathrm{min}$ = 1 $\mathrm{h}$ of motion data with the 2 sensors (see Tab. 1). In our study, we found that a sliding windows of size 5 $\mathrm{s}$ and no overlap hold enough prominent features at low computational costs to achieve the highest accuracy of the classification. Therefore, the description of the dataset, the parameterization of the methods, and the results in Section V refer to this optimal setting of the parameter.

## 5. Model Parametrization

In this Section, we report the final configuration parameters for each method: Baseline ResNet, baseline TS-Transformer, our LateFusion, our SoftFusion, and our MMTM model.

**Table 2:** Specification of all selected waveforms.

| **Category** | **Bandwidth** [MHz] | **Number of bands** | **Intermediate Frequency** [MHz] | **Specification** |
|---|---|---|---|---|
| **Noise** | 12.78 | 2 | ± 11.5087 | BOC(10, 5), BOC(15, 12.5) |
| **Noise** | 4 | 1 | 0 | |
| **Noise** | 5 | 2 | ± 10.23 | BOC(10, 5) |
| **Noise** | 5 | 2 | ± 15.345 | BOC(15, 12.5) |
| **Noise** | 35 | 1 | 0 | |
| **Noise** | 50 | 1 | 0 | |
| **FreqHopper** | 2.5 | 2 | ± 10.23 | 1 $\mu$s Dwell time |
| **FreqHopper** | 5 | 2 | ± 10.23 | 100 $\mu$s Dwell time |
| **FreqHopper** | 5 | 2 | ± 15.345 | 1 $\mu$s Dwell time |
| **FreqHopper** | 5 | 2 | ± 15.345 | 100 $\mu$s Dwell time |
| **FreqHopper** | 5 | 2 | ± 10.23 | 1 $\mu$s Dwell time |
| **FreqHopper** | 35 | 1 | 0 | 100 $\mu$s Dwell time |
| **Chirp** | 10 | 1 | 0 | Linear, 10 $\mu$s repetition interval |
| **Chirp** | 35 | 1 | 0 | Linear, 10 $\mu$s repetition interval |
| **Chirp** | 10 | 1 | 0 | Linear, 100 $\mu$s repetition interval |
| **Chirp** | 35 | 1 | 0 | Linear, 100 $\mu$s repetition interval |
| **Chirp** | 10 | 1 | 0 | Linear, 1000 $\mu$s repetition interval |
| **Chirp** | 35 | 1 | 0 | Linear, 1000 $\mu$s repetition interval |
| **Chirp** | 35 | 1 | 0 | Parabolic, 100 $\mu$s repetition interval |
| **Modulated** | 10 | 2 | 0 | BOC(10, 5) |
| **Modulated** | 5 | 2 | 0 | BOC(15, 2.5) |
| **Modulated** | 10 | 1 | 0 | BPSK(10) |
| **Modulated** | 1 | 1 | 0 | BPSK(1) |
| **Pulsed** | 2.5 | 1 | ± 15.345 | 10% Duty cycle, 100 $\mu$s repetition time |
| **Pulsed** | 2.5 | 1 | ± 15.345 | 50% Duty cycle, 100 $\mu$s repetition time |
| **Pulsed** | 35 | 1 | 0 | 10% Duty cycle, 100 $\mu$s repetition time |
| **Pulsed** | 35 | 1 | 0 | 10% Duty cycle, 10 $\mu$s repetition time |
| **Pulsed** | 35 | 1 | 0 | 50% Duty cycle, 100 $\mu$s repetition time |
| **Pulsed** | 35 | 1 | 0 | 50% Duty cycle, 10 $\mu$s repetition time |
| **Pulsed** | 35 | 1 | 0 | 50% Duty cycle, 1 $\mu$s repetition time |
| **Multitone** | 18 | 1 | 0 | 9 tones, [0, 1, 2, 13, 14, 15, 16, 17] $\times$ 1.023 MHz |
| **Multitone** | 40 | 1 | 0 | 21 tones, [-20:2:20] $\times$ 1 MHz |
| **Multitone** | 35 | 1 | 0 | 8 tones, [-15, -14, -10, -1, 0, 1, +10, +14, +15] $\times$ 1.023 MHz |

**Table 3:** Total amount of data acquisition.

| **Setup** | **Spectrograms (ME) (Snapshots)** | **Time series (LC) (5 s sliding window)** | **Sum** |
|---|---|---|---|
| Laboratory setup | 5,000 | 9,000 | 14,000 |
| Real-world setup | 15,000 | 30,000 | 45,000 |
| **Sum** | 20,000 | 39,000 | 59,000 |

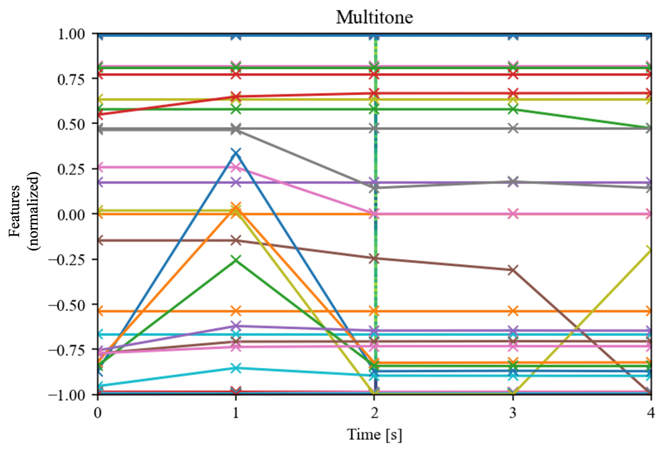


**Figure 17:** Comparing the sample duration of the LC and ME sensors. The vertical green line is the spectrogram of the ME Data. It is highly distorted along the time axis. The sample duration of the ME sensor is 20 ms. The horizontal lines are the time series data of the LC sensor. The sample duration of the LC sensor is 5 s.

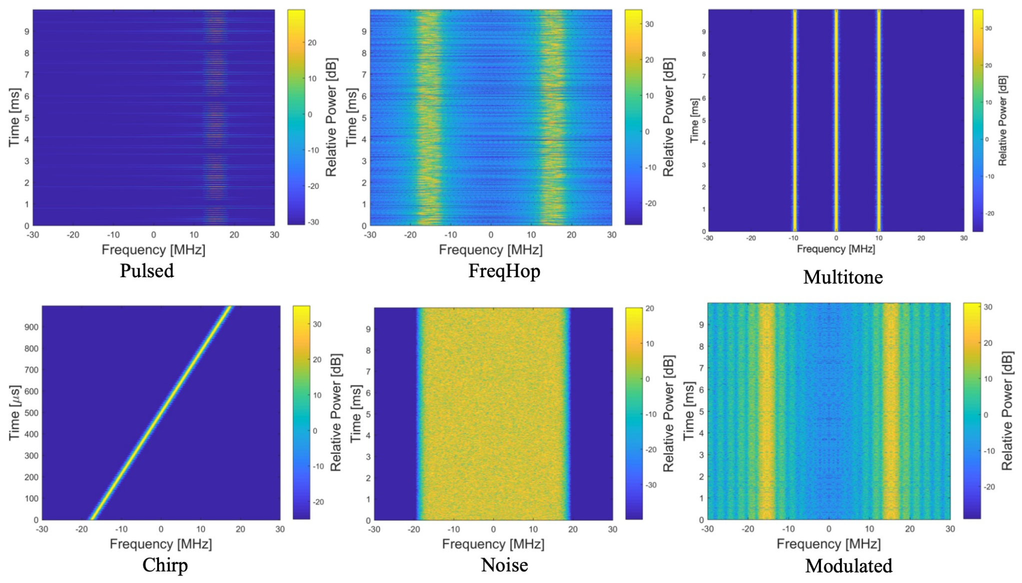


**Figure 18:** Exemplary waveforms of each 6 classes.

**Baseline ResNet.** As a baseline, we use ResNet18 as Voigt (2021) concluded. We choose weights pre-trained on the ImageNet[2] dataset, which contains 14 million images from 1000 categories. So, the network can already recognize for example lines and we do not need a large amount of training data. For training, we use a dropout rate of 0.5 and a learning rate of $1.2e^{-5}$.

**Baseline TS-Transformer.** As second baseline, we use TS-Transfomer64 with 3 encoders, 16 attention heads, and a feed-forward width of 256 as Voigt (2021) concluded. As an activation function, we use Gaussian Error Linear Units (GELU). Since our time series data uses a sliding window of 5 s we use a sequence length of 5. For training, we use a dropout rate of 0.1 and a learning rate of $1.2e^{-5}$.

**LateFusion model.** As a first fusion method, we use LateFusion by just concatenating the last layer of the baseline ResNet and TS-Transformer nets as described above. We investigate two variations, one with a separated dropout layer (see Fig. 8a) for each net and one with the shared dropout layer after concatenation (see Fig. 8b). For the separated dropout layer, we use a dropout rate of 0.5 for the ResNet and a dropout rate of 0.1 for the TS-Transformer. For the variation with a shared dropout layer, we use a dropout rate of 0.2. For training, we use $1.2e^{-5}$ as the learning rate.

[2]`https://www.image-net.org`

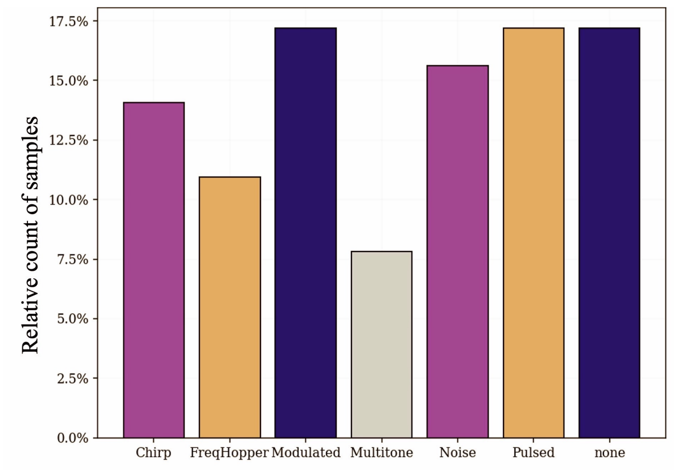


**Figure 19:** Example distribution of a scenario (pauses in between the experiments were removed). In total 100%.

**SoftFusion model.** As a second fusion method, we investigate SoftFusion (see Fig. 9a). This method also considers the last layer of each network. It uses the widely applied attention mechanism (Vaswani et al., 2017) to keep the most relevant features while discarding useless or noisy information. SoftFusion2 is a custom architecture and is very similar, but uses two Fully Connected Layers instead of only one (see Fig. 9b). For both variations we use a dropout rate of 0.5 for the ResNet and 0.1 for the TS-Transformer. For training, we use $1.2e^{-5}$ as the learning rate.

**MMTM model.** As an intermediate fusion method, we investigate MMTM (see Fig. 10). We add two MMTM blocks to fuse the ResNet and the TS-Transformer on an intermediate level. The first block was placed after the Max Pool layer of the ResNet and the first dropout layer of the TS-Transformer. The second block was placed just before the head of the ResNet and before the last encoder of the TS-Transformer. In the end, the last layer of the two networks are concatenated as with LateFusion. We use a dropout rate of 0.5 for the ResNet and 0.1 for the TS-Transformer. For training, we use a learning rate of $1.2e^{-5}$.

For the training of every model we use *Adam* as an optimizer and *CrossEntropyLoss* as the loss function[3].

## V. RESULTS

This Section reports the results of state-of-the-art methods and our novel fusion method along different experimental setups. First, Section V.4 discusses the value of using two sensors from different families over using two different modalities from a single sensor. Next, Section V.2 reports the results of the comparison of different fusion methods. Section V.3 provides even more details on the most promising LateFusion mechanic. Then, Section V.5 reports the results of the computational performance evaluation of both state-of-the-art and fusion techniques. Finally, Section V.6 discusses the effects of our uncertainty estimation component. Before we discuss our findings, let us first introduce the metrics we apply to evaluate our experiments.

### 1. Metrics

To evaluate our MML framework, we use the well-known pessimistic F-$\beta$-score (see equation 9) in all experiments to weight missed interference more heavily in the total error.

$$\begin{aligned} F_\beta &= (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \\ &= \frac{(1+\beta^2) \cdot \text{true positive}}{(1+\beta^2) \cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}} . \end{aligned} \tag{9}$$

[3]Model parameters: Training/Testing split: 80/20%, Epochs: 128;
TST from *tsai* (`https://timeseriesai.github.io/tsai`): Dimension: 64, Encoders $N_{enc}$: 3, Feed-forward width: 256, Sequence length: 5, Attention heads: 16, Activation function: Gaussian Error Linear Units (GELU);
ResNet18 from *fastai* (`https://docs.fast.ai`): Pre-trained on *ImageNet*,
Fusion: Dropout: 0.2, Learning rate: $1.2e^{-5}$, Optimizer: Adam, Loss function: CrossEntropyLoss;

**Table 4:** Comparison of the different fusion methods. *Training was performed on *Apple M1 Max*, 10-core CPU, 24-core GPU, using tsai: 0.3.1, fastai: 2.7.9, and pytorch: 1.12.1.

| **Fusion method** | **Accuracy** | **F-$\beta$=2 score** | **Training time*** [min] |
|---|---|---|---|
| LateFusion | **95.32%** | **95.32%** | **22:20** |
| SoftFusion | 93.91% | 93.90% | 22:27 |
| SoftFusion2 | 92.97% | 92.97% | 22:26 |
| MMTM | 89.70% | 89.69% | 42:25 |

In contrast to the state-of-the-art, we also evaluate the inference time and computational costs of all prominent methods independently and end to end (concerning our framework), see Section III.1.

By comparing the inference performance of the complete system with data from two different sensors and data from a single sensor, we show the costs and benefits of different variants. To the best of our knowledge, we are the first to investigate the reliability of the method with multipath effects on GNSS and interference signals in constrained and complex environments. As opposed to limiting the evaluation to ideally controlled laboratory evaluations.

## 2. Comparison of different fusion methods

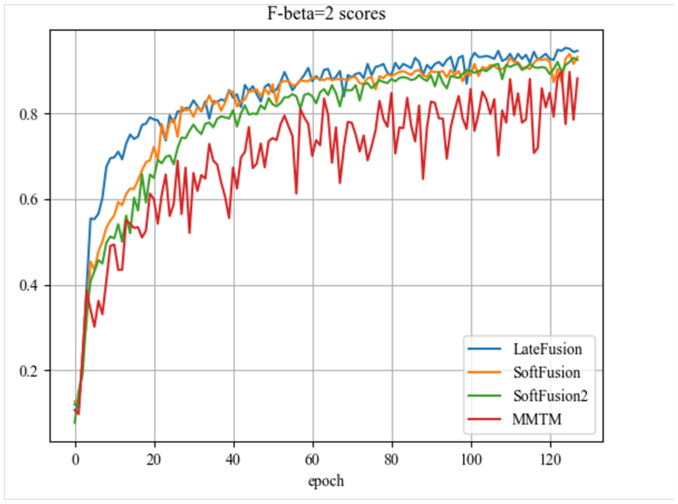


**Figure 20:** F-$\beta$=2 scores of different Fusion methods.

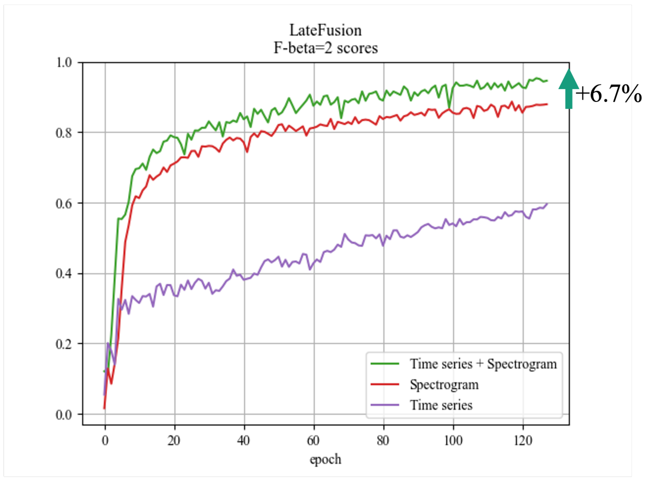
This experiment addresses the following research question (RQ1): "Which of the investigated fusion methods performs the best on the data recorded?". We compared LateFusion (with one shared dropout layer), SoftFusion, SoftFusion2, and MMTM. We used laboratory and real-world data for training and real-world data for testing. The training/testing split is 80/20%. To evaluate the different fusion methods, we use the well-known pessimistic F-$\beta$=2 score. It weights missed inference more heavily in the total error. After training for 128 epochs, we can clearly observe that LateFusion is the fastest and most accurate method tested, see Figure 20. Using MMTM, we reach a maximum F-$\beta$=2 score of 89.7% (see Fig. 4). So, MMTM is not appropriate for this use case.

To answer RQ1: LateFusion outperforms the other methods investigated in accuracy as well as in computational effort.

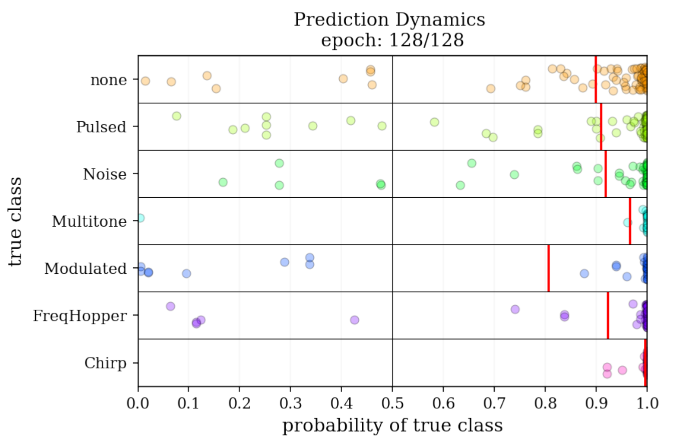
## 3. Spectrogram vs. time series of two different sensor types

This experiment addresses the following research question (RQ2): "How much does LateFusion of ResNet and TS-Transformer improve the accuracy compared to just ResNet or TS-Transformer?". We use ResNet and LateFusion (with one shared dropout layer) and train on lab and real-world data and test on real-world data. The training/testing split is 80/20%. We compare the pessimistic F-$\beta$=2 score (see Fig. 21a) and prediction dynamics (see Fig. 21b) after training for 128 epochs of ResNet, TS-Transformer, and LateFusion.

We also compared the confusion matrices of the model after training with ResNet and LateFusion (see Fig. 22).



**(a)** F-$\beta$=2 scores of LateFusion using only time series, spectrograms, or both time series and spectrograms.



**(b)** Prediction dynamics after training a model with LateFusion for 128 epochs. The dots on the left indicate the uncertainty. The red lines show the accuracy of each class.

**Figure 21:** Results of our pipeline when we employ LateFusion.

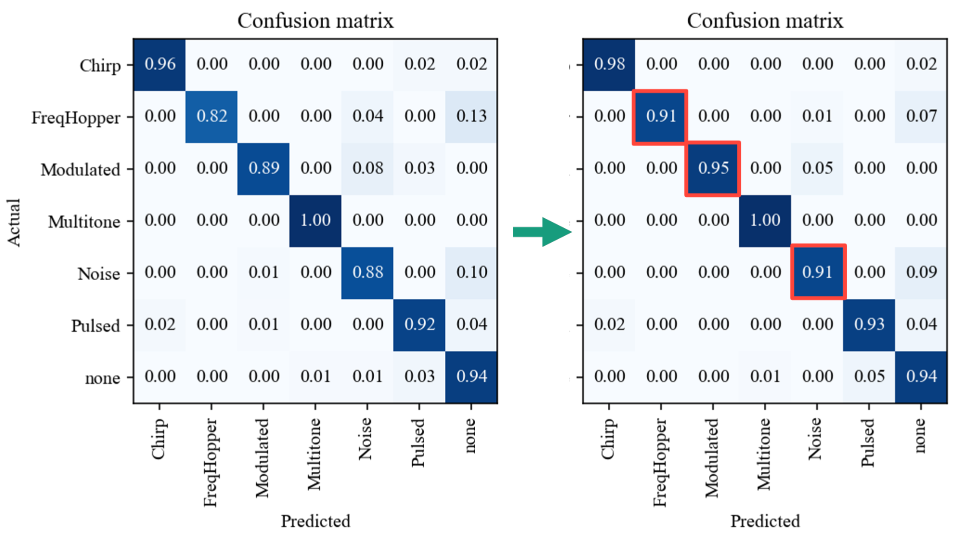


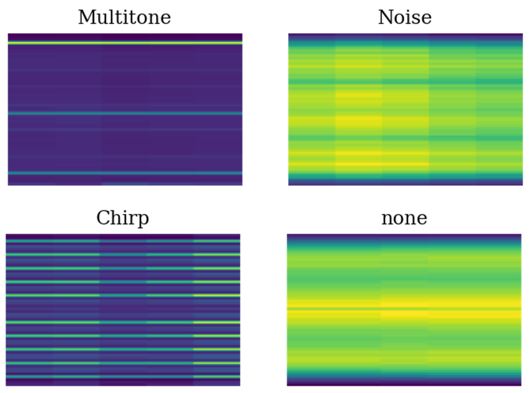
**Figure 22:** Comparison of confusion matrices of a model using LateFusion on spectrograms only (left) and using both time series and spectrograms (right). Especially the classification accuracy of *FreqHopper*, *Modulated*, and *Noise* is improved.

Using LateFusion on both, time series data and spectrograms, the F-$\beta$=2 score increases by 6.7% to 95.32%, in comparison to only using spectrograms (see Figure 21a). Especially the classification accuracy of *FreqHopper*, *Modulated*, and *Noise* is improved (see Figure 21).

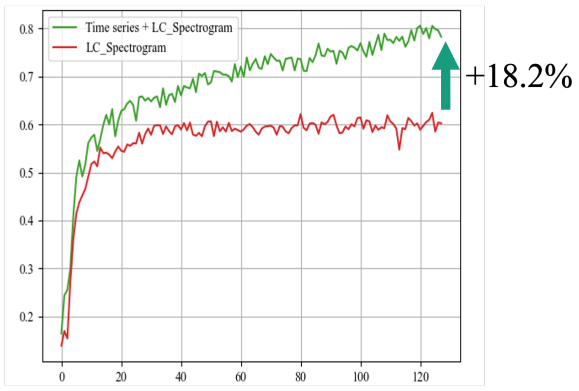
To answer the research question (RQ2): Using LateFusion improves the F-$\beta$=2 score by 6.7% compared to only using ResNet and by 35.8% compared to TS-Transformer. Especially, the classification accuracy of *FreqHopper*, *Modulated*, and *Noise* is improved.

## 4. Spectrogram vs. time series of one single sensor

This experiment addresses the following research question (RQ3): ”Does using two modalities from a single senor improve classification accuracy in contrast to using each modality individually?” In contrast to our key research question (see Section V.2), we investigate the effects of multimodal fusion of two modalities from the same sensor, i.e., the same datastream but from two different perspectives, namely spectrum and time. Hence, we apply fusion techniques only on data from one LC sensor to improve classification accuracy and keep the computational effort at a minimum.



**(a)** Every 5 s of LC data were converted into a spectrogram. The chirp signal has a bandwidth of 35 MHz but the LC sensor records only a bandwidth of 2.56 MHz.



**(b)** F-$\beta$=2 scores of LateFusion on time series and spectrograms, both generated from LC data.

**Figure 23:** Results of our pipeline when we employ LateFusion.

We generate spectrograms, each 5 s, from the 64 PSD values, recorded by the LC sensor, see Fig. 23a. By employing both data formats, time series and spectrograms of the same LC sensor, our LateFusion method increases in the F2 score of 18% to 80% on average, compared to only using the spectrograms (F2 score of 62%), see Fig. 23b. The accuracy of the classes *FreqHopper* and *Noise* were improved drastically (by 30% from F2<27% to F2<57% resp. 33% from F2<47% to F2<80%). However, in direct comparison with two modalities from two different sensor types, we found that using time and spectral features from two different sensor types (LC and ME) improves the classification by 15.1% up to 95.3%, but also increases the computational cost significantly from 1.28 ms to 1.39 ms. We think that the computational cost increases, as the spectral data from the ME sensor is much more complex than from the LC sensor (775×616 px vs. 610×450 px) and so, increases all consecutive processing steps, from FFT over sliding windows to convolutions.

To answer RQ3: we found that using two modalities from a single senor drastically improves its classification accuracy in contrast to using each modality individually. However, it also increases the computational effort by 9.6%. Instead, when we exploit the two modalities from two different sensors, both the accuracy drastically increases by 15.1% and the computational cost increases slightly by 8.5%.

## 5. Computational performance evaluation

We evaluated the computational effort to address the following research question (RQ4): "How much does the computational effort increase by using LateFusion in contrast to only using ResNet?".

The inference of LateFusion takes about 1.39 ms and is only 5% slower than using ResNet18 alone. Therefore, it can meet real-time capabilities. We also tried to reduce the computational cost by scaling down the spectrogram. The accuracy also drops fast (compare *LateFusion with 1/64 image size* in Figure 26).

To answer RQ4: we found that using LateFusion in contrast to using only ResNet increases computational effort by only about 5%.

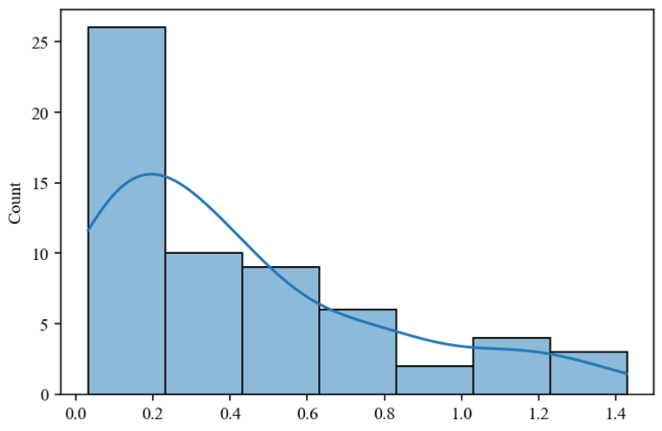
## 6. Uncertainty estimation

To answer the research question (RQ5) "Is it possible to detect unknown waveforms by only considering the uncertainty of a prediction?" we evaluated the uncertainty estimations.

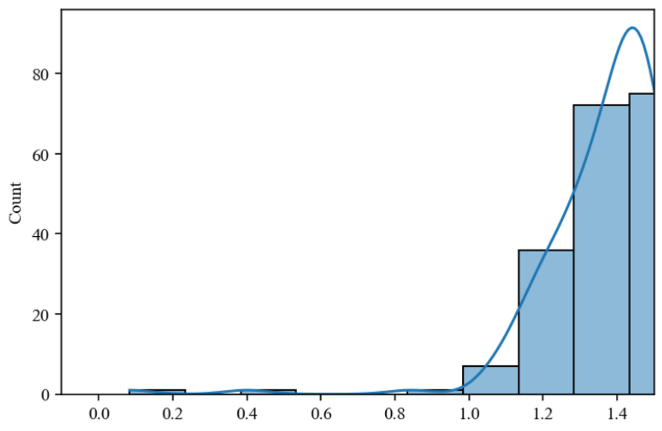
By integrating a dropout layer at the end of the network, the uncertainty of the prediction can be estimated. For each analysis 10 iterations of Monte Carlo dropout (MC-Dropout) as shown by Gal and Ghahramani (2016) were performed. For evaluation, we used entropy (Figure 24) and BALD (Figure 25) as specified by Gal et al. (2017).

### *a) Entropy*

We computed the entropy of predictions of samples from all known classes and compared it with the entropy of predictions of unknown waveforms. By analyzing the entropy of the prediction, unknown waveforms can be reliably detected but not classified (see Fig. 24).

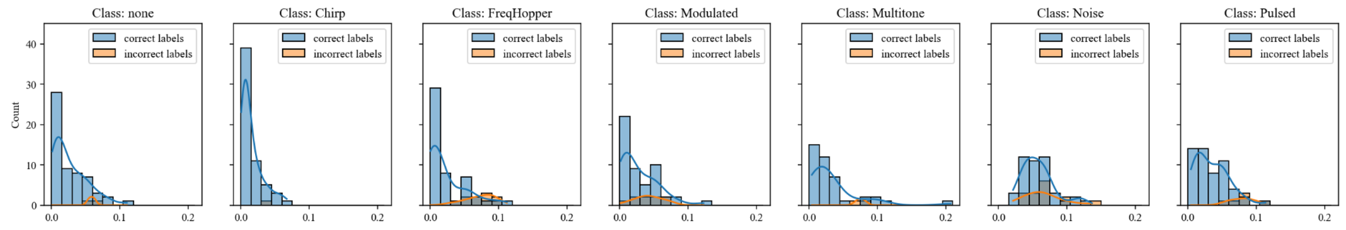


**(a)** Entropy of predictions, uncertainty of samples from all known classes.

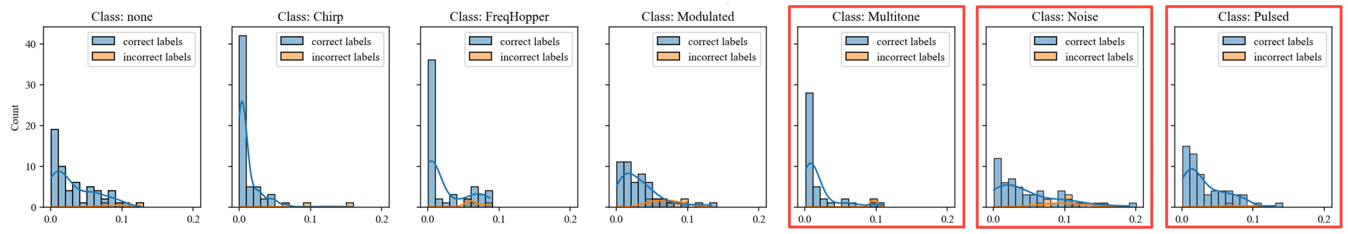


**(b)** Entropy of predictions, uncertainty of unknown waveforms.

**Figure 24:** Differences in the uncertainty of known and unknown waveforms.



**(a)** BALD estimation of a model using spectrograms only.



**(b)** BALD estimation of a model using both spectrogram and time series.

**Figure 25:** Comparison of BALD (Gal et al., 2017; Subedar et al., 2019) estimations for each class.

To answer RQ5: considering only the entropy of the prediction we can clearly distinguish between known and unknown waveforms.

*b) BALD*

In comparison to a network that employs both, spectrograms and time series data, the uncertainty for the classes *Multitone*, *Noise*, and *Pulsed* is reduced by about 20% (see Fig. 25).

To conclude, our fusion pipeline (with the best possible configuration, namely LateFusion) outperforms state-of-the-art in all different experiments. On average, we achieve a detection accuracy of 98% with a variance of ±0.2%. Instead, we achieve a classification accuracy of 95% with a variance of ±0.7% on average. Our pipeline performs a single inference at 1.39 ms.

## 7. Generalization

While training and testing on data recorded in the lab all samples can be classified correctly, so the accuracy is 100%. Using this trained model on real-world data, the accuracy drops to about 76.7%. While training on lab and real-world data and testing on real-world only we achieve a maximum accuracy of about 95.3%.

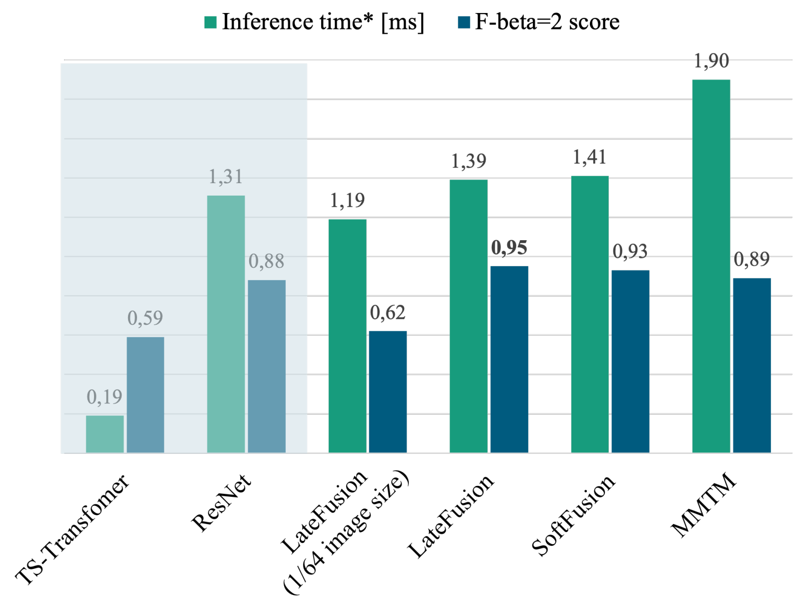


**Figure 26:** Comparison of inference times and F-$\beta$=2 scores. Standalone TS-Transformer and ResNet for reference only.

## VI. CONCLUSION

This paper proposes a deep learning framework that considers both the spatial and temporal relationships between samples (from the datastream of both the same or different sensor types) when fusing ResNet18 and TS-Transformers with a joint loss function to compensate for the weaknesses of both methods considered individually. The fundamental idea of using both spatial and time-sensitive features in our data-driven fusion framework allows us to reliably identify (on average F-$\beta$=2 score of about 98.2%) and categorize (on average F-$\beta$=2 score of about 95.3%) 33 different interference signals in seven categories even in multipath situations. This works even well (F-$\beta$=2 score about 80.1%), when we fuse both modalities only from a single bandwidth-limited low-cost sensor, instead of a fine-grained high-resolution ME and coarse-grained low-resolution low-cost sensor.

Our MML classifier (that performs best on average on two modalities from two different sensors) improves interference classification (on average F-$\beta$=2 score by about 6.7% to 95.3%) even in complex multipath environments, e.g., in tunnels or houses, and increases computational costs by only 5% up to 0.08 ms as it implicitly extracts local phenomena from the image classifier and extracts time series features.

Our real-world experiments show that our novel fusion pipeline with an adapted late fusion technique and uncertainty measure significantly outperforms the state-of-the-art (by 7% on average, i.e., in 95% of all cases our approach classifies correct on unknown data) even in complicated realistic scenarios with multipath propagation and environmental dynamics. In addition, our uncertainty estimates help separate unknown interference classes from those we trained our MML model on. Hence, we can guarantee that we almost always predict correctly and plausibly on known interference data and can justify when our model is not allowed to classify unknown data, as our uncertainty measures suggest.

For future work, we propose to investigate simplified versions of ResNet18 such as TCN and fuse them with recurrent neural networks instead of TS-Transformer.

## ACKNOWLEDGMENTS

This work has been carried out within the DARCY project, funding code 50NA2017, sponsored by the Federal Ministry for Economic Affairs and Energy (BMWi), and supported by the German Aerospace Center (DLR), the Bundesnetzagentur (BNetzA), and the Federal Agency for Cartography and Geodesy (BKG). This work was also supported by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics Data Applications (ADA-Center) within the framework of ”BAYERN DIGITAL II”.

## REFERENCES

Abavisani, M., Joze, H. R. V., and Patel, V. M. (2019). Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1165–1174.

Amin, M. G., Borio, D., Zhang, Y. D., and Galleani, L. (2017). Time-frequency analysis for GNSSs: From interference mitigation to system monitoring. *IEEE Signal Processing Magazine*, 34(5):85–95.

Bartl, S., Berglez, P., and Hofmann-Wellenhof, B. (2017). GNSS interference detection, classification and localization using software-defined radio. In *European Navigation Conf. (ENC)*, pages 159–169.

Borio, D. (2018). Sub-band robust GNSS signal processing for jamming mitigation. In *European Navigation Conf. (ENC)*, pages 72–83.

Breiman, L. (2001). Random forests. *Jo. Machine learning*, 45(1):5–32.

Broumandan, A., Jafarnia-Jahromi, A., Daneshmand, S., and Lachapelle, G. (2016). Overview of spatial processing approaches for GNSS structural interference detection and mitigation. *Proc. of the IEEE*, 104(6):1246–1257.

Chen, C., Rosa, S., Miao, Y., Lu, C. X., Wu, W., Markham, A., and Trigoni, N. (2019). Selective sensor fusion for neural visual-inertial odometry. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10542–10551.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. Intl. Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, volume 1, pages 4171–4186.

Dovis, F. (2015). *GNSS Interference Threats and Countermeasures*. Artech House GNSS technology and applications series GNSS interference, threats, and countermeasures. Artech House.

Elango, A., Ujan, S., and Ruotsalainen, L. (2022). Disruptive GNSS signal detection and classification at different power levels using advanced deep-learning approach. In *Proc. Intl. Conf. on Localization and GNSS (ICL-GNSS)*, pages 1–7.

Feigl, T., Eberlein, E., Kram, S., and Mutschler, C. (2021). Robust ToA-Estimation using Convolutional Neural Networks on Randomized Channel Models. In *Proc. Intl. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8.

Feigl, T., Kram, S., Woller, P., Siddiqui, R. H., Philippsen, M., and Mutschler, C. (2020). RNN-aided Human Velocity Estimation from a Single IMU. *Sensors Jo.*, 13(4512):1–31.

Feigl, T., Mutschler, C., and Philippsen, M. (2018a). Human compensation strategies for orientation drifts. In *IEEE Conf. on Virtual Reality and 3D User Interfaces (VR)*, pages 409–414.

Feigl, T., Mutschler, C., and Philippsen, M. (2018b). Supervised Learning for Yaw Orientation Estimation. In *Proc. Intl. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10.

Feigl, T., Nowak, T., Philippsen, M., Edelhäußer, T., and Mutschler, C. (2018c). Recurrent Neural Networks on Drifting Time-of-Flight Measurements. In *Proc. Intl. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10.

Fukushima, K. and Shouno, H. (2015). Deep convolutional network neocognitron: improved interpolating-vector. In *Intl. Joint Conf. on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Intl. Conf. on machine learning*, pages 1050–1059. PMLR.

Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *Proc. Intl. Conf. on Machine Learning*, ICML'17, page 1183–1192. JMLR.org.

Gross, J. N. and Humphreys, T. E. (2017). GNSS spoofing, jamming, and multipath interference classification using a maximum-likelihood multi-tap multipath estimator. In *Proc. Intl. Technical Meeting of The Institute of Navigation*, pages 662–670.

Hashemi, A., Thombre, S., Giorgia Ferrara, N., Zahidul, M., Bhuiyan, H., and Pattinson, M. (2019). STRIKE3-case study for standardized testing of timing-grade GNSS receivers against real-world interference threats. In *Proc. Intl. Conf. on Localization and GNSS (ICL-GNSS)*, pages 1–8.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *IEEE Proc. Conf. on computer vision and pattern recognition*, pages 7132–7141.

Joze, H. R. V., Shaban, A., Iuzzolino, M. L., and Koishida, K. (2020). MMTM: Multimodal transfer module for CNN fusion. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13289–13299.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732.

Katsaggelos, A. K., Bahaadini, S., and Molina, R. (2015). Audiovisual fusion: Challenges and new approaches. *Proc. of the IEEE*, pages 1635–1653.

Kim, Y. S., Mokaya, F., Chen, E., and Tague, P. (2012). All your jammers belong to us: Localization of wireless sensors under jamming attack. In *IEEE Intl. Conf. on Communications (ICC)*, pages 949–954.

Mehr, I. E. and Dovis, F. (2022). Detection and classification of GNSS jammers using convolutional neural networks. In *Proc. Intl. Conf. on Localization and GNSS (ICL-GNSS)*, pages 01–06.

Morales Ferre, R., de la Fuente, A., and Lohan, E. S. (2019). Jammer classification in GNSS bands via machine learning algorithms. *Sensors*, 19(4841).

Niitsoo, A., Edelhäußer, T., Eberlein, E., Hadaschik, N., and Mutschler, C. (2019). A Deep Learning Approach to Position Estimation from Channel Impulse Responses. *Sensors*, 19(5).

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567.

O'Driscoll, C., Dorner, H., Castel, B., Rubino, D., Rügamer, A., and Felber, W. (2022). Snapshot RTK with Galileo PRS: A feasibility analysis. In *Proc. Intl. Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022)*.

Pirsiavash, A., Broumandan, A., Lachapelle, G., and O'Keefe, K. (2017). Detection and classification of GNSS structural interference based on monitoring the quality of signals at the tracking level. In *ESA Intl. Collq. of Scientific and Fundamental Aspects of Galileo*, pages 1–10.

Rappaport, T. S. (1996). *Wireless Communications: Principles and Practice*. Electrical engineering. Prentice Hall.

Reuters (2022). Finland detects GPS disturbance near Russia's Kaliningrad.

Rubino, D., Rügamer, A., Lukčin, I., Taschke, S., Stahl, M., and Felber, W. (2016). Galileo PRS snapshot receiver with serverside positioning and time verification. In *Proc. Intl Conf. DGON POSNAV*.

Rügamer, A. and Kowalewski, D. (2015). Jamming and Spoofing of GNSS Signals - An Underestimated Risk?! In *Proc., FIG Working Week 2015, May 17 - 21, 2015, Sofia, Bulgaria*.

Rügamer, A., Lukcin, I., Rohmer, G., and Thielecke, J. (2013). GNSS interference detection using a compressed sensing analog to information converter approach. In *Proc. Intl. Conf. Technical Meeting of The Institute of Navigation*, pages 843–855.

Rügamer, A., Meister, D., van der Merwe, J. R., Otto, C., Stahl, M., and Felber, W. (2017). Versatile and configurable GNSS interference detection and characterization station. In *Proc. ION Pacific PNT Meeting*.

Schroeder, C. E. and Foxe, J. (2005). Multisensory contributions to low-level,'unisensory'processing. *Current Opinion in Neurobiology*.

Snoek, C. G. M., Worring, M., and Smeulders, A. W. M. (2005). Early versus late fusion in semantic video analysis. In *Proc. Intl. Conf. on Multimedia*, MULTIMEDIA '05, page 399–402, New York, NY, USA. Association for Computing Machinery.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Jo. machine learning research*, 15(1):1929–1958.

Subedar, M., Krishnan, R., Meyer, P. L., Tickoo, O., and Huang, J. (2019). Uncertainty-aware audiovisual activity recognition using deep bayesian variational inference. In *IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 6300–6309.

Swinney, C. J. and Woods, J. C. (2021). GNSS jamming classification via CNN, transfer learning & the novel concatenation of signal representations. In *Intl. Conf. on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–9.

van der Merwe, J., Cortés, I., Garzia, F., Rügamer, A., and Felber, W. (2022a). Exotic FMCW waveform mitigation with an advanced multi-parameter adaptive notch filter (MPANF). In *Proc. Intl. Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+)*.

van der Merwe, J. R., Contreras Franco, D., Jdidi, D., Feigl, T., Rügamer, A., and Felber, W. (2022b). Low-cost COTS GNSS interference detection and classification platform: Initial results. In *Intl. Conf. on Localization and GNSS (ICL-GNSS)*, pages 1–8.

van der Merwe, J. R., Garzia, F., Rügamer, A., and Felber, W. (2021). Advanced and versatile signal conditioning for GNSS receivers using the high-rate DFT-based data manipulator (HDDM). *NAVIGATION: Journal of the Institute of Navigation*, 68(4):779–797.

van der Merwe, J. R., Garzia, F., Rügamer, A., Cortés, I., and Felber, W. (2020). Adaptive notch filtering against complex interference scenarios. In *European Navigation Conf. (ENC)*, pages 1–10.

van der Merwe, J. R., Meister, D., Otto, C., Stahl, M., Rügamer, A., Etxezarreta Martinez, J., and Felber, W. (2017). GNSS interference monitoring and characterisation station. In *European Navigation Conf. (ENC)*, pages 170–178.

van der Merwe, J. R., Rügamer, A., Garzia, F., Felber, W., and Wendel, J. (2018). Evaluation of mitigation methods against COTS PPDs. In *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 920–930.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proc. Intl. Conf. on Neural Information Processing Systems (NIPS)*, page 6000–6010, Red Hook, NY, USA.

Voigt, J. M. (2021). Classification of GNSS jammers using machine learning. Masterthesis, University of Agder.

Wu, F., Luo, H., Jia, H., Zhao, F., Xiao, Y., and Gao, X. (2021). Predicting the noise covariance with a multitask learning model for Kalman filter-based GNSS/INS integrated navigation. *IEEE Trans. on Instrumentation and Measurement*, 70:1–13.

Xing, Z. and Gao, Y. (2020). A modulation classification algorithm for multipath signals based on cepstrum. *IEEE Trans. on Instrumentation and Measurement*, 69(7):4742–4752.

Yang, J. H., Kang, C. H., Kim, S. Y., and Park, C. G. (2012). Intentional GNSS interference detection and characterization algorithm using AGC and adaptive IIR notch filter. *Intl. Journal of Aeronautical and Space Sciences*, 13(4):491–498.

Zangvil Y, Z. Y. (2019). Tesla spoofing discussed, explained.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. (2021). A transformer-based framework for multivariate time series representation learning. In *Proc. ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*, page 2114–2124, New York, NY, USA.