Few-shot transfer learning for individualized braking intent detection on neuromorphic hardware

Journal of Neural Engineering

OPEN ACCESS

RECEIVED
9 July 2024

REVISED
3 January 2025

ACCEPTED FOR PUBLICATION
30 January 2025

PUBLISHED
13 February 2025

PAPER

Few-shot transfer learning for individualized braking intent detection on neuromorphic hardware

Nathan A Lutes[1], **Venkata Sriram Siddhardh Nadendla**[2,*] **and K Krishnamurthy**[1]

[1] Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, 400 W. 13th Street, Rolla, MO 65409, United States of America

[2] Department of Computer Science, Missouri University of Science and Technology, 500 W. 15th Street, Rolla, MO 65409, United States of America

[*] Author to whom any correspondence should be addressed.

E-mail: nadendla@mst.edu, nalmrb@umsystem.edu and kkrishna@mst.edu

Keywords: EEG, brain-computer interface, spiking neural networks, few-shot learning, transfer learning, neuromorphic hardware, advanced driver assist systems

Supplementary material for this article is available online

Abstract

Objective. This work explores use of a few-shot transfer learning method to train and implement a convolutional spiking neural network (CSNN) on a BrainChip Akida AKD1000 neuromorphic system-on-chip for developing individual-level, instead of traditionally used group-level, models using electroencephalographic data. The efficacy of the method is studied on an advanced driver assist system related task of predicting braking intention. *Approach.* Data are collected from participants operating an NVIDIA JetBot on a testbed simulating urban streets for three different scenarios. Participants receive a braking indicator in the form of: (1) an audio countdown in a nominal baseline, stress-free environment; (2) an audio countdown in an environment with added elements of physical fatigue and active cognitive distraction; (3) a visual cue given through stoplights in a stress-free environment. These datasets are then used to develop individual-level models from group-level models using a few-shot transfer learning method, which involves: (1) creating a group-level model by training a CNN on group-level data followed by quantization and recouping any performance loss using quantization-aware retraining; (2) converting the CNN to be compatible with Akida AKD1000 processor; and (3) training the final decision layer on individual-level data subsets to create individual-customized models using an online Akida edge-learning algorithm. *Main results.* Efficacy of the above methodology to develop individual-specific braking intention predictive models by rapidly adapting the group-level model in as few as three training epochs while achieving at least 90% accuracy, true positive rate and true negative rate is presented. Further, results show the energy-efficiency of the neuromorphic hardware through a power reduction of over 97% with only a $1.3\times$ increase in latency when using the Akida AKD1000 processor for network inference compared to an Intel Xeon central processing unit. Similar results were obtained in a subsequent ablation study using a subset of five out of 19 channels. *Significance.* Especially relevant to real-time applications, this work presents an energy-efficient, few-shot transfer learning method that is implemented on a neuromorphic processor capable of training a CSNN as new data becomes available, operating conditions change, or to customize group-level models to yield personalized models unique to each individual.

1. Introduction

Research in computing hardware, including graphics processing units (GPUs) and field programmable gate arrays, has rapidly accelerated in recent years, routinely producing faster and more powerful processing units. However, von-Neumann architecture hardware suffers from a notoriously stubborn limitation—tremendous computing power is achieved at the cost of higher energy requirements. This has led to a more focused examination of alternative hardware paradigms that can approach the capabilities of state-of-the-art central processing units (CPUs) and GPUs, but with significant energy savings. One promising approach is *neuromorphic hardware*, a new computing architecture that closely resembles biological neural networks in its construction and operation [1–3]. These types of processors use asynchronous, small bit-width encoded message passing, similar to the brain, to achieve their remarkable energy efficiency. The most recent advances in neuromorphic computing have shown competitive latency at drastically less energy consumption than the best von-Neumann style devices [4]. The major reduction in energy required for inference and online training of algorithms lends neuromorphic processors to be prime candidates for use in shared-resource environments; especially, in power-constrained applications.

Alongside the technical advancement in the design and development of brain-inspired neuromorphic hardware, the design and development of biologically influenced learning algorithms has also progressed [3, 5]. One such example is the so-called 'third' generation of neural networks—spiking neural networks (SNNs) [6–9], including deep learning versions of SNNs, such as convolutional spiking neural networks (CSNNs). Sharing in similarity to how neuromorphic hardware takes inspiration from biological neural networks, SNNs also mimic these networks in the manner in which they pass information from one layer to the next. The output of each neuron is a vector of binary values along a temporal dimension with the 'on' (i.e. 1) values being referred to as 'spikes'. Most spiking neurons contain internal dynamics that transform the weighted sum of input spikes from the previous layer into a retained memory state. When this state reaches a critical threshold, the neuron emits a spike and the state experiences a total or partial reset. This mechanism dictates the frequency of the output spikes over time with the output of the neuron being 'off' (i.e. 0) for every internal state value below the critical threshold. Although several experiments have been used to demonstrate comparable performance of SNNs with that of state-of-the-art artificial neural networks [10], their main appeal lies in their inherent ability to be directly mapped to neuromorphic hardware allowing straightforward exploitation of the processor's energy efficiency attribute. However, a limitation to exploit the full potential of SNNs has been the slow advancement in the design of on-chip training algorithms, which is being addressed with the recent availability of an edge learning algorithm integrated within the Akida AKD1000 processor [11, 12].

1.1. EEG-based advanced driver assistance

One technical field that could potentially benefit from adoption of neuromorphic hardware and SNNs is wireless brain-computer-interface (BCI) technologies [13]. Electroencephalograph (EEG), commonly associated with BCI, is increasing in popularity because of its non-invasiveness, mobility and flexibility, making it ideally suited for practical applications beyond clinical settings. One emerging sub-field of BCI is EEG-based advanced driver-assist systems (ADAS), a group of assistive technologies designed to assist with driving and/or parking decisions thereby aiding a driver in safely operating their vehicle [14]. An example of an EEG-ADAS system is described in Ford Global Technologies patent for a vehicle control system using electrical impulses from motor cortex activity in the driver's brain [15]. This system is used to help train a world champion cyclist to become a rally car driver by monitoring his EEG signals to better understand the driver's state during training [16].

Indeed, the field of intelligent ADAS, or ADAS systems imbued with computer-driven decision making and control capabilities, has seen progress in recent years, including some early work involving neuromorphic computing techniques. For example, a NM500 neuromorphic processor that contains 76 neurons was used to recognize traffic information images marked on the road (such as crosswalks) and output a speed control signal, thus creating an effective vehicle speed regulation system [17]. In another work, the NM500 processor was used for pedestrian image detection on an embedded device, and its performance and energy savings are compared against pedestrian detectors designed for CPU and GPU hardware [18]. Akida neuromorphic system-on-chip (NSoC) has also been deployed for ADAS related purposes [19, 20]; however, this did not include specific EEG-ADAS applications. In light of the growing interest in the use of neuromorphic hardware and ADAS, and noting the relative infancy of these two fields, there exists numerous opportunities for significant progress to be made.

1.2. Relevant work

Related to developing individual-level models, meta-learning (a.k.a. 'learning to learn') has been presented as an approach to optimize the learning process itself, instead of optimizing a single model for a single task [21–23]. 'One-shot learning', an extreme sub-method of meta-learning, has been used previously to train a SNN in only 'one-shot' or by only showing the network an input associated with the new task one time to produce sufficient learning [24]. This was accomplished by augmenting SNNs with adaptive neurons to increase their ability to generalize and by using two neural networks, one for a 'teacher' and another for a 'learner'; the learner being responsible for actually learning any new task, and the teacher learning an optimal learning signal to distribute to the learner such that its learning of the new task is optimized. The learning optimization, in this case, is accomplished through a gradient estimation strategy referred to as 'natural e-prop'. In another approach, using entropy theory, a gradient-based few-shot learning scheme in a recurrent SNN architecture was presented [25]. The potential energy efficiency of implementing this scheme in a neuromorphic system was discussed, but no actual results were presented. Few-shot transfer learning was previously explored through development of a 'meta-transfer learning' method which combines the training schemes of few-shot learning and transfer learning to more effectively utilize typical transfer learning models for few-shot learning [26]. However, their work did not include expansion to deep spiking neural networks or deployment to real hardware.

Although previous studies detailing neuromorphic hardware within an ADAS context is limited, studies utilizing EEG-adjacent data, such as EMG data, have demonstrated the capability and utility of neuromorphic hardware. For example, neuromorphic hardware was used in the classification of hand signals [27], and in an event-based camera sensor fusion approach [28]. The former study included a rigorous analysis of the hardware during the analysis task and obtained classification accuracy greater than 80% with hardware power consumption of only 0.05 mW. The latter study expands the EMG-only input to also include an event-based camera and compares a fully neuromorphic processing approach to a traditional machine learning technique on a portable GPU. The authors report similar discrimination performance between the two systems with the neuromorphic hardware showing increased latency but substantial improvement in energy efficiency.

In a previous study by the current authors, EEG data was used to predict prior-to-movement driver braking intention using biological systems' inspired algorithms for the first time [29]. The CSNN algorithm was chosen because of its compatibility with energy-efficient neuromorphic hardware, but the actual implementation was not considered. Further, this prior work was focused on developing a group-level model using data obtained from participants in a stress-free environment.

1.3. Contribution

The main contribution of this work is to present an energy efficient, few-shot transfer learning method for training a deep SNN using BrainChip Akida AKD1000 processor for development of individual-level cognitive models and demonstrate the efficacy of this method on an EEG-based ADAS application to detect anticipatory brain potentials at the individual level. Past EEG studies focused on group-level models aimed at generalizing to a large population. However, this scale of model can be less than ideal because individuals exhibit different responses to the same task. Therefore, it is imperative to develop and use models customized to each individual, especially in real-time applications. The use of a neuromorphic processor is also novel in the context of EEG-ADAS literature.

This work extends the previous results of [29] on predicting braking intention on two fronts: i) individual-level models are developed using a few-shot transfer learning method; and ii) two additional experiments are considered: one with elements of physical fatigue and cognitive distraction [30, 31] and the other with pseudo-realistic braking stimulation in the form of traffic light signals.

1.4. Organization

Details of the EEG experiments, EEG data preprocessing and Akida AKD1000 processor details can be found in section 2. The few-shot transfer learning methodology and its application to the EEG datasets are explained in section 3, and results are presented in section 4 and discussed in section 5. Finally, the paper concludes in section 6.

2. Experiments, datasets and hardware

2.1. Experiment design and data acquisition

Three experiments are conducted to evoke and record anticipatory signals associated with the intent to actuate a brake pedal during a driving task under different conditions. These anticipatory potentials can be observed prior to the onset of the actual braking action. The conditions of the three experiments differ primarily in the imperative cues used to initiate braking and the physical and cognitive state of the participants during data collection. In all three experiments, participants are required to continuously operate a JetBot (built using Waveshare's Jetbot AI kit and Nvidia's 4GB Jetson-Nano) around a testbed that simulates urban streets while randomly receiving external stimuli instructing when to brake. The JetBot is controlled via a Logitech G29 Driving Force racing wheel and foot pedal setup paired with a live video feed from the JetBot on-board camera cast to a computer screen. The EEG data are collected using a Neuroelectrics ENOBIO 20 EEG headset, which is configured to use the international 10–20 standard electrode setup using 19 of the 20 channels at a data

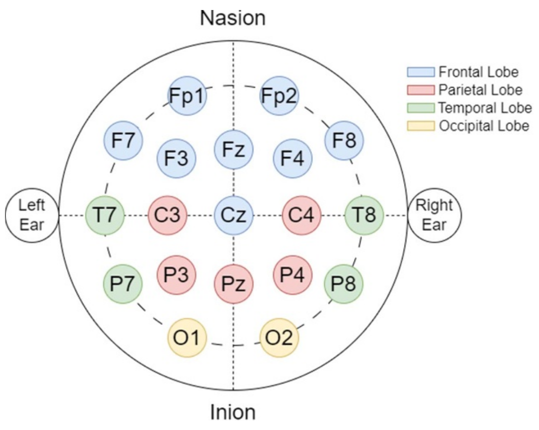

Figure 1. Schematic of the 10-20 configuration of EEG electrodes used for collecting data in this study. Grouping of the EEG electrodes correspond to frontal, parietal, temporal and occipital brain lobes [32].

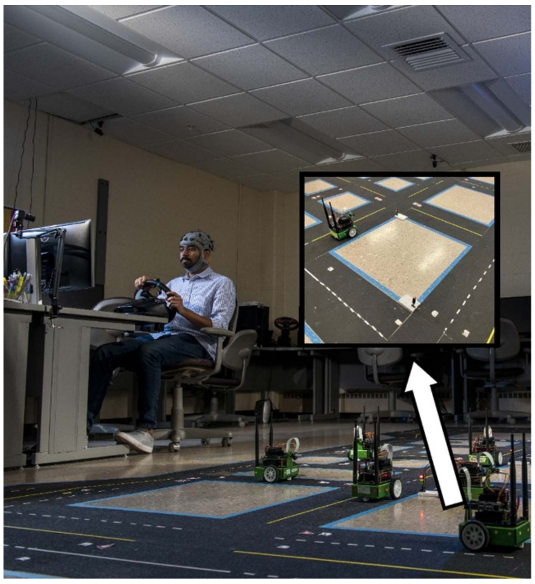

Figure 2. Photograph of the experimental setup featuring several jetbots on a portion of the testbed, a participant wearing the Enobio 20 EEG headset, the steering wheel to navigate a jetbot, and the computer monitor on which the camera feed is displayed. Insert in the photograph shows Experiment 3 set-up. Note that although multiple jetbots are shown, only the jetbot controlled by the participant is present on the testbed during the experiments in this study. The authors have confirmed that any identifiable participants in this study have given their consent for publication. Photo used with permission from Michael Pierce/Missouri S&T.

sampling rate of 500 Hz. The 20th channel serves as the common mode sense channel and is fixed to the participants right ear lobe. A schematic of the 10–20 electrode configuration and the corresponding brain lobes are shown in figure 1. Data quality is monitored via real-time indicators in the collection software. The data itself is later filtered and preprocessed for model training. Figure 2 shows a photograph of the experimental set-up.

Experiment 1: This experiment administers the imperative stop command via audio cues and without any additional stimulus to the driver. It is considered to be the nominal baseline dataset because data are collected from the participants in a stress-free environment. The participants are able to navigate the Jetbot with complete attention and receive the external braking stimuli via an automated, auditory countdown from 5 to 1 followed by an auditory 'stop' command, at which point the participants would depress the brake pedal with their foot. The participants would continue to keep the brake depressed, holding the Jetbot stationary, until they received another automated, auditory 'start' command. The period between start and stop commands containing the full verbal countdown, during which time EEG is recorded, is referred to as a 'trial' and 30 trials make a 'set'. Detailed information about Experiment 1 can be found in [29].

Experiment 2: This experiment is a modified version of Experiment 1, the difference being the addition of physical fatigue and cognitive distraction elements [30, 31]. As such, its procedure is mostly derived from that of Experiment 1, i.e. using audio cues to provide the braking imperative, but in this case participants are subjected to increased physical and cognitive stress before and during the EEG recording. The addition of these stressors creates conditions with the potential to induce changes to the EEG data streams, increasing the difficulty to predict the anticipatory potential, but in doing so creates a dataset more representative of data from a real-world scenario. The physical fatigue stems from requiring participants to traverse three flights of stairs (a total of 80 steps, up and down) prior to the start of every recording set. On the other hand, the cognitive distraction is administered via a sequence of text messages sent to the participants' phones at random intervals ranging from 15 to 30 seconds while they operated the JetBot and their EEG data is collected. Each message contains a simple question that the participant is required to respond to as quickly as possible without ignoring the normal driving task (i.e. requiring them to continue to navigate the Jetbot, and start and stop as instructed). An example of a question posed to the participants is 'How many is 10 divided by 2?'. Note that the responses to these questions are not saved and are not part of any analysis.

The necessity to constantly switch focus between navigating the jetbot and answering questions on a cell phone, which is not present in Experiment 1, requires increased mental effort by and increased mental workload on the participants. Because participants respond to the added stressors in different ways, in an attempt to enforce fairness, an emphasis is placed on consistency of stressor imposition rather than a measurement of stress sufficiency.

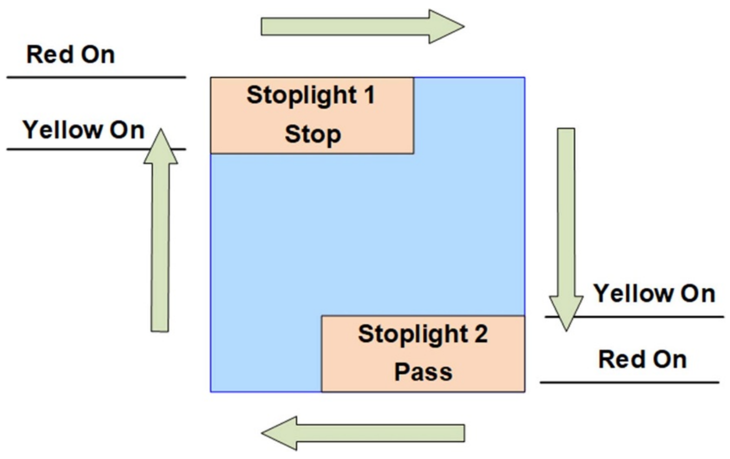

Figure 3. Schematic illustrating Experiment 3 set-up (see insert in figure 2 for a photograph of the set-up). Participants navigate the jetbot clockwise around a square track, stopping at a stoplight located at the top left corner of the track and intentionally ignoring the stoplight at the bottom right corner of the track. One revolution around the square is considered as one trial. Solid black lines with text denote the approximate locations where the noted light color would appear on the stoplight.

Experiment 3: Compared to the two previous experiments, Experiment 3 presents the participant with a different, yet more realistic, braking imperative. In this case, participants are required to sequentially stop at, or in contrast purposefully ignore, a visual cue given in the form of a stoplight turning from yellow to red. As shown in figure 3, the domain of this experiment is constrained to a square section of the testbed with two model stoplights positioned in opposite corners of the square. The Jetbot begins at one corner of the square track and the participants navigate it around the square in a clockwise manner, approaching each stoplight sequentially. The participants are instructed to stop at the first stoplight and to continue driving without braking through the second. Upon nearing each stoplight, the participants are presented with the stoplight's yellow light for 1.6 seconds before the stoplight switches to red. At the stoplight in which the participants stop, they remain stopped until the red light switches off at 2.15 seconds. The stoplights are activated manually by the experiment coordinator. A 'trial' for Experiment 3 is defined as one full cycle around the square track where the participants encounter both the stop and the pass-through lights. Markers are recorded within the EEG data for each time the stoplights turned yellow and red such that each trial has two sets of yellow followed by red light markers.

The total number of participants in the three experiments are 15, 11 and 11, respectively; however, for the analysis, only 11 participants from Experiment 1 are considered for consistency among all three experiments. Some participants participated in multiple or all experiments; however, the 11 participants chosen from Experiment 1 are not necessarily the same as the participants in the other two experiments. There are fewer participants in Experiments 2 and 3 because some of the participants from Experiment 1 were unavailable during the times when those experiments were completed, or they chose not to participate. All participants were Missouri University of Science and Technology students or professors in healthy condition with normal or corrected-to-normal vision, and with normal hearing. The three experiments received approval from the University of Missouri Institutional Review Board and all experiments were performed in accordance with relevant guidelines and regulations. Written informed consent was obtained from all participants prior to their participation. Note that because each experiment is considered separately in the statistical analysis that is explained later, it is not necessary to include the same participants in each experiment, thereby providing flexibility in recruiting participants.

2.2. Data preprocessing

In Experiment 1, each participant completes eight sets, each comprising of thirty trials, resulting in a total of 240 trials. In Experiment 2, each participant completes five sets of thirty trials each, resulting in a total of 150 trials. Only five sets are completed in Experiment 2 to accommodate participant time and availibility constraints. The experiments are carefully monitored by the authors and trials are removed if a participant fails to actuate the brake within 0.25 s of the stop command or if an incident occurs which might corrupt the data. Timing of the braking activity is enforced to minimize significant variations in the time period when anticipatory brain potentials are observed between trials. Datasets of both Experiments 1 and 2 are cleaned and preprocessed following a procedure outlined in [29], which results in each trial being broken into segments between

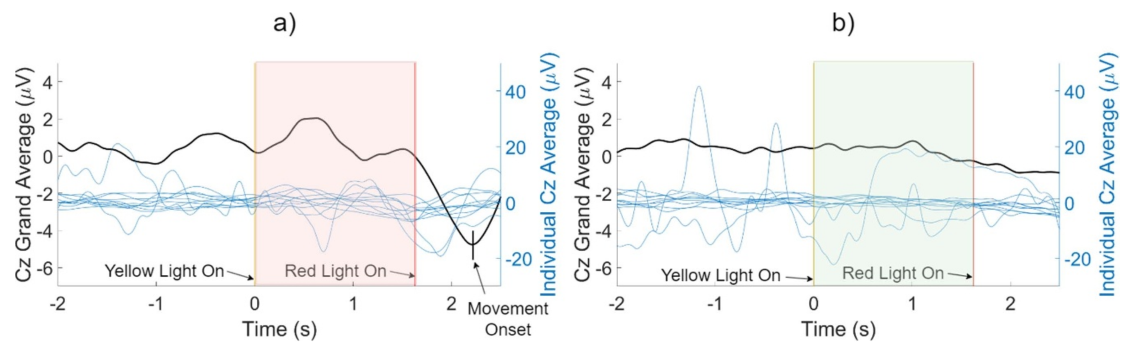

Figure 4. Experiment 3 data with markers for yellow and red traffic lights. Black line is the Cz grand average of all individuals across all trials. Blue lines are Cz averages for each of the 11 individual participants across all trials. Only data collected during the time period that is shaded in red or green is used for analysis. (a) Case where participants stopped at the red light. (b) Case where participants ignored and drove through the red light without stopping.

counts. The segments between counts 5 and 4, 4 and 3, 3 and 2, and 2 and 1 are given labels of 0 (negative). On the other hand, the segment between 1 and stop is given a label of 1 (positive). The remaining EEG data are not considered. The segments are padded to have the same overall length of 996 columns. With 19 channels, there are 18 924 data points for each segment.

For Experiment 3, each participant completes 3 sets of 30 trials (complete revolutions around the square track), resulting in 90 trials per participant. Similar to both Experiments 1 and 2, participants must actuate the brake pedal within a specified time limit, 0.4 s in this case, after red light activation or the trial is discarded. As previously noted, this automatic rejection mechanism is intended to minimize variation in the temporal location of the anticipatory brain potentials. This experiment follows the same preprocessing scheme as Experiments 1 and 2, with the exception of the segmentation strategy. Instead of five segments, only two segments are extracted from each trial: one comprising the EEG data corresponding to the traffic light in which the participant stopped and the other corresponding to the traffic light in which the participant passed through without stopping. The segments began at the yellow light marker and end at 2.15 s after the yellow light marker. The segment end time is determined by observing the location of the onset of movement in the stop data, evidenced by the peak of the dip in the Cz electrode (see figure 1) signal grand average, and ending the segment immediately prior to this peak. The Cz grand average is calculated by averaging the Cz electrode signal from all participants and across all trials. This average along with data markers and segmentation are shown in figure 4. Because the data segments for both stop and pass-without-stopping instances are the same size, 1074 data points, no additional padding is needed for this data set.

For all three experiments, the preprocessed data are converted from floating-point data to 8-bit signed integers by quantizing the data into integer bins ranging from −127 to 128. The data quantization is necessary for compatibility with BrainChip MetaTF ML framework, which comprises three Python packages, used with the Akida processor [33].

The datasets from all three experiments are split into group-level data, consisting of: (i) conglomeration of eight participants' data; and (ii) individual-level data, consisting of data from the remaining three participants. Thus, in total there are four subsets of data for each experiment's dataset: one group-level subset and three individual-level subsets. To obtain a more generalized set of results and prevent bias stemming from cherry-picking participants, the analysis is repeated ten times by randomly choosing different participants from the respective experiment datasets to compose the group and individual-level subsets each time, with each participant being in the individual-level subsets at least once.

The averages and standard deviations of the total positive and negative examples within the group and individual subsets of each respective experiment are shown in table 1. Note that the driver number in the table is used merely for counting purposes and is not meant to associate an individual to the number. The datasets from Experiment 1 and Experiment 2 suffer from a class imbalance problem of approximately four negative classes for every one positive class. This is typically solved by calculating class weights for each class and incorporating these weights into the training loss function. This approach is used for training the group model with the class weights being calculated as:

$$w_{c,i} = \frac{N_D}{2N_{c,i}} \tag{1}$$

where N_D and $N_{c,i}$ are the total number of data points in the training partition and the number of class i data points in the training partition, respectively.

Table 1. Class distributions statistics of group- and individual-level model subsets. Note that because the analysis is repeated ten times for rigor, the sizes of the subsets varied from iteration to iteration when different drivers are included in the group- and individual-level model subsets for each iteration.

Experiment Number	Subset	Driver	Total Mean (SD)	Positive Class Mean (SD)	Negative Class Mean (SD)
1	Group	Drivers 1-8	4931 (318)	986 (64)	3944 (254)
	Individual	Driver 9	462 (265)	91 (58)	371 (207)
	Individual	Driver 10	620 (129)	122 (31)	498 (99)
	Individual	Driver 11	695 (136)	141 (29)	554 (108)
2	Group	Drivers 1-8	2909 (246)	561 (56)	2347 (191)
	Individual	Driver 9	282 (130)	53 (30)	229 (101)
	Individual	Driver 10	388 (128)	76 (27)	313 (101)
	Individual	Driver 11	420 (64)	83 (15)	337 (50)
3	Group	Drivers 1-8	948 (57)	476 (29)	472 (29)
	Individual	Driver 9	128 (29)	64 (15)	64 (15)
	Individual	Driver 10	127 (31)	63 (15)	64 (15)
	Individual	Driver 11	94 (52)	48 (25)	46 (27)

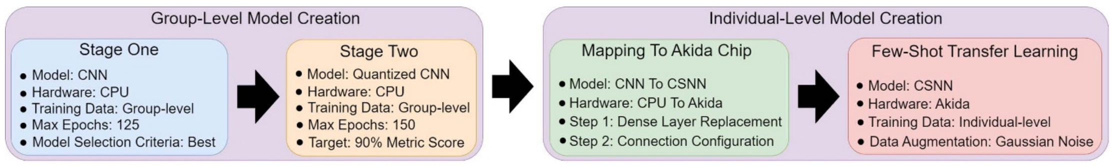
Group-Level Model Creation
Stage One
• Model: CNN
• Hardware: CPU
• Training Data: Group-level
• Max Epochs: 125
• Model Selection Criteria: Best
Stage Two
• Model: Quantized CNN
• Hardware: CPU
• Training Data: Group-level
• Max Epochs: 150
• Target: 90% Metric Score
Individual-Level Model Creation
Mapping To Akida Chip
• Model: CNN To CSNN
• Hardware: CPU To Akida
• Step 1: Dense Layer Replacement
• Step 2: Connection Configuration
Few-Shot Transfer Learning
• Model: CSNN
• Hardware: Akida
• Training Data: Individual-level
• Data Augmentation: Gaussian Noise

Figure 5. Schematic of the few-shot transfer learning method, which incorporates a three-step approach; the first two sub-steps creates a group-level model on the CPU hardware (off-chip learning), and the third step maps this model to the Akida NsoC and implements few-shot learning on the individual-level data to create individual-level models (on-chip learning).

However, this solution is not compatible with Akida edge learning, which does not utilize a typical loss function. As such, for training the individual models, the positive class samples in the individual-level subsets are duplicated three times to approximately equalize the number of positive and negative class examples. Experiment 3 dataset had approximately equal numbers of positive and negative class inputs; therefore, no additional remedy is required.

2.3. Hardware

The neuromorphic processor used in this work is BrainChip's Akida AKD1000 NSoC. This processor is capable of performing on-chip inference as well as limited on-chip training via the Akida edge learning algorithm, which enables updating the spiking output layer's binary weights online [33]. It is an event-based processor that is highly energy-efficient and does not require an external CPU to operate. The Akida AKD1000 processor contains 80 neural processing units, thereby enabling modeling of 1.2 million neurons and 10 billion synapses [34]. Because of the edge-learning capability, Akida AKD1000 processor can support various online learning methods such as incremental, continuous and one-shot learning at the processor level. As explained in the next section, for group-level model training, a PC with 32GB of DDR4 RAM and an Intel Xeon W-2123 3.60GHz processor with four cores and two threads per core for a total of eight threads is used.

3. Method

3.1. Few-shot transfer learning

The few-shot transfer learning method is designed to be compatible with the Akida AKD1000 processor. It involves the following three-step process (see figure 5), which has been successfully used to solve classification problems in edge-computing applications, including weed detection in agriculture [35] and cloud cover detection for small satellites [36].

(i) ***Group-level model creation***—The group-level model is created off-chip in two stages. A continuous convolutional neural network (CNN), modeled using Tensorflow [37], is first created and trained on group-level data consisting of data from 8 out of 11 drivers. As shown in figure 6, the CNN is a standard CNN with rectified linear unit (ReLU) activation function at the end of each convolutional layer and a linear activation function output layer with two neurons, for each of the binary classes. These activation functions are chosen based on

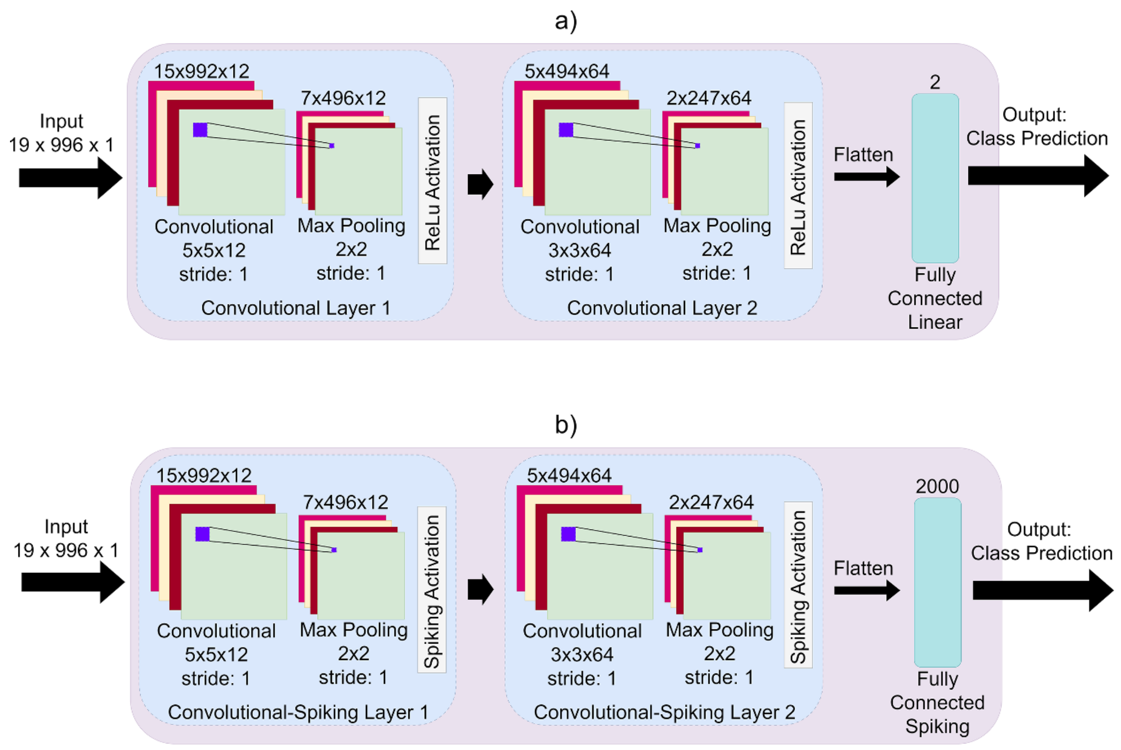

Figure 6. Schematic of the neural network architectures. (a) CNN. (b) CSNN. The CNN architecture is used during the group training on the CPU hardware. It is then converted to a CSNN and mapped to the Akida NSoC for the individual-level training. Note the large increase in output neurons for the CSNN compared to the CNN. These additional neurons are required for adequate class distribution learning during the Akida unsupervised edge learning process.

compatibility with the MetaTF ML framework conversion library. The continuous parameter CNN is trained for 125 epochs with model parameters being saved for each epoch and the best model parameters with respect to minimizing the loss being chosen to proceed forward to the second stage involving quantization. The quantization process involves converting the CNN weights to discrete 8-bit integers and the activation function outputs to single-bit integers to provide the spiking behavior, effectively converting the CNN to a CSNN using the CNN2SNN package [38]. The quantized network with discrete weights and activations is then fine tuned by further training on the same data to recoup any performance loss as a result of the quantization process. This training of the group-level model is terminated when the chosen classification performance metrics, i.e. accuracy, true positive rate (TPR) and true negative rate (TNR), are greater than 90% to prevent overfitting, which is typically less than 25 epochs, with a maximum number of training epochs of 150.

(ii) ***Mapping the CSNN to the Akida AKD1000 processor***—After the final group-level model parameters are determined, the model undergoes two additional steps before being mapped to the Akida processor. This first consists of replacing the previous dense output layer with a MetaTF ML framework fully connected output layer. This layer functions like a standard fully connected layer but contains binary weights and includes many more output neurons than the previous dense layer, as illustrated by the '2000' annotation over the final layer in figure 6. The extra neurons are included because of the Akida edge learning algorithm which requires there to be many neurons per class to better represent the class variability, similar to clustering algorithms where the cluster represents the distribution of data. Replacing the trained last layer with a new last layer allows the model to be customized for new data while relying on the feature extraction layers trained during the development of the group-level model. Such an approach is in-line with classical transfer learning theory [39].

The second step is to configure the new fully connected layer by defining the number of weights that connect the previous layer to the final layer; an important parameter for the edge learning algorithm. This value determines the number of non-zero, trainable synaptic weights for each neuron. Following BrainChip's guidelines [33], the number of connecting weights is determined by calculating the mean of the total spiking activity emanating from the second convolutional layer (the preceding layer to the final, fully connected layer) across the individual-level subset and multiplying this value by a constant set to be equal to 1.2. With the addition of the new final layer, the CSNN is mapped to the Akida processor for creating individual-level models through few-shot learning.

(iii) ***Few-shot on-chip learning for creation of individual-level models***—Individual-level data from three drivers, held out from training the group-level model, are used for creating the individual-level models. Unlike training the group-level models on the CPU hardware, individual-level models are trained on the Akida processor. For each individual driver, the edge learning algorithm is used to refine the last layer parameters, with each training task starting from a new, randomly initialized final layer. Since the individual-level subsets are considerably smaller than the group-level subset, a data augmentation approach of creating additional samples by adding white noise with zero mean and one standard deviation is employed [40, 41]. Because the analysis is repeated 10 times, a total of 30 individual-level models are trained per experiment for a grand total of 90 individual-level models for all three experiments. Four augmentations are created for each data sample thus increasing the size of each individual-level subset by five fold. Although the objective is for few-shot learning, individual model training is carried out for 25 epochs with the training metrics recorded for each epoch to study the effect of longer training on the classification results.

3.2. Neural network details

As shown in figure 6, the initial CNN model has a layout consisting of two convolutional layers ending in ReLU activation functions followed by a final fully connected layer. The first convolutional layer has 12 filters, a kernel size of 5×5 and has a max pooling layer with a pool size of 2×2. The second convolutional layer has 64 filters, a kernel size of 3x3 and a max pooling layer with a pool size of 2×2. All convolutional layers and max pooling layers used the default layer padding of ‘valid’ or no padding. The final layer is a fully connected layer with linear activation and two outputs (for the binary labels) with the loss function operating on the logits output by the network. When the CNN is converted to the CSNN and mapped to the Akida processor, the weights and activations are quantized to produce spiking behavior. The final layer is also replaced with a fully connected spiking layer with 2000 neurons (1000 for each class, which is determined by experimentation based on model performance). The default values of the edge learning algorithm hyper-parameters are used. These values are: ‘initial_plasticity’ = 1.0, ‘learning_competition’ = 0.0, ‘min_plasticity’ = 0.1 and ‘plasticity_decay’ = 0.25.

4. Results

4.1. EEG signal grand averages

The grand averages calculated by averaging the EEG signals from all 11 participants across all trials, individual averages calculated by averaging the EEG signals for each of the 11 individuals across all trials, and the associated EEG scalp plots are shown in figures 7–9 for Experiments 1, 2 and 3, respectively. In all three figures, the EEG signals are staggered for better visualization and the scalp plots have the same color bar range. Several interesting observations can be made.

A sharp negative variation can be observed in the Cz grand average in all three figures. In Experiments 1 and 2, this begins around the 2 marker and continues until about 1 second before the stop marker. In Experiment 3, this can be observed in figure 9(a), at the red light activation marker and continues for about 0.6 seconds after the marker. However, this is noticeably absent in figure 9(b), which shows the EEG grand averages of the instances where the participant did not stop. This negative variation, referred to as a contingent negative variation (CNV), is indicative of movement intent [42–44] and can be seen in several areas of the brain across all three experiments, but most notably occurs in the centro-medial region. The primarily centralized location of the phenomenon is the motivation for using the Cz grand average as the basis for the segmentation boundaries as illustrated in figure 4. Indeed, an ablation study performed in [29] confirmed that only a few channels located in the centro-medial region are necessary for satisfactory prediction of the CNV signal.

The effects of physical fatigue and cognitive distraction can be observed by comparing Experiments 1 and 2 scalp plots (figures 7(b) and 8(b)). In general, the EEG grand averages for Experiment 2 appear to have much more variability than Experiment 1 indicating dynamic brain activity. Experiment 2 shows activity in the O1 and O2 electrodes over the occipital lobe and P4 over the parietal lobe between the 4 and 3 markers. While it is difficult to pinpoint the exact cause of the activation because of volume conduction, increased activity in these electrodes indicate visual and cognitive processing, respectively [45, 46]. Experiment 2 also shows a period of high positive activity across many of the brain regions between the 3 and 2 markers. Between the 2 and 1 marker, the F7 and F8 electrodes over the prefrontal cortex and the Fp1, Fp2, F7, F3, Fz, F4 electrodes over the frontal lobe are active indicating cognitive and executive functions, respectively. The scalp plot before the stop command in figure 8(b) shows a small region on

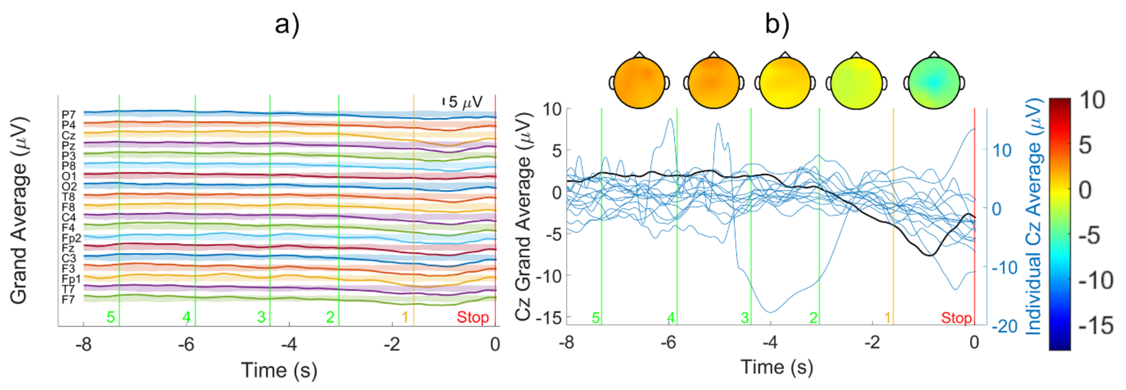

Figure 7. Experiment 1 preprocessed EEG signals. (a) Channel potentials with associated countdown and 'Stop' command markers and scale. Solid lines are the grand averages of all individuals across all trials. The shaded regions represent the range from minimum to maximum values across all trials (drawn at 1/40 scale of the grand average). (b) Black line is the Cz grand average signal of all individuals across all trials with scalp maps representing the grand average at the midpoint between two neighboring markers. Blue lines are Cz averages for each of the 11 individual participants across all trials. Color bar on the right displays the Cz channel potential in μV. Adapted from [29]. CC BY 4.0.

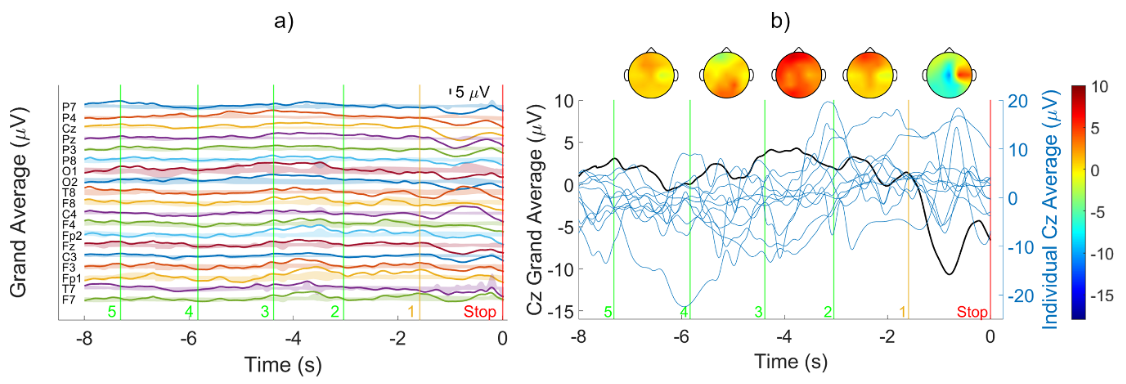

Figure 8. Experiment 2 preprocessed EEG signals. (a) Channel potentials with associated countdown and 'Stop' command markers and scale. Solid lines are the grand averages of all individuals across all trials. The shaded regions represent the range from minimum to maximum values across all trials (drawn at 1/300 scale.) (b) Black line is the Cz grand average signal of all individuals across all trials with scalp maps representing the grand average at the midpoint between two neighboring markers. Blue lines are Cz averages for each of the 11 individual participants across all trials. Color bar on the right displaying the Cz channel potential in μV. Grey lines represent Cz averages for each individual participant across all trials.

the right lateral part of the brain that has the opposite effect of the CNV, instead increasing to large positive values. This is because of the C4 and T8 electrodes as can be seen in figure 8(a). Increased activity in the C4 electrode over the central region arises from executing hand movements. Participants are instructed to rapidly respond to the cognitive distraction questions on their cell phones, thus requiring them to quickly remove their hands from the wheel, respond and then re-engage the steering wheel. The T8 electrode over the right temporal lobe shows increased activity because of auditory processing and memory recall that occurs while responding to the cognitive distraction questions. The increased brain activity levels and steeper decline in the CNV is the effect of physical fatigue and cognitive distraction throughout the countdown leading to a much shorter reaction time for participants in Experiment 2 who are less prepared for the stop command than those in Experiment 1. Interestingly, the magnitude in the Cz grand average CNV signal is much larger for Experiment 2 than for Experiment 1. This is also because of the reduced reaction time for distracted participants causing the brain to make rapid preparations for the upcoming braking action. Although it is not possible to disentangle the effects of physical fatigue and cognitive distraction, it is posit that the higher brain activity is the result of cognitive distraction.

In figure 9, besides the CNV in figure 9(a), notable differences between figures 9(a) and (b) occur one second prior to the yellow light marker and at the red light marker. Prior to the yellow light marker in figure 9(b), the Fp2 and F4 electrodes over the frontal lobe and the O1 and O2 electrodes over the occipital lobe have higher activity, indicating restraint on seeing the stoplight as per the experiment instructions. At the red light marker in figure 9(a), there is higher activity on the left and front sides of the

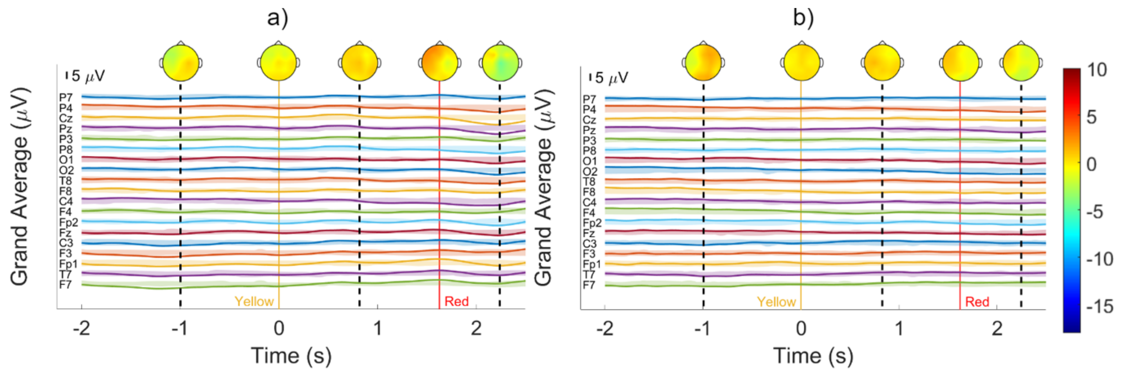

Figure 9. Experiment 3 preprocessed EEG signals. (a) Channel potentials for stopping at red light with markers for yellow and red lights, and scale. (b) Channel potentials for driving through without stopping at the red light with markers for yellow and red lights, and scale. For both (a) and (b), Solid lines are the grand averages of all individuals across all trials. The shaded regions represent the range from minimum to maximum values across all trials (drawn at 1/40 scale of the grand average). Also, scalp maps represent the grand average at the marker location or at the temporal location denoted by the dashed lines. The color bar on the right displays the channel potential in μV.

brain compared to the instance where participants purposefully neglect to stop. Higher activity is seen in the F7 electrode over the prefrontal cortex, F3, Fp1, Fp2 and Fz over the frontal lobe, T7 over the temporal lobe and C3 over the central region indicating cognitive, motor, memory and executive task activities associated with increased attention, recollection of the experiment instructions to stop and action planning.

4.2. Analysis results: performance

Results are presented for two different studies to evaluate the efficacy of the proposed methodology: i) a baseline all-channel study (ACS) utilizing all 19 channels; and ii) a five-channel ablation study (FCAS) using only the Cz, Pz, C3, C4 and Fz centro-medial channels as done in [29]. To implement the ablation study, unused data channels are replaced with null values to allow usage of the same architecture configuration. To determine if the median values obtained from the ACS and FCAS are statistically different within each of the three experiment, two-sided Wilcoxon rank sum tests are conducted. Figures 10–12 show results for the three experiments for the first two sub-steps when group-level models are created and at select intervals during the creation of the individual-level models (see figure 5). Each figure shows notched box plots representing the distribution of median values for each performance metric, i.e. classification accuracy, TPR, TNR and area under the receiver operating characteristic (ROC) curve (AUC) for both studies when the analysis is repeated 10 times. The p-values obtained are also shown if there is no overlap in the notches of two box plots in each subplot, indicating that the ACS and FCAS are statistically different at the 5% significance level. Because the method presented in this study is a few-shot learning method, only the training metrics are reported, consistent with previous literature [24].

Figure 13 shows the average of the metric scores, computed across all 90 individual models (three individual models $\times$ three experiments $\times$ 10 folds per experiment), as transfer learning progresses across the training epochs. The bottom x-axis displays the training epoch number and the top x-axis displays the time elapsed since transfer learning started. Note that the training times differ between experiments and individual models, and are generally dependent on the number of training samples included in each epoch. As a result, the per-epoch training times shown in the figure are the grand averages of the per-epoch times across all 90 individual models, similar to the metric scores. The metric score averages are shown as continuous lines and the shaded regions correspond to one standard deviation. Note that the values reported at epoch '0' in the figure are the averages and standard deviations of the performance metric scores obtained from a preliminary inference using individual models with initial, untrained final layers before any edge learning occurs. The results shown in figure 13 are obtained using the zero default value of the 'learning_competition' parameter in the edge learning algorithm. Results similar to figure 13 are obtained with other values of 'learning_competition'. These results are shown in figures S1–S3 in the supplementary material.

4.3. Analysis results: power and latency

Table 2 show the means and standard deviations of the power consumed and inference latency for all individual-level subsets across all 10 tests, for both the ACS and FCAS, subdivided by experiment. For this evaluation, each of the three held out individual-level subsets for a given test are passed as a single batch to the quantized group-model CNN on the CPU and the MetaTF ML framework based CSNN on the Akida processor and the time required and power consumed for inference are recorded for comparison. At this

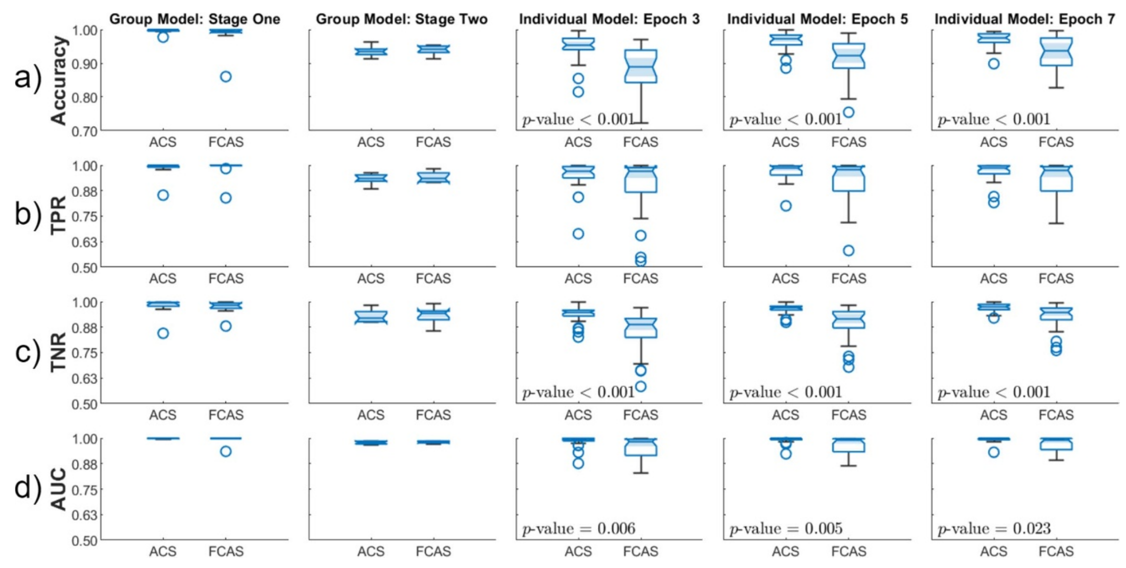

Figure 10. Experiment 1 training results for group- and individual-level models. (a) Accuracy. (b) True positive rate (TPR). (c) True negative rate (TNR). (d) Area under the receiver operating characteristic (ROC) curve (AUC). On each notched box, the central mark shows the median, bottom and top edges show the 25th and 75th percentiles, respectively, and the whiskers extend to the most extreme data points that are not considered as outliers, which are defined as 1.5× the interquartile range away from the top or bottom of the box. Each subplot shows a comparison of the results for all-channel study (ACS) and five-channel ablation study (FCAS) when the analysis is repeated 10 times. Non-overlapping shaded notched regions in a subplot implies the median values are statistically different at the 5% level for the particular subplot case. In such cases, the p-value is shown in the subplot.

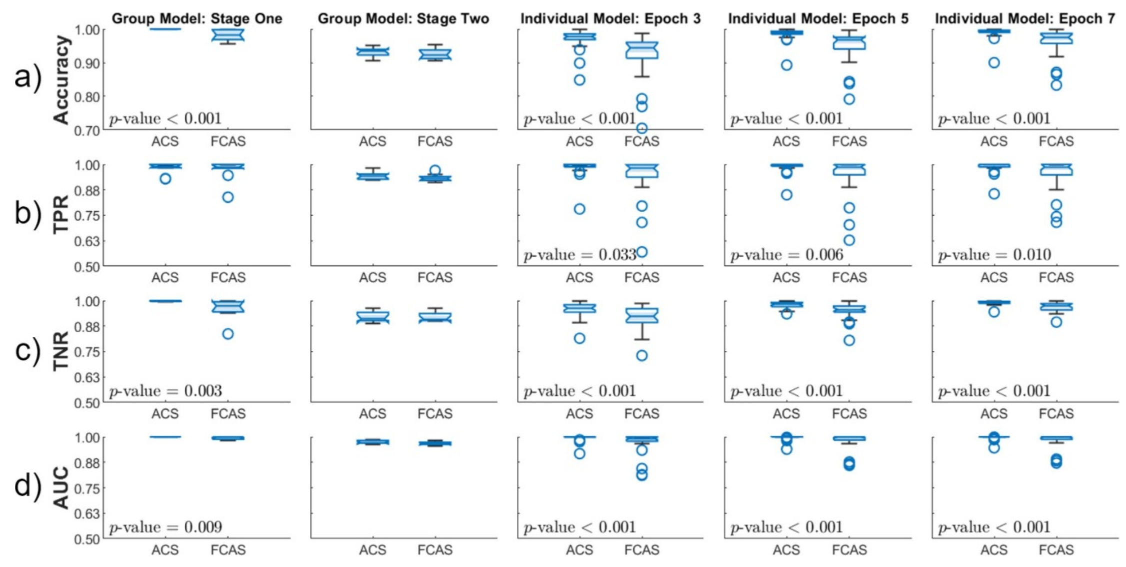

Figure 11. Experiment 2 training results for group- and individual-level models. (a) Accuracy. (b) True positive rate (TPR). (c) True negative rate (TNR). (d) Area under the receiver operating characteristic (ROC) curve (AUC). On each notched box, the central mark shows the median, bottom and top edges show the 25th and 75th percentiles, respectively, and the whiskers extend to the most extreme data points that are not considered as outliers, which are defined as 1.5× the interquartile range away from the top or bottom of the box. Each subplot shows a comparison of the results for all-channel study (ACS) and five-channel ablation study (FCAS) when the analysis is repeated 10 times. Non-overlapping shaded notched regions in a subplot implies the median values are statistically different at the 5% level for the particular subplot case. In such cases, the p-value is shown in the subplot.

time, the MetaTF API only supports power consumption at an inference level and does not support measurement of the network training-level power consumption. The CPU power consumed is calculated by recording the power required for inference and then subtracting the power consumed when code execution is suspended. This is intended to provide a fairer comparison with the Akida processor by removing any power consumed by background processes such as the operating system from the inference power measurement. The CPU power was measured using the *PyJoules* Python package. This package monitors the CPU energy consumed while running Python code by using the 'Running Average Power Limit'

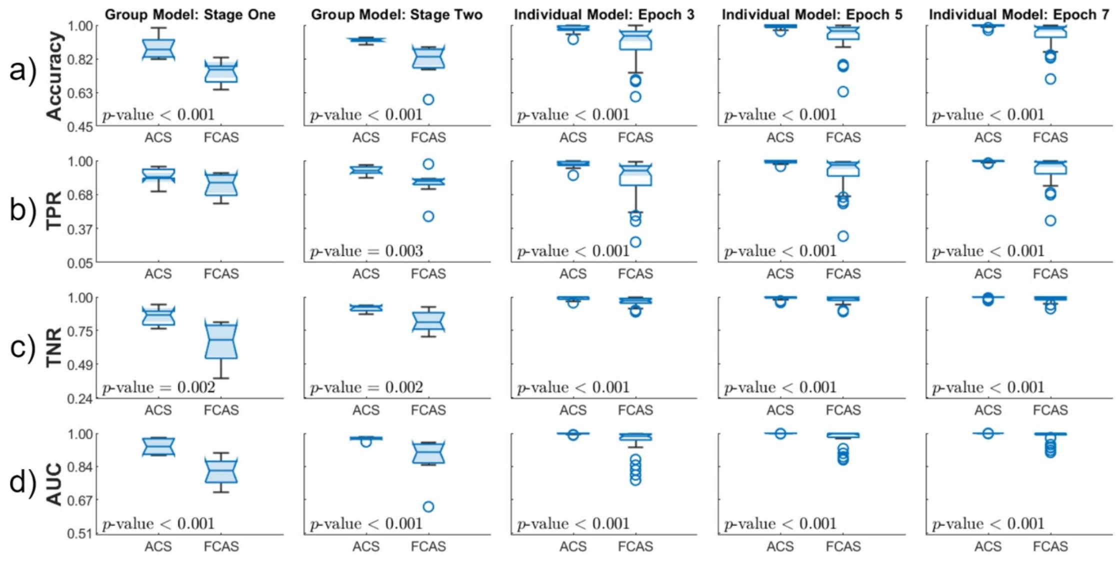

Figure 12. Experiment 3 training results for group- and individual-level models. (a) Accuracy. (b) True positive rate (TPR). (c) True negative rate (TNR). (d) Area under the receiver operating characteristic (ROC) curve (AUC). On each notched box, the central mark shows the median, bottom and top edges show the 25th and 75th percentiles, respectively, and the whiskers extend to the most extreme data points that are not considered as outliers, which are defined as 1.5× the interquartile range away from the top or bottom of the box. Each subplot shows a comparison of the results for all-channel study (ACS) and five-channel ablation study (FCAS) when the analysis is repeated 10 times. Non-overlapping shaded notched regions in a subplot implies the median values are statistically different at the 5% level for the particular subplot case. In such cases, the p-value is shown in the subplot.

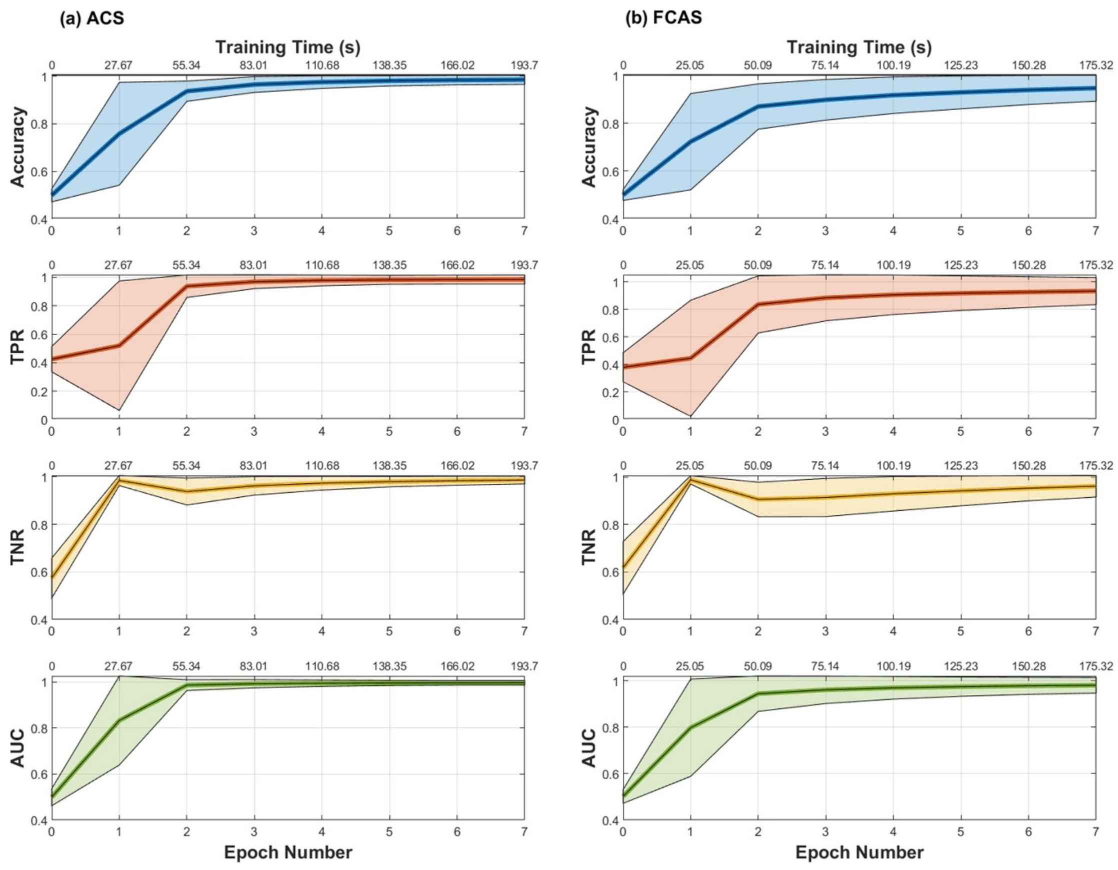

Figure 13. Performance of individual-level models as a function of transfer learning shots (epochs) for learning competition = 0 (default). (a) Average metric score with one standard deviation shaded region for accuracy, true positive rate (TPR), true negative rate (TNR) and area under the receiver operating characteristic (ROC) curve (AUC), respectively, for all-channel study (ACS). (b) Average metric score with one standard deviation shaded region for accuracy, TPR, TNR and AUC, respectively, for Five-channel Study (FCAS).

Table 2. Power consumption and inference time comparison between central processing unit (CPU) and Akida NSoC hardware for individual-level models. CNN—convolutional neural network, CSNN—convolutional spiking neural network, SD—standard deviation, ACS—all-channel study, FCAS—five-channel ablation study, PR—percent reduction, LR—latency ratio.

			Power (W)				Inference Time (s)			
			ACS		FCAS		ACS		FCAS	
Experiment	Model	Hardware	Mean (SD)	PR	Mean (SD)	PR	Mean (SD)	LR	Mean (SD)	LR
1	CNN	CPU	43.31 (2.35)	—	42.06 (3.79)	—	4.49 (3.30)	—	5.63 (2.71)	—
	CSNN	Akida	0.875 (.001)	97.98	0.874 (0.0001)	97.92	5.56 (4.15)	1.24	5.36 (2.72)	0.95
2	CNN	CPU	43.1 (0.65)	—	41.8 (1.1)	—	3.67 (1.37)	—	3.17 (1.97)	—
	CSNN	Akida	0.872 (0.001)	97.97	0.875 (0.001)	97.91	4.80 (1.94)	1.30	2.76 (1.76)	0.87
3	CNN	CPU	39.62 (5.35)	—	37.8 (2.32)	—	0.72 (0.29)	—	0.87 (0.34)	—
	CSNN	Akida	0.874 (0.0002)	97.79	0.874 (0.0001)	97.69	0.94 (0.36)	1.30	0.72 (0.26)	0.83

(RAPL) technology that is standard on Intel CPUs. The power consumed is computed by dividing the energy usage reported for a given computation (e.g. Python script) by the duration of the computation. The percent reduction (PR) in the power consumed and the latency ratio (LR) are calculated as:

$$\mathrm{PR} = 100 \times \left(\frac{\mathrm{CPU}_{\mathrm{power}} - \mathrm{Akida}_{\mathrm{power}}}{\mathrm{CPU}_{\mathrm{power}}} \right) \quad (2)$$

$$\mathrm{LR} = \frac{\mathrm{Akida}_{\mathrm{inference\ time}}}{\mathrm{CPU}_{\mathrm{inference\ time}}}. \quad (3)$$

Note that a latency ratio greater than 1 indicates an increased latency for Akida.

Due to the different amounts of data collected for each experiment and the data quality checks administered as previously discussed, the total number of data points differ for each individual-level subset (see table 1). Therefore, the inference power consumed and inference latency for each individual-level subset are different. Because of these differences, the data are expressed as means and standard deviations.

5. Discussion

5.1. Performance

Several observations can be made considering the results for Experiment 1 shown in figure 10. For both the ACS and FCAS, stage one group-level training results for accuracy, TPR and TNR have tight distributions with most values above 90%. The AUC score distributions indicate consistently near-perfect classifiers with a single outlier in the FCAS. Stage two group-model training shows greater distributions, but this is expected because of the stopping criteria aimed at reducing over-fitting to the quantized group model. Three observations could be made examining the individual-level model box plots. First, the majority of values in the ACS are at or above 90% score for all metrics, even at only 3 shots (epochs) with the median and 1st and 3rd quartile values improving with more shots. The FCAS shows somewhat larger distributions in general but most metric values still seem to be above 80%. Second, except for TPR, ACS and FCAS results are significantly different as indicated by the non-overlapping notched regions in the box plots using a 5% significance level. This is in contrast to the group-level model. Therefore, for better results, FCAS could be used for group-level models with reduced computational burden, whereas ACS would be necessary for individual-level models. Finally, as expected, all training results improve when training is conducted over more epochs.

Results for Experiment 2 are presented in figure 11. The group model training results are similar to those obtained for Experiment 1 albeit with greater variability and slightly lower median and 1st and 3rd quartile values. This is expected because of the increased brain activity due to the physical and cognitive distraction noted in figure 8. The stage one results also show statistically significant differences between the ACS and FCAS results but these differences are resolved in stage two, indicating that the final outcome of the group model training is still statistically similar. The trend of results for training individual-level models are also on par with Experiment 1 results with perhaps a few more outliers, indicating robustness of the Akida edge learning algorithm even in the presence of physical and cognitive distraction. The AUC scores indicate robust classifiers at each stage with generally tight distributions with some outliers. This also lends credence to the learning strategy robustness even when the results for ACS and FCAS are statistically different.

Figure 12 shows results for Experiment 3 where participants stop at the traffic light or intentionally ignore the stoplight. Except for the TPR at stage one of creating group-level models, all Experiment 3 results show a significant difference between the ACS and FCAS. This is not surprising and to be expected because Experiment 3 includes a visual stimulus in monitoring the stoplight for a braking imperative. Visual information is processed in the occipital lobe, where O1 and O2 electrodes are placed; however, these channels are not included in the ablation study that focused on the centro-medial region.

Thus, critical information from these channels are missing in the ablation study data. It should also be noted that the group model performance for experiment three is, in general, worse than the other two experiments. This could be due to the relatively smaller dataset obtained for experiment 3. It is interesting to note, however, that the ACS individual models still manage to achieve good performance, producing tight spreads near 90th percentile and above. The AUC scores for the ACS are still very good and show very tight spreads during the individual model training stages. The FCAS individual models also typically have good AUC scores although with considerably more outliers.

It is apparent from the results showed in figure 13 that, in general, the individual models deployed on the Akida processor adapt to each individual driver's data quickly, only requiring two epochs on average to achieve at least 90% metric score for accuracy, TPR and TNR when training with data from all channels and requiring only three epochs on average when training with five channels. Although the standard deviations of all metrics besides TNR are higher after the first training epoch, they reduce significantly after the second training epoch and beyond in both ACS and FCAS. This demonstrates an overarching consistency in the training results in just a few shots.

5.2. Power and latency

Across all three experiments, both ACS and FCAS show a substantial reduction in the inference power consumption using the Akida processor compared to the CPU hardware, achieving percent reductions greater than 97%. Interestingly, the Akida processor has a slightly larger (about 1.3×) latency compared to the CPU hardware when all channels are used, but a smaller (about 0.9×) latency only five channels are used in FCAS. The latency results of the ACS are not altogether surprising as the Intel Xeon CPU is a very powerful CPU capable of making a large number of computations in a short amount of time; however, the superior latency of the Akida processor in the FCAS is indicative of the competitiveness of neuromorphic hardware with cutting edge von-Neumann hardware albeit at a much lower power requirement. This corroborates with previous studies comparing neuromorphic and von-Neumann paradigms [28, 36]. The Akida processor achieves this through its sparse, asynchronous computing regime.

As the quantity of data collected from burgeoning low-cost sensors and task complexity increases, power and latency may become bottlenecks for practical applications or large-scale problems. In edge computing and mobile applications, low-power computing will be essential because of the limited capacity of energy storage devices used to power these systems. Successful deployment of machine learning algorithms requires low latency for inference, let alone situations where machine learning models need to be adapted on-line to improve their performance because of, for example, changes in the environment.

An emerging application area for neuromorphic computing is closed-loop control systems. In these systems, it is well known that time delay can limit the control bandwidth and affect the control system stability. Therefore, low latency is desired to close the loop between sensing and actuation. Moreover, the performance of a discrete-time system controller can better approximate a continuous-time system with a shorter sampling period. Further, neural adaptive control methods will be necessary to tune the nominal control system because of, for example, modeling errors. This imposes stricter constraints on the latency requirement because of the additional computations that need to be performed in real time.

5.3. Broader implication and comparison with other neuromorphic hardware

Although the few-shot transfer learning method shown in figure 5 is to create individual-level models, the general methodology of off-chip learning and on-chip few-shot learning can be used to build neuromorphic learning machines. Because a benchmark problem currently does not exist, a direct, quantitative comparison of the proposed methodology with other neuromorphic implementations is not possible. Nonetheless, a few select results from the literature are presented to better understand the possibilities.

The approach presented here is similar to a couple of other studies in which the same Akida NSoC is used [35, 36]. A predictive model capable of identifying wild radish weeds at four different growth stages achieved a classification accuracy of 99.73%, outperforming other popular CNN architectures [35]. However, no comparison is made to traditional von Neumann hardware to demonstrate the energy efficiency of using an Akida NSoC. A CNN is trained and later converted to an SNN at the chip level in a cloud cover detection application [36]. This approach achieved a 95.46% pixel-wise accuracy and 79.37 mean intersection-over-union, and when compared to the pre-converted CNN deployed on a Jetson TX2, the SNN on a Akida NSoc consumed 35× less energy.

A deep CNN is adapted to perform classification tasks on an IBM TrueNorth neuromorphic hardware [47]. State-of-the-art inference accuracy results are obtained on eight standard datasets while running between 1200 and 2600 frames/s and using between 25 and 275 mW. Intel's Loihi is another neuromorphic chip that has been used in a variety of different applications [4]. In one example, an adaptive neural network is used to augment an existing force controller to control a Kinova Jaco robot that is holding an unknown weight [48]. The adaptive controller is tuned on-line to account for unexpected forces affecting movement. The adaptive controller implemented on Intel's Loihi demonstrated a 2.45× improvement in accuracy compared to a standard proportional-integral-derivative controller. The lower latency resulted in a 1.49× and 1.57× improvement over a CPU and GPU implementation, respectively. Further, the CPU and GPU required 4.6× and 43.2×, respectively, more power than Loihi. In another example, an event-based vision algorithm implemented as a SNN on a Kapoho Bay device with two Loihi chips is used in a drone controller [49]. A computational timestep of 50 μs on Loihi allowed an update rate of 20 kHz compared to 1 kHz update rate on a CPU. Loihi's lower latency enables faster control action, which enabled the SNN controller track horizon rotating at higher speeds with a lower error.

5.4. Limitations

While the performance of the method on the data presented shows promise, this is not achieved without limitations. First, the mechanization of the method is intrinsically coupled with BrainChip's proprietary Akida NSoC and MetaTF API, including their proprietary edge learning algorithm. Thus, adaptations of the method to other neuromorphic hardware may not be straightforward and could require modification to the methodology or it may provide less than satisfactory results. Second, the method requires prior training and maintenance of a group-level model before individual-level models can be created. This, in turn, carries over some of the drawbacks of previous, purely group-level approaches in that a large dataset must be acquired, and time and resources must be spent creating the group-level model. Lastly, the method, as it is presented in this work, is not truly 'online'. For actual deployment, individual-level data must be collected for every driver using a contrived data acquisition scheme and the model trained on this data before implementing and continuing to train during actual driving scenarios. This data acquisition scheme must be defined appropriately such that enough examples of different braking and normal driving conditions are present to adequately represent the driver's individual response. Nonetheless, this work presents a novel application of a methodology able to create individual-level models at the hardware level for EEG-ADAS related tasks.

6. Conclusion

The results show that the methodology presented was effective to develop individual-level models deployed on a state-of-the-art neuromorphic processor with predictive abilities for ADAS relevant tasks, specifically braking intent detection. This study explored a novel application of deep SNNs to the field of ADAS using a neuromorphic processor by creating and validating individual-level braking intent classification models with data from three experiments involving pseudo-realistic conditions. These conditions included cognitive atrophy through physical fatigue and real-time distraction and providing braking imperatives via commonly encountered visual stimulus of traffic lights. The method presented demonstrates that individual-level models could be quickly created with a small amount of data, achieving greater than 90% scores across three classification performance metrics, accuracy, TPR and TNR, and high AUC scores in a few shots on average for both the ACS and FCAS. This demonstrated the efficacy of the method for different participants operating under non-ideal conditions and using realistic driving cues and further suggests that a reduced data acquisition scheme might be feasible in the field. Furthermore, the applicability to energy-constrained systems was demonstrated through comparison of the inference power consumed with a very powerful CPU in which the Akida processor offered power savings of 97% or greater. The Akida processor was also shown to be competitive in inference latency compared to the CPU. Future work could include implementation of the method presented on a larger number of participants, other neuromorphic hardware, different driving scenarios, and in real-world scenarios where individual-level models are created by refining previously developed group-level models in real time.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

ORCID iDs

Nathan A Lutes https://orcid.org/0000-0003-0512-5809
Venkata Sriram Siddhardh Nadendla https://orcid.org/0000-0002-1289-7922
K Krishnamurthy https://orcid.org/0000-0002-0219-6153

References

[1] Schuman C D, Potok T E, Patton R M, Birdwell J D, Dean M E, Rose G S and Plank J S 2017 A survey of neuromorphic computing and neural networks in hardware (arXiv:1705.06963)
[2] Clark K and Wu Y 2023 Survey of neuromorphic computing: A data science perspective *2023 IEEE 3rd Int. Conf. on Computer Communication and Artificial Intelligence (CCAI)*
[3] Shrestha A, Fang H, Mei Z, Rider D P, Wu Q and Qiu Q 2022 A survey on neuromorphic computing: Models and hardware *IEEE Circuits Syst. Mag.* **22** 6–35
[4] Davies M, Wild A, Orchard G, Sandamirskaya Y, Guerra G A F, Joshi P, Plank P and Risbud S R 2021 Advancing

neuromorphic computing with loihi: A survey of results and outlook *Proc. IEEE* **109** 911–34

[5] Schuman C D, Kulkarni S R, Parsa M, Mitchell J P, Date P and Kay B 2022 Opportunities for neuromorphic computing algorithms and applications *Nat. Comput. Sci.* **2** 10–19

[6] Tan C, Šarlija M and Kasabov N 2020 Spiking neural networks: Background, recent development and the neucube architecture *Neural Process. Lett.* **52** 1675–701

[7] Dora S and Kasabov N 2021 Spiking neural networks for computational intelligence: An overview *Big Data Cogn. Comput.* **5** 67

[8] Tavanaei A, Ghodrati M, Kheradpisheh S R, Masquelier T and Maida A 2019 Deep learning in spiking neural networks *Neural Netw.* **111** 47–63

[9] Maass W 1997 Networks of spiking neurons: The third generation of neural network models *Neural Netw.* **10** 1659–71

[10] Christensen D V *et al* 2022 2022 roadmap on neuromorphic computing and engineering *Neuromorph. Comput. Eng.* **2** 022501

[11] Vanarse A, Osseiran A, Rassau A and van der Made P 2019 A hardware-deployable neuromorphic solution for encoding and classification of electronic nose data *Sensors* **19** 4831

[12] Ferré P, Mamalet F and Thorpe S J 2018 Unsupervised feature learning with winner-takes-all based STDP *Front. Comput. Neurosci.* **12** 24

[13] Douibi K, Bars S, Lemontey A, Nag L, Balp R and Breda G 2021 Toward EEG-based BCI applications for Industry 4.0: Challenges and possible applications *Front. Hum. Neurosci.* **15** 705064

[14] Swief A and El-Habrouk M 2018 A survey of automotive driving assistance systems technologies *2018 Int. Conf. on Artificial Intelligence and Data Processing (IDAP)* pp 1–12

[15] Hassani A, Ravindran A and Nagasamy V 2023 Vehicle computer command system with a brain machine interface *US Patent* 11,780,445

[16] Barker P 2019 How Gee Atherton is using brain tech to improve his rallying skills (available at: www.redbull.com/us-en/gee-atherton-rallying-ford-eeg-helmet)

[17] Kim J 2020 New neuromorphic AI NM500 and its ADAS application *Aeta 2018—Recent Advances in Electrical Engineering and Related Sciences: Theory and Application* ed I Zelinka, P Brandstetter, D T Trong, D V Hoang and S B Kim (Springer International Publishing) pp 3–12

[18] Kang M, Lee Y and Park M 2020 Energy efficiency of machine learning in embedded systems using neuromorphic hardware *Electronics* **9** 1069

[19] Brainchip Inc. Designing smarter, safer cars with essential AI (available at: https://brainchip.com/designing-smarter-and-safer-cars-with-essential-ai/)

[20] Ward-Foxton S 2022 Mercedes applies neuromorphic computing in EV concept car EE Times (available at: www.eetimes.com/mercedes-applies-neuromorphic-computing-in-ev-concept-car/)

[21] Thrun S and Pratt L 1998 *Learning to Learn* (Springer Science and Business Media, LLC)

[22] Andrychowicz M, Denil M, Gomez S, Hoffman M W, Pfau D, Schaul T and de Freitas N 2016 Learning to learn by gradient descent by gradient descent (arXiv:1606.04474v1)

[23] Hochreiter S, Younger A S and Conwell P R 2001 Learning to learn using gradient descent *Artificial Neural Networks—ICANN 2001* (https://doi.org/10.1007/3-540-44668-0_13) pp 87–94

[24] Scherr F, Stöckl C and Maass W 2020 One-shot learning with spiking neural networks (available at: https://doi.org/10.1101/2020.06.17.156513)

[25] Yang S, Linares-Barranco B and Chen B 2022 Heterogeneous ensemble-based spike-driven few-shot online learning *Front. Neurosci.* **16** 850932

[26] Sun Q, Liu Y, Chua T S and Schiele B 2019 Meta-transfer learning for few-shot learning (arXiv:1812.02391)

[27] Donati E, Payvand M, Risi N, Krause R and Indiveri G 2019 Discrimination of emg signals using a neuromorphic implementation of a spiking neural network *IEEE Trans. Biomed. Circuits Syst.* **13** 795–803

[28] Ceolini E, Frenkel C, Shrestha S B, Taverni G, Khacef L, Payvand M and Donati E 2020 Hand-gesture recognition based on emg and event-based camera sensor fusion: A benchmark in neuromorphic computing *Front. Neurosci.* **14** 637

[29] Lutes N, Nadendla V S S and Krishnamurthy K 2024 Convolutional spiking neural networks for intent detection based on anticipatory brain potentials using electroencephalogram *Sci. Rep.* **14** 8850

[30] Xu R, Zhang C, He F, Zhao X, Qi H, Zhou P, Zhang L and Ming D 2018 How physical activities affect mental fatigue based on eeg energy, connectivity and complexity *Front. Neurol.* **9** 915

[31] Almahasneh H, Chooi W T, Kamel N and Malik A S 2014 Deep in thought while driving: An eeg study on drivers' cognitive distraction *Trans. Res.* F **26** 218–26

[32] Rojas G M, Alvarez C, Montoya C E, de la Iglesia-Vayá M, Cisternas J E and Gálvez M 2018 Study of resting-state functional connectivity networks using eeg electrodes position as seed *Front. Neurosci.* **12** 235

[33] Brainchip Inc. 2022 Akida examples *Akida Development Environment (MetaTF) Documentation* (available at: https://doc.brainchipinc.com/examples/index.html)

[34] Ivanov D, Chezhegov A, Kiselev M, Grunin A and Larionov D 2022 Neuromorphic artificial intelligence systems *Front. Neurosci.* **16** 959626

[35] Le V N T, Tsiknos K, Carlson K D and Ahderom S 2022 An energy-efficient akidanet for morphologically similar weeds and crops recognition at the edge *2022 Int. Conf. on Digital Image Computing: Techniques and Applications (DICTA)* pp 1–8

[36] Kadway C, Dey S, Mukherjee A, Pal A and Bézard G 2023 Low power and low latency cloud cover detection in small satellites using on-board neuromorphic processors *2023 Int. Joint Conf. on Neural Networks (IJCNN)* pp 1–8

[37] Abadi M *et al* 2015 Tensorflow: Large-scale machine learning on heterogeneous systems www.tensorflow.org/extras/tensorflow-whitepaper2015.pdf)

[38] Brainchip Inc 2022 CNN2SNN toolkit *Akida Development Environment (MetaTF) Documentation* (available at: https://doc.brainchipinc.com/user_guide/cnn2snn.html)

[39] Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H and He Q 2021 A comprehensive survey on transfer learning *Proc. IEEE* **109** 43–76

[40] Rommel C, Paillard J, Moreau T and Gramfort A 2022 Data augmentation for learning predictive models on eeg: A systematic comparison *J. Neural Eng.* **19** 066020

[41] Wang F, Zhong S h, Peng J, Jiang J and Liu Y 2018 Data augmentation for eeg-based emotion recognition with deep convolutional neural networks *MultiMedia Modeling* pp 82–93

[42] Khaliliardali Z, Chavarriaga R, Gheorghe L A and Millan J D 2015 Action prediction based on anticipatory brain potentials during simulated driving *J. Neural Eng.* **12** 066006

[43] Khaliliardali Z, Chavarriaga R, Zhang H, Gheorghe L A, Perdikis S and Millan J d 2019 Real-time detection of driver's movement intention in response to traffic lights *bioRxiv* (https://www.biorxiv.org/content/10.1101/443390)

[44] Garipelli G, Chavarriaga R and delR Millan J 2013 Single trial analysis of slow cortical potentials: A study on anticipation related potentials *J. Neural Eng.* **10** 036014

[45] Walker J 2007 Brain task map (available at: https://soft-dynamics.com/pdf/Brain_TaskMap_en.pdf)

[46] Walker J E 2009 Recent advances in quantitative eeg as an aid to diagnosis and as a guide to neurofeedback training for cortical hypofunctions, hyperfunctions, disconnections and hyperconnections: Improving efficacy in complicated neurological and psychological disorders *Appl. Psychophysiol. Biofeedback* **35** 25–27
[47] Esser S K *et al* 2016 Convolutional networks for fast, energy-efficient neuromorphic computing *Natl Acad. Sci. USA* **113** 11441–6
[48] DeWolf T, Jaworski P and Eliasmith C 2020 Nengo and low-power ai hardware for robust, embedded neurorobotics *Front. Neurorobot.* **14** 568359
[49] Vitale A, Renner A, Nauer C, Scaramuzza D and Sandamirskaya Y 2021 Event-driven vision and control for uavs on a neuromorphic chip *2021 IEEE Int. Conf. on Robotics and Automation (ICRA)*