university of groningen / faculty of science and engineering

Transfer Learning in Spiking Neural Networks

Master's Project

Petra Leferink (s3172783)
Msc. Artificial Intelligence
Faculty of Science and Engineering, University of Groningen
Supervised by Prof. dr. Elisabetta Chicca, Prof. dr. Niels Taatgen, and Msc. Maxime Fabre

December 13, 2023

Abstract

Over the past decade, spiking neural networks coupled with event-based sensors have shown the potential for constructing energy-efficient and low-latency embedded systems. Existing embedded systems often employ fixed models from prior-to-deployment training, which can cause performance degradation from a shift of data distribution between domains due to noise, environmental conditions or user specificities. To mitigate this, transfer learning is employed, which involves initial training in a source domain to grasp common knowledge that exists across domains, followed by fine-tuning in a target domain. This study specifically explores the effectiveness of transfer learning with Spiking Neural Networks (SNNs) in speech recognition tasks. The focus lies on speech processed with artificial cochleas, where differences in information density across domains can arise from different temporal resolutions. To delve into the dynamics of transfer learning, we explored the impact of SNN, training-related, and data-processing parameters on the performance. The results show that transferring from a source domain with a greater information density to a less information-dense target domain outperforms the opposite transfer learning scenario. Furthermore, smaller differences between domains yield better final performance. The neural membrane potential time constant influences the performance of the model greatly, as we observed that a model with smaller neural membrane potential time constants outperformed a model with larger ones. Lastly, an analysis of the synaptic weights of the network showed that the weights to the output layer change the most during the target domain training. These results advance our knowledge of the impact of transfer learning on event-based data coupled with SNNs. The insights regarding the influence of various parameters on performance may have practical implications, potentially enhancing the development of more effective embedded systems.

Contents

1 List of Acronyms

ACC Accumulate

ALIF Adaptive Leaky Integrate-and-Fire

ANN Artificial Neural Network

BPTT Backpropagation Through Time

BP Backpropagation

DFA Direct Feedback Alignment

e-prop eligibility propagation

FA Feedback Alignment

IF Integrate-and-Fire

LIF Leaky Integrate-and-Fire

MAC Multiply-and-Accumulate

RNN Recurrent Neural Network

SHD Spiking Heidelberg Digits

SNN Spiking Neural Network

2 Introduction

Nowadays, an increasing number of devices feature built-in voice assistants. A crucial component for effectively operating these voice assistants is the inclusion of a suitable sensor. The often-employed traditional sensors operate by sampling signals at a specific frequency at which each element transmits its current value. While proven to be highly successful, these traditional sensors have some drawbacks [34]. Due to their fixed sampling rate, these sensors cannot capture rapid changes in the signal that unfold between two sample moments. Another consequence of this fixed sample rate is the transmission of a considerable amount of redundant information when the input signal does not change, which leads to potential energy waste.

To solve these issues, event-based sensing has emerged as a promising technology for efficient and low-latency embedded systems over the past decade. Event-based sensors acquire and transfer information differently than traditional sensors [34]. Instead of providing absolute values at a given frequency, event-based sensors employ the 'send-on-delta' sampling scheme. In this approach, a sensing element, such as a pixel, sends an event when this change exceeds a certain threshold [24]. The signal-driven sampling in event-based sensing offers several conceptual advantages [33]. First of all, since event-based sensors only collect information when there is a change in signal strength, no redundant information is transferred and processed. Second, because certain sensing elements remain silent when there is no change in signal intensity, others can respond with a higher frequency. This adaptive sampling frequency ensures that information is not lost when the rate of change is fast. Third, event-based sensors offer the possibility for lower power operations because of their efficiency in information collection. This can be an advantage in power-constrained scenarios like mobile devices. Lastly, combined with suitable hardware, event-based sensors can sense with low latency. This is because of their ability to present information at an adaptive frequency.

One category of event-based sensors comprises artificial cochleas [33]. Examples of artificial cochleas are the address-event representation ear [7], the resonant gate transistor microphone [19], the binaural auditory sensor of Jiménez-Fernández et al. [15], and the silicon cochlea system of Wang et al. [36]. These sensors draw inspiration from the human auditory system, producing a temporally structured sparse stream of digital address events as their output. Each address corresponds to an active channel that represents a frequency band. With voice assistants in mind, a possible application of these artificial cochleas is to utilize them in speech recognition tasks. For example, Abdollahi and Liu [1] employed an artificial cochlea successfully for digit recognition. Another example comes from the research of Jansen and Niyogi [14], in which an artificial cochlea solves speech recognition tasks across varying noise levels.

Based on the sparse stream of information produced by an artificial cochlea, tasks like speech recognition can be solved by deploying a Spiking Neural Networks (SNNs). The neurons of an SNN communicate with asynchronously emitted binary spikes [3] rather than bits or numbers that are synchronously produced by each layer in a traditional artificial neural network Artificial Neural Network (ANN) [21]. Each neuron in an SNN accumulates the incoming spikes into its membrane potential. Optionally, the membrane potential decays when no spike is received. Similar to biological neurons, the neuron emits a spike when the membrane potential crosses a firing threshold, after which the membrane potential is reset. This asynchronous nature of SNNs has several potential advantages. The first of them is that the need for computational resources is reduced due to less information being transferred [23]. Namely, neurons in an SNN only send information when the membrane potential crosses a threshold, in contrast to the synchronous transfer of information by each neuron in an ANN. The second potential advantage is the reduced need for computational resources due to the type of operations performed by an SNN. SNNs only perform Accumulate (ACC) operations because high precision weight values must be read and accumulated in the membrane potential when spikes are received [16]. On the other hand, the operations in ANNs are ACC because two high-precision values need to be multiplied before accumulating in the neuron. This is a more costly operation than ACC [12]. See Figure 1 for a graphical illustration of the operational differences between an ANN and an SNN. Lastly, SNNs offers efficient hardware implementation on, for example, the LOIHI chip proposed by Davies et al. [9].

For all neural networks, learning is crucial in enabling the network to tackle the tasks it is designed to solve. The strength of ANNs lies in their ability to learn from just a set of examples of the task, facilitated by powerful learning rules such as Backpropagation (BP) algorithm [21]. In BP, the weights of a network are iteratively updated based on the backpropagated gradient of the error. In this way, the difference between the network and target output is minimized. When an ANN is recurrently connected, it can be trained with

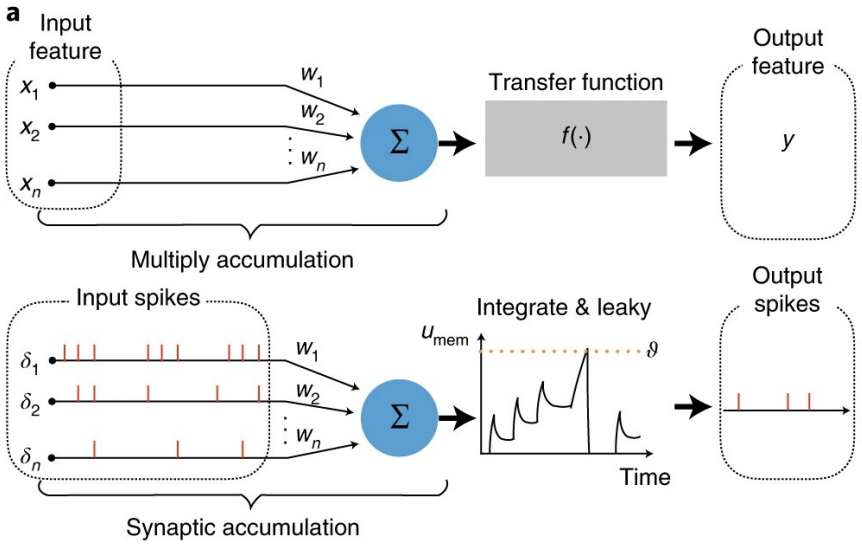


Figure 1: Comparison of ANN and SNN computational complexity. The upper scheme represents an artificial neuron that computes a weighted sum over the input and generates the output from this sum using a non-linear transfer function, which is a Multiply-and-Accumulate (MAC) operation. The bottom scheme displays a spiking neuron that receives weighted spikes that are accumulated into the membrane potential, including decay when no spikes are received, which is a Accumulate (ACC) operation. The membrane potential develops further through time, according to differential equations. When the membrane potential crosses a threshold, a spike is emitted, and the potential is reset. Adapted from [39].

Backpropagation Through Time (BPTT). In BPTT, the network is virtually unrolled in time to a feed-forward neural network. Next, the backpropagation algorithm is applied to this feed-forward neural network, where the gradient is backpropagated through the unrolled temporal dimension as if it were an extra spatial one [21].

SNNs possess inherently recurrent connections because the neurons in SNNs use the membrane potential in the previous timestep when calculating the current membrane potential. Therefore, when introducing a pseudo-derivative to handle the non-differentiable dynamics of spiking neurons, BPTT can also be employed to train an SNN [5]. The need for a pseudo-derivative arises due to the non-differentiable nature of SNNs induced by the spikes, which are not differentiable events. Nevertheless, using BPTT in SNNs has some drawbacks [3]. The first is that BPTT is computationally heavy in terms of the memory it needs to store the intermediate neuron states for the backward pass as well as in terms of the computational operations in BPTT. This may be impractical when deployed on hardware. Another drawback is that BP(TT) is not online because the error needs to be propagated backwards in time, which causes phase locking. Therefore, a more favourable learning rule to train an SNN is eligibility propagation (e-prop) [3]. E-prop resolves the need to unroll the network in time and to back-propagate the gradient of the error by introducing eligibility traces. In their turn, the eligibility traces keep track of the temporal information. In this way, e-prop resolves the offline nature and high computational needs of BPTT.

When using an embedded intelligent processor combined with an event-based sensor, common practice involves training the system in the pre-deployment phase to use it for inference in the post-deployment phase. In many cases, this system is not entirely asynchronous and, as a result, requires binning of the continuous data into discrete timesteps. Even when the system is completely asynchronous, binning might be needed for data compression. This binning of data can be on various levels depending on the used processor. Therefore, the data obtained with the sensor in the post-deployment phase can have a different temporal resolution than the data in the pre-deployment phase. Moreover, in cases where fully continuous asynchronous systems are used in both phases, the temporal precision can also differ due to differences in sensitivity in different sensors. In this context, there are two possible scenarios to consider. First, the precise binning procedure in the post-deployment phase might not be known during the pre-deployment training. Second, the sensor in the post-deployment phase may have a very high resolution or temporal precision, which asks for overwhelming computational demands in pre-deployment training. To mitigate this, the pre-deployment training might be on a lower-resolution dataset to ensure feasibility.

These scenarios can be framed as a shift in data distribution in the source domain (pre-deployment phase) and the target domain (post-deployment phase). Here, the shift in data distribution is caused by the different temporal resolutions. When the system is exclusively trained in the source domain, it may not perform optimally on the post-deployment data in the target domain due to this change in data distribution caused by a shift in temporal resolution. In such cases, domain adaptation is desired [38]. Domain adaptation entails initial training in a source domain followed by training in a different-but-related target domain such that the training in the source domain allows for more effective generalization to the target domain. Consequently, achieving effective domain adaptation can improve the system's post-deployment performance.

For ANNs, achieving effective domain adaptation can be accomplished by employing one of the many existing transfer learning methods [27, 41, 38]. However, limited research has been done on transfer learning in SNNs. An example of domain adaptation by transfer learning is the study of Zhan et al. [40]. They use feature similarity measures to improve the performance of object recognition tasks with images from different source and target domains. However, their study does not include an exploration of the impact of different SNN parameters, and since they employ image data, the aspect of temporal resolution remains unexplored.

Hence, in this study, we utilize transfer learning to achieve domain adaptation for SNNs, where the domains differ from each other in temporal resolution. This objective is analyzed within a simulated setting while solving a speech recognition task on the Spiking Heidelberg Digits (SHD) dataset [8]. The SHD dataset consists of spoken digits that are converted to spikes by an artificial cochlea model. This dataset includes binning to discrete timestep and, thus, a temporal resolution. The difference in temporal resolution between the source and target domain is investigated in both directions: from a higher resolution in the source domain to a lower resolution in the target domain and from a lower resolution in the source domain to a higher resolution in the target domain. Furthermore, we explore to what extent various SNN parameter settings influence the effectiveness of the transfer learning.

In the remainder of this report, the used neuron model, e-prop, domain adaptation, transfer learning and the SHD dataset are presented in more detail in the theoretical framework in Chapter 3. Chapter 4 describes the method used to implement domain adaptation with e-prop and the experiments performed to evaluate its efficiency. The results are presented and discussed in Chapter 5. Lastly, Chapter 6 gives the implications of the results, together with possible extensions and improvements of the framework, and concludes with future directions.

3 Theoretical Framework

The theoretical framework first describes the used neuron model in Section 3.2, followed by an explanation of the employed learning rule in Section 3.2. Furthermore, domain adaptation and how to acquire it are described in section Section 3.3. Lastly, the dataset is described in Section 3.4.

3.1 The (adaptive) leaky integrate-and-fire neuron

A prominent spiking neuron model is the Leaky Integrate-and-Fire (LIF) neuron. In short, spikes that arrive from other neurons through synaptic connections are multiplied by the corresponding synaptic weights and are integrated by the leaky membrane potential [3]. The neuron fires when its membrane potential reaches the firing threshold. In more detail, each LIF neuron has an observable state z^t and a hidden state $\mathbf{h}_j^t$. The observable state indicates whether the neuron j emits a spike ($z_j^t = 1$) or not ($z_t^j = 0$) at time t. In the case of a LIF neuron, the hidden state contains only the membrane potential v^t. The input I_j^t of the neuron is computed in the following way:

$$I_t^j = \sum_{i \neq j} W_{ji}^{rec} z_i^t + \sum_i W_{ji}^{in} x_i^{t+1}, \tag{1}$$

where x_j^{t+1} is the observable state of neuron i. Based on this input, the membrane potential of neuron j, v_j^{t+1}, is updated as follows:

$$v_j^{t+1} = \alpha v_j^t - z_j^t v_{th}. \tag{2}$$

In this equation, the term $-z_j^t v_{th}$ represents the reset of the membrane potential v_j after the neuron spikes. The decay factor α is defined as:

$$\alpha = e^{-\delta t/\tau_m}, \tag{3}$$

where δt is a timestep and τ_m is the membrane time constant. In this way, the time constant τ_m sets the membrane potential's decay speed. A higher τ_m leads to a slower decay of the membrane potential and visa versa. Based on the membrane potential v_j^t and the threshold potential v_{th}, the observable state z_j^t is calculated as:

$$z_j^t = H(v_j^t - v_{th}), \tag{4}$$

where $H(x)$ is the Heaviside step function:

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 0. \end{cases} \tag{5}$$

An extension of the LIF neuron introduces an adaptive component a_j^t to the effective firing threshold A_j^t, resulting in an Adaptive Leaky Integrate-and-Fire (ALIF) neuron. Among others, Bellec et al. [2] show that including this adaptive threshold component can enhance the performance of SNNs that includes explicit recurrence when the data has a temporally rich structure. Now, the hidden state contains a second variable a_j^t that sets the variable component of the firing threshold, and thus, $\mathbf{h}_j^t = [v_j^t, a_j^t]$. The adaptive component of the threshold increases with every output spike and then decreases exponentially back to the baseline threshold v_{th}. This mechanism can be described as follows:

$$z_j^t = H(v_j^t - A_j^t), \tag{6}$$

$$A_j^t = v_{th} + \beta a_j^t, \tag{7}$$

$$a_j^{t+1} = \rho a_j^t + z_j^t, \tag{8}$$

$$\rho = e^{-\delta t/\tau_a}. \tag{9}$$

In these formulas, the threshold factor β determines the impact of this adaptive component on the effective firing threshold, thus setting the rate of change in the threshold potential. Lastly, τ_a is the adaptation time constant.

3.2 Learning Rules

Backpropagation through time As mentioned in Chapter 2, LIF and ALIF neurons inherently exhibit recurrent connections since they update their membrane potential using their own observable state from the previous timestep (See Equation 1 and Equation 2). With some adaptations, an SNN can be trained with Backpropagation Through Time (BPTT), the conventional learning rule used for training artificial Recurrent Neural Networks (RNNs) [13].

In BPTT, the network is unrolled to a virtual feedforward neural network and the backpropagation algorithm is applied to it [21]. The network parameters are optimized by minimising a given loss or error, defined by a function E (see Figure 2a-c for a graphical representation of BPTT). in the forward pass, the input is passed through each layer of the unrolled version of the network. The error E is calculated based on the network's output. In the backward pass, the error E is backpropagated through the network to calculate the gradient of the error $\nabla E = \frac{dE}{dW_{ji}}$ for each weight W_{ji} from neuron i to neuron j. The gradient ∇E specifies how the weight W_{ji} should change to reduce the loss E. By using the chain rule to compute the partial derivatives, the gradient ∇E is determined. For SNNs, this gradient can be estimated with a pseudo-derivative for spikes, given the implicit discrete variable z_j^t is non-differentiable [13]. Even though good results can be obtained with BPTT on SNNs, several potential issues are associated with its implementation.

One category of challenges is related to the offline nature of BPTT [3]. As illustrated in Figure 2b, in the forward pass, all timesteps of a sample need to be passed through the network before the error is calculated. In the backward pass (Figure 2c), the error is propagated backwards through all timesteps of the sample to calculate the gradient ∇E for each timestep. This phase locking causes BPTT to be offline and computationally heavy regarding memory uses, as the network must store the intermediate neuron states for later use in the backward pass. As introduced in Chapter 2, a learning rule that can alleviate these issues is e-prop [3]. This learning rule simplifies the gradient and introduces an online error module, and consequently makes learning online in time and less computationally demanding.

Another potential problem of traditional BPTT is the symmetry problem or weight transportation problem [11]. Typically, in the BPTT process, the layer-by-layer backpropagation of the error is executed through the transpose of the weights in the forward pass. This can be an issue because each neuron needs to have information about all downstream synaptic weights, and the prediction and learning phases are separated, which makes the learning rule not online in time and space [25]. To overcome this challenge, Nøkland et al. [25] propose Direct Feedback Alignment (DFA), assigning a random fixed backwards weight from the output to each neuron.

Eligibility propagation The key innovation that makes e-prop an online learning rule is that the gradient $\nabla E = \frac{dE}{dW_{ji}}$ is represented as a sum of products over the time steps t of the SNN computation, where the second term is a local online gradient that does not depend on the error E:

$$\frac{dE}{W_{ji}} = \sum_t \frac{dE}{dz_j^t} \cdot \left[\frac{dz_j^t}{dW_{ji}}\right]_{\text{local}}. \tag{10}$$

The local gradient $\left[\frac{dz_j^t}{dW_{ji}}\right]_{\text{local}}$ is a sum of the partial derivatives concerning the hidden state $\mathbf{h}_j^t$ of neuron j at time t, which is updated during the forward computation of the SNN by a recursion. The local gradient collects the maximal amount of information about the network gradient $\frac{dE}{W_{ji}}$ that can be locally computed in a forward manner. For simple neuron models, such as an LIF or ALIF neuron, this local gradient reduces to a variation of terms that are commonly referred to as eligibility trace for synaptic plasticity, denoted as e_{ji}^t:

$$e_{ji}^t = \left[\frac{dz_j^t}{dW_{ji}}\right]_{\text{local}}. \tag{11}$$

However, most biological neurons have additional variables that evolve on a slower timescale. These slower processes are crucial in SNNs to achieve similar computing capabilities to those of a long short-term memory network. Accordingly, the gradient of the error $\frac{dE}{W_{ji}}$ can be defined as a learning signal L_j^t that represents these slower processes. The learning signal L_j^t is approximated by considering only the current loss at the output neurons k of the SNN as follows:

$$L_j^t = \sum_k B_{jk}(y_k^t - y_k^{*,t}), \tag{12}$$

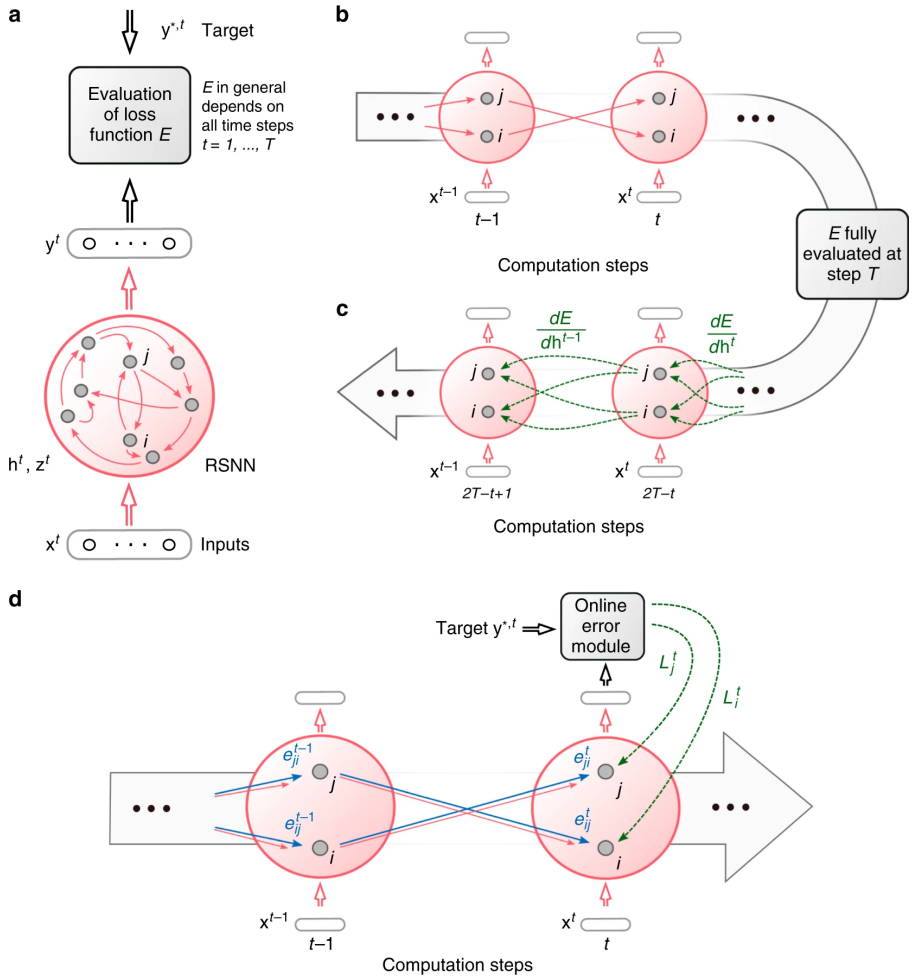


Figure 2: Comparison of the computations in BPTT and e-prop in an SNN. **a** An recurrently connected SNN with $\mathbf{x}^t$ as the inputs of the network, $\mathbf{h}^t$ as its hidden state and $\mathbf{z}^t$ as its observable state produces the output $\mathbf{y}^t$. Together with the target output $\mathbf{y}^{*,t}$, the output is evaluated with the loss function E. **b** When using BPTT, the network is virtually unrolled to a feedforward neural network, creating a new set of neuron copies for each timestep t. The connection between neuron i and neuron j is replaced by an array of feedforward connections, one for every timestep t, where each connection goes from neuron i at time t to neuron j at $t+1$. The connections in this array have all the same weight. **c** After the loss is computed, the gradients of this loss are propagated backwards in time and space through the rolled-out version of the network. **d** The online learning dynamics of e-prop involve the feedforward computation of eligibility traces (highlighted in blue). These are combined with the online learning signals (highlighted in green) according to Equation 13. Reprinted from Bellec et al. [3].

where $y_k^t - y_k^{*,t}$ is the deviation of the output of neuron k at time t y_k^t to the target output $y_k^{*,t}$. Furthermore, B_{jk} is the neuron-specific backward weight from output neuron k to neuron j, which can be set with several methods.

In conclusion, the network gradient $\frac{dE}{dW_{ji}}$ can now expressed as the following summation:

$$\frac{dE}{dW_{ji}} = \sum_t L_j^t e_{ji}^t. \tag{13}$$

A graphic representation of e-prop can be found in Figure 2d. Bellec et al. [3] demonstrated that e-prop can reach similar performance to BPTT on several speech recognition tasks.

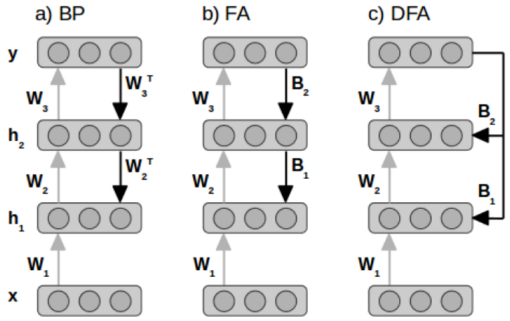


Figure 3: An overview of the different error transportation methods. The grey arrows represent the forward path of the input $\mathbf{x}$ through the hidden layers of the network $\mathbf{h}_i$ to the output layer $\mathbf{y}$ while multiplying with the corresponding weights $\mathbf{W}_i$. The black errors indicate the backpropagation of the error back through the network, where the weights depend on the transportation method. a) Symmetric weights in BPTT, where the backward weights are a transpose of the forward weights. b) Feedback Alignment (FA), where the backward weights are fixed random weights. c) Direct Feedback Alignment (DFA), where the backward weights are fixed random weights that are directly connected to the output layer. Adapted from Nøkland et al. [25].

Direct feedback alignment To address the weight symmetry problem, the backward weight B_{jk} in Equation 11 can be chosen with different methods. Liao et al. [22] show that these weights do not have to be the same as the forward weights as is usually the case in BPTT (see Figure 3a). Their method Feedback Alignment (FA) shows that a neural network can learn how to use fixed random weights to reduce the error (see Figure 3b). However, FA is still not local and online in time because the error needs to be backpropagated through every layer. To solve these issues, Nøkland et al. [25] propose DFA. In DFA, all backward weights B_{jk} are directly connected from the output layer to each neuron k in the network (see Figure 3c). Therefore, the error does not need to be backpropagated through each layer but is directly communicated to the other layers in the network. Therefore, there is no need for a separate backward phase, and the feedback becomes an extension of the forward pass. In their study, Nøkland et al. [25] show that this method works as well as BP and FA on several classification tasks.

3.3 Acquiring domain adaptation

Domain Adaptation Acquiring effective domain adaptation entails effectively using the knowledge gained from the source domain $\mathcal{D}_S$ to help the learning of a different-but-related target domain $\mathcal{D}_T$ [27]. Each domain consists of a sample space $\mathcal{X}$, a probability distribution $\mathcal{P}(X)$, where $X \in \mathcal{X}$, and a label space $\mathcal{Y}$. This study assumes that the source domain $\mathcal{D}_S$ and the target domain $\mathcal{D}_T$ have different probability distributions but the same label space. Therefore, the relationship between the two domains can be described as $\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{P}(X_S) \neq \mathcal{P}(X_T)$, $\mathcal{P}(y_S|X_S) \neq \mathcal{P}(y_T|X_T)$, and $\mathcal{Y}_S = \mathcal{Y}_T$. As described in the introduction (see Chapter 2), several scenarios arise in which the source domain $\mathcal{D}_S$ is different than the target domain $\mathcal{D}_T$ when utilizing event-based sensors and SNNs for spoken language. In some instances, this discrepancy involves a shift in temporal resolution resulting from the binning of data along the temporal dimension.

Transfer learning The domain adaptation can be effectively acquired through the implementation of different transfer learning techniques. The underlying idea of transfer learning is that a latent common knowledge representation in features can be learned during the training in the source domain and subsequently improves the training in the target domain in terms of accuracy or training efficiency [38]. One common approach to transferring this shared knowledge is copying the weights (of some layers) from the model that is trained in the source domain.

In the area of speech, common features extend across different domains and even across languages [35, 37]. These features can be acquired through training in the source domain and then used to enhance training in the target domain. Considerable research has delved into employing transfer learning for deep neural networks within the domain of spoken language. For instance, Kunze et al. [20] demonstrated that pre-training a model on an English dataset (source domain) facilitated faster training on a German dataset (target domain). Notably, small adaptations in the network's weights were sufficient for good performance in the target domain. Furthermore,

Qin et al. [29] conducted a study in which a deep neural network was initially trained in a source domain that included multiple languages. By transferring the weights of some layers to the target domain model, which only had one language, they achieved state-of-the-art results on several speech recognition tasks. Another example is the study of Wang and Hansen [37]. Their objective was to develop a model capable of speaker recognition across different acoustic domains, including various microphones and different types of background noises. The training was initially performed in a source domain that consisted of augmented data from various speakers, sources, and background noise levels. Subsequently, in the target domain, only the last few layers of the model were retrained. Results showed that the source-domain training accelerated the recognition of different speakers in the target domain.

These studies highlight the effectiveness of transfer learning in the field of spoken language. Nevertheless, minimal attention had been devoted to transfer learning for SNNs [40], particularly within the domain of speech. Simultaneously, SNNs, combined with event-based sensors, are very suited to deal with the temporal nature of speech [3, 34]. Moreover, given the practical scenarios demanding domain adaptation when employing SNNs and event-based sensors, this study aims to explore transfer learning for SNNs, specifically within the context of speech recognition.

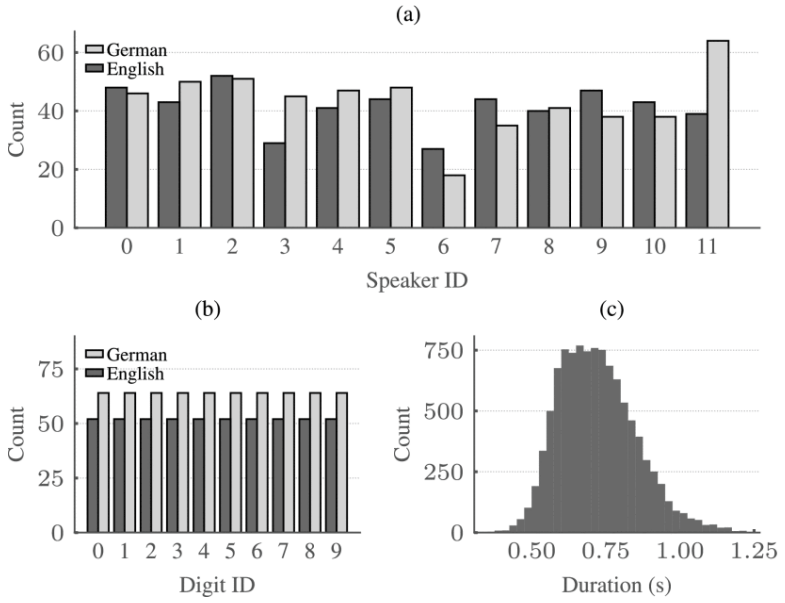


Figure 4: SHD dataset characteristics. a) The number of spoken digits in English and German per speaker. b) The number of samples per digit in English and German. c) A histogram of the duration of the samples. Reprinted from Cramer et al. [8].

3.4 Spiking Heidelberg Dataset

The dataset used in this study is the Spiking Heidelberg Digits (SHD) dataset [8], a benchmark speech recognition for SNNs. The dataset consists of twenty classes of English- and German-spoken digits from eleven different speakers. The number of samples per user, number and language and the duration of the samples are displayed in Figure 4. The audio files of the spoken digits are converted to spikes by deploying an artificial cochlea model that consists of three parts:

1. First, the data is fed through a hydrodynamic basilar membrane. Different frequency bands cause movement at different locations at the basilar membrane. This leads to a spatial dispersion similar to the spatial dispersion in a human cochlea. The sensitivity to difference in frequency is determined by the number of locations at which the movement is measured, often referred to as channels. Each channel represents a different frequency band. Therefore, a higher number of channels results in each channel representing a smaller frequency band, which leads to the model being more sensitive to different frequencies.

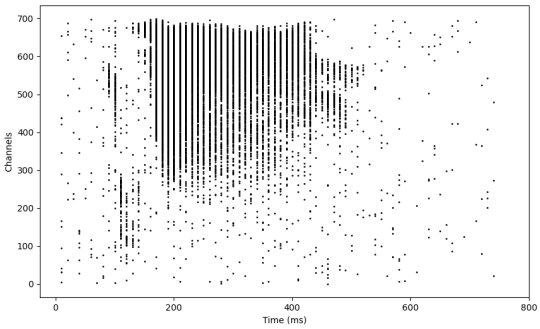
2. The movement of the basilar membrane is translated into spikes by a hair cell model. This biology-inspired model includes a certain probability for spiking and a refractory period in which no spike can be released. Forty hair cells per channel on the basilar membrane are used for the SHD dataset.

3. Lastly, bushy cells integrate the output of the hair cells per channel. In this case, there is a single bushy cell per group of forty hair cells. Furthermore, the bushy cells are implemented as LIF neurons.

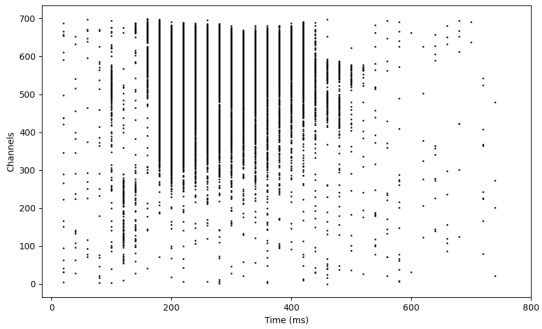
Quintana et al. [30] proved that an SNN trained with e-prop could learn to recognize the digits in the SHD dataset with an accuracy of approximately 80%.

4 Methods

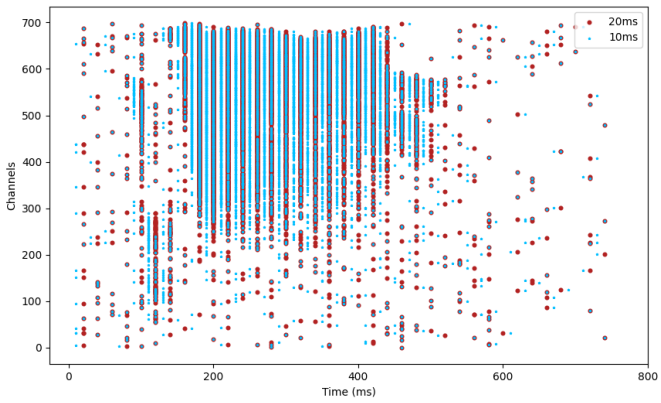
In this chapter, more information about the dataset is given in Section 4.1. Next, the transfer learning scheme is specified in Section 4.4. Section 4.2 and Section 4.3 describe, respectively, the SNN used in this study and the specifications of the e-prop implementations. The experimental set-up of this study is described in Section 4.5. Lastly, the metrics and the evaluation method can be found in Section 4.6.



(a) The spikes of a sample after the binning procedure with a window of 10 ms.



(b) The spikes of the same sample after the binning procedure with a window of 20 ms.



(c) The sample binned with a resolution of 20 ms in red and with a resolution of 10 ms in blue.

Figure 5: The comparison of a 10 ms resolution (Figure 5a) and a 20 ms resolution (Figure 5b). For extra clarity on the effect of different temporal resolutions, the result of both binning windows is compared in Figure 5c.

4.1 Dataset

Spiking Heidelberg Digits In this study, the SHD dataset as offered by Cramet et al. [8] is used [1]. As described in Section 3.4, this dataset consists of 8156 samples of spoken digits that are transformed into spikes by an artificial cochlea model with a basilar membrane composed of 700 channels. The division suggested by Cramer et al. [8] is followed to create a train and test dataset. In this partition, the samples of two speakers are exclusively in the test set. 5% of the samples are placed in the test set for the remaining speakers. The other 95% of the samples are assigned to be the train set. As this study focuses on domain adaptation, a dataset for both domains with a different temporal resolution is required. Therefore, the train and the test set are both subdivided into a set for the source domain and a set for the target domain. Randomly, 80% of the samples of both the train and test set are set to be the source train and test set. The target train and test set consists of

[1]The dataset is downloaded from `https://ieee-dataport.org/open-access/heidelberg-spiking-datasets` on March 2nd 2023.

the remaining 20% of the train and test set samples. The specific number of samples within each dataset can be found in Table 1.

Table 1: The **N train** and **N test** columns display the number of samples in respectively the train and test set for both the **Source** and **Target** domain. The remaining two columns give the mean ± standard deviation number of samples per class in the two train sets (**Mean per class train**) and in the two test sets (**Mean per class test**).

	N train	Mean per class train	N test	Mean per class test
Source	6524	326.2±10.48	1811	90.55±6.51
Target	1632	81.6±8.58	453	22.65±3.95

Temporal resolution To introduce different temporal resolutions in the source and target domains, we employ a binarizing binning method with different binning windows. The binning window determines the number of steps a sample is divided into. Within each binning window, the spikes are merged and binarized to 0 if there was no spike or else 1. Notably, each channel can only emit one spike per binning window, potentially losing information if a channel produces multiple spikes within the window. In this method, a smaller window results in the sample being divided into more steps, corresponding to a higher resolution. Figure 5 displays an example of applying different binning windows to the same sample. Figure 5a shows the sample with a binning window and thus a temporal resolution of 10 ms. Figure 5b shows the same sample, but now binned with a binning window of 20 ms, resulting in a sample with a resolution that is half as high as the resolution in Figure 5a. Figure 5c shows a comparison of the two resolutions.

4.2 The spiking neural network

The SNN utilized in this study has a single hidden layer of 450 ALIF neurons and an output layer of $k = 20$ ALIF neurons, each corresponding to a class of the SHD dataset. The neurons in the hidden layer are explicitly recurrently connected as this can improve the performance when using temporally rich structured data [6]. Following Bellec et al. [3], the ALIF neurons in the network have a refractory period of 5.0 ms after emitting a spike. During this refractory period, the observable state z_j^t is fixed to zero, and thus, the neuron cannot emit a spike.

The SNN makes a prediction *pred* by summing the spikes emitted by the neurons in the output layer over each timestep in the sample. The prediction *pred* of the network is then set to be the label that is represented by the neuron whose sum has the highest value:

$$pred = \underset{k}{\operatorname{argmax}}\left(\sum_{t=0}^{T} y_t^k\right). \tag{14}$$

4.3 E-prop implementation

E-prop is implemented as described in Section 3.2, with the following specifications. The method used to determine the backward weights B_j^t is DFA [25]. As described in Section 3.2, this method entails setting all backwards weights to a random fixed value and connecting them directly to the output layer. Furthermore, the pseudo-derivative ψ_j^t to deal with the non-differentiable nature of the spiking neurons in this study is a linear one:

$$\psi_j^t = \frac{1}{v_{th}}\, y_{pd} \max\left(0, \left|\frac{v_j^t - A_j^t}{v_{th}}\right|\right), \tag{15}$$

where the scaling factor y_{pd} is set to 0.3 [3].

Lastly, the weights are updated using the Adam optimizer [17] based on the calculated gradients.

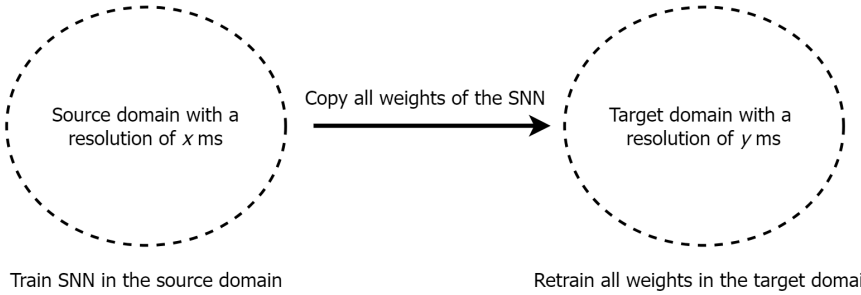


Figure 6: The transfer learning scheme adopted in this study. The source domain is displayed on the left. In this domain, the data has a temporal resolution of x ms, and the SNN is trained here for n epochs. Next, the complete SNN is retrained in the target domain (displayed on the right), where the temporal resolution is y ms.

4.4 Transfer learning scheme

In this study, domain adaptation is obtained by first training the SNN in the source domain, where the data has a temporal resolution of x ms for n epochs without prior knowledge of the temporal resolution in the target domain. This phase allows the SNN to learn the latent common features. Subsequently, all the parameters of the SNN are transferred to the target domain. In this domain, where the data has a temporal resolution of y ms, the complete SNN undergoes retraining for m epochs, during which all weights are fine-tuned and updated. The aim of this target domain training is for the model to adapt to the new domain and thereby enhance its performance within that domain. As described in Chapter 2 and Section 3.3, this adaptation is needed because there is a potential degradation in performance resulting from a shift in data distributions between the two domains. A graphical illustration of this transfer learning can be found in Figure 6.

Table 2: The default resolutions and numbers of source domain training in the different parameter experiments for the **Reduced resolution** and **Increased resolution** experiments. **Source resolution** gives the default resolution in the source domain and **Target resolution** in the target domain. Furthermore, the default number of epochs of source domain training is displayed in the **Source training** columns and the number of epochs of target domain training in **Target training**.

	Source resolution	**Target resolution**	**Source training**	**Target training**
Reduced resolution	10 ms	20 ms	100	10
Increased resolution	20 ms	10 ms	100	50

4.5 Experimental set-up

In addition to investigating the overall working of the transfer learning scheme, a series of experiments are conducted to delve into the impact of various parameters on the effectiveness of the transfer learning. The parameters that are explored are the temporal resolution, membrane time constant τ_m, the threshold factor β, and the number of epochs of source domain training.

The experiments are all done in two directions: from a higher resolution to a lower resolution and vice versa. In each experiment, Xavier initialization is used to initialize the weights [10]. Furthermore, the membrane potential is initialized as $v_j^0 = 0$, and the adaptive threshold as $A_j^0 = 0$. All hyperparameter settings of each experiment can be found in Appendix A. Unless mentioned otherwise, the following resolutions are used (see Table 2). For the experiments where the resolution reduces between the source and target domain, the resolution is 10 ms in the source domain and 20 ms in the target domain. For the experiments where the resolution increases between the domains, the resolution in the source domain is 20 ms and 10 ms in the target domain. Furthermore, unless mentioned otherwise, the training in the source domain is 100 epochs. The training in the target domain is 10 epochs for the reduced resolution experiments. In the increased resolution experiments, the target domain training is 50 epochs. The target domain training is longer in these experiments because there is more information in the target domain, which causes the model to need more time to adapt to the new data. The remainder of this section provides a more in-depth description of the experiments conducted for each parameter.

Temporal resolution experiments In these experiments, the influence of the shift in temporal resolution on the effectiveness of the transfer learning framework is explored. A larger shift in temporal resolution could imply a larger difference in data distribution, posing potential challenges for the network to adapt to the new domain during target domain training. Therefore, it is interesting to explore the network's capability to manage the shift in temporal resolutions between domains effectively. The resolutions that are included in these experiments are the following:

- **Reduced resolution:** The resolution in the source domain is 10 ms. In the target domain, the resolutions (in ms) are $\{12.5, 15, 20, 25, 30, 35\}$.
- **Increased resolution:** In this experiment, the resolution of the data in the source domain is 20 ms. The data in the target domain is binned with a window (in ms) of $\{1, 5, 10, 15\}$.

Membrane time constant experiments The influence of the membrane potential time constant τ_m on the effectiveness of transfer learning is also explored in two experiments. This is of interest because the time constants influence at what timescales the network is capable of capturing information [6]. Given the varying timescales in the source and target domain information, it is valuable to get insight into how the time constants affect the model's performance in different situations. In both the reduced and increased resolution experiment, the explored membrane potential time constants (in ms) are $\tau_m \in \{50, 100, 250, 400, 600\}$.

Threshold factor experiments Another parameter that could influence the effectiveness of the transfer learning is the neuron threshold factor β. This parameter sets how quickly the adaptive component of the firing threshold changes and might, therefore, also impact the network's capability to capture information at different timescales. Moreover, a threshold factor of $\beta = 0$ represents a LIF neuron, where the firing threshold is a fixed value. To explore the threshold factor's impact for both the reduced and increased resolution scenario, the experiment is repeated for $\beta \in \{0, 2, 4, 6, 8\}$.

Number of epochs of training in the source domain experiments The number of epochs of training in the source domain is also a factor that can influence the final performance of the model in the target domain. It could be that the model first learns more general features and, later, training more specialises in the specific dataset [21]. Learning these source domain-specific features might possibly have a negative influence on the performance of the model in the target domain. To investigate the influence of source domain training length on performance, the following experiments are run. In both the reduced and increased resolution experiments, the model is trained in the target domain for 40, 70 and 100 epochs. Each of these experiments is repeated for $\tau_m \in \{100, 150, 200, 250, \infty\}$. The last of these represents an integrate-and-fire neuron without leakage.

Weights analysis To gain insight into the specific weight changes occurring during target domain training, we conducted a small supplementary experiment. In this experiment, the resolution was higher in the source domain than in the target domain, and the default settings, as displayed in the reduced resolution case in Table 2, are used. The weights of the model at the end of the source and target domain training are compared.

4.6 Metrics and evaluation

Several metrics are used to evaluate the performance of the model. First and foremost, the accuracy at different moments in training is looked into. For the performance in the source domain, the accuracy of the source-domain training and test dataset at the end of the training in the source domain are considered. To investigate the performance in the target domain, the accuracy of the target-domain training and test set are considered at different moments. These accuracies are studied at the end of the training in the source domain, so at epoch zero of the training in the target domain. This is to see how the model performed in the target domain without additional target domain training. Second, the target-domain accuracies are considered after one epoch of training in the target domain to see how quickly the model starts to learn in the target domain-specific characteristic. Lastly, the target domain test set's final accuracy is defined as the maximum accuracy on this dataset during the training in the target domain.

To get a grasp of how much the model's performance can improve in the training in the target domain, the gain in accuracy is also considered in the first and last epoch of training in the target domain. The gain is defined as follows:

$$\text{gain} = \text{current target domain accuracy} - \text{target domain accuracy at epoch 0.} \tag{16}$$

Additionally, as mentioned earlier, each experiment is repeated three times to enhance reliability. Therefore, we analyzed the mean and standard deviation of each metric to better understand the variation between runs.

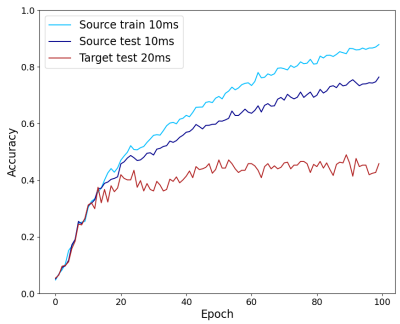
5 Results

Section 5.1 displays how the transfer learning scheme generally works. The following sections describe how different parameters influence the performance of the transfer learning scheme, with Section 5.2 changing temporal resolution, Section 5.3 changing the membrane potential time constant, Section 5.4 changing the threshold factor and Section 5.5 changing the number of epochs of source domain training for different membrane potential time constants. All sections first show the results of the higher to lower resolution and lower to higher resolution experiments, followed by a short discussion and comparison of the results. Lastly, the results of the weight analysis experiment are shown in Section 5.6.

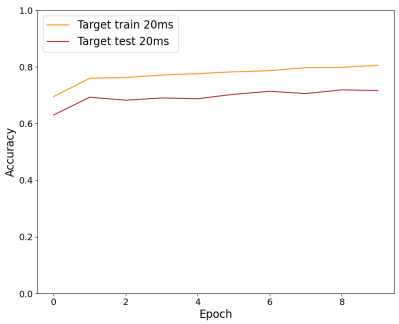
5.1 General working of the transfer learning scheme

Below are the results of two experiments with different shifts in resolution: one when the resolution is reduced when shifting to the target domain and one where the resolution is increased between the domains. The results are obtained with the parameter settings as displayed in Table 2 in Chapter 4 and Table 4 in Appendix A.

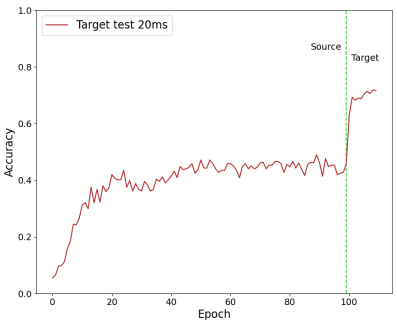
Reduced resolution After 100 epochs of training in the source domain, where the data has a resolution of 10 ms, the training accuracy reaches 89.3%, while the test accuracy is 78.0%. At that moment, without any training in the target domain, the accuracy on the target domain test set with a resolution of 20 ms is 41.4%. After a single epoch of training in the target domain, the accuracy on this target test set has already increased to 58.9%. The final test accuracy in the target domain is 71.9%, with a corresponding training accuracy of 80.5%. A graphical representation of the progression of the performances can be found in Figure 7. These results imply that the accuracy in the target domain can already increase substantially in only a few epochs of target domain training. This is advantageous, especially in the context of embedded systems, where post-deployment training tends to be costly [40].



(a) Accuracy on the source domain train and test set, and on the target domain test set, during the training in the source domain.



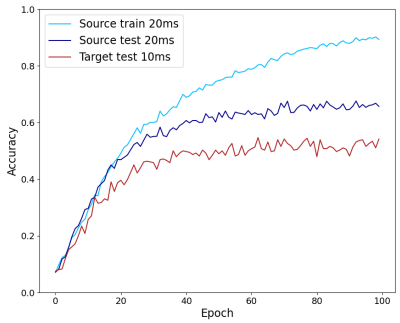
(b) The accuracy on the target domain train and test set during training in the target domain.



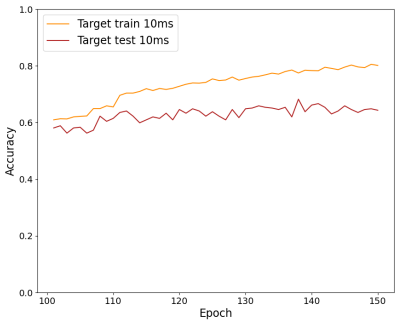
(c) Accuracy on the target domain test set during both training phases. The dashed green line marks the point where the training in the source domain is finished, and the training in the target domain starts.

Figure 7: The performances of the network in the reduced resolution experiment. The different accuracies during training in the source domain are displayed in Figure 7a and during the training in the target domain Figure 7b. Furthermore, Figure 7c shows the accuracy of the target test set during training in both the source and target domain.

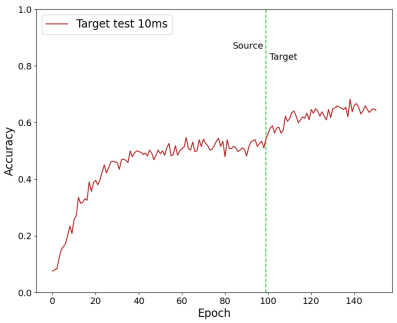
Increased resolution After 100 epochs of training in the source domain, with a 20 ms resolution, the training accuracy is 89.3%, while the test accuracy reaches 67.5%. Furthermore, the accuracy on the target domain test set, where the resolution is 10 ms, is 54.2%. After one epoch of training in the target domain, the accuracy on the target test set increases to 58.1%. Finally, the target train and test accuracies are 68.2% and 77.5%, respectively. Interestingly, the accuracy on the target test set slightly surpasses the source domain test accuracy. This indicated that the extra amount of information in the target domain data is relevant for the network to improve performance. A graphical representation of the progression of the performances can be found in Figure 8.



(a) Accuracy on the source domain train and test set, and on the target domain test set, during the training in the source domain.



(b) The accuracy on the target domain train and test set during training in the target domain.



(c) Accuracy on the target domain test set during both training phases. The dashed green line marks the point where the training in the source domain is finished, and the training in the target domain starts.

Figure 8: The different accuracies during training in the source domain are displayed in Figure 8a and during the training in the target domain Figure 8b. Furthermore, Figure 8c shows the accuracy of the target test set during training in both the source and target domain.

Comparison A first observation is that the accuracy of the source test dataset is higher in the reduced resolution experiment than in the increased resolution one (78.0% versus 67.5%). However, this pattern is not mirrored in the source training dataset, where performance remains approximately equal across both experiments (89.3%). This observation suggests that more overfitting occurs during the source domain training when the temporal resolution in that domain is lower. A hypothesis that could explain this is that the features in the source domain when the temporal resolution is 10 ms contain more relevant information and less noise than when the temporal resolution is 20 ms. Therefore, it could be that the model may become more attuned to noise in the source domain of the increased resolution experiment, which has a negative impact on the source domain test performance.

Additionally, it is noteworthy that the large jump in accuracy in the first epoch of target domain training, which can be observed in the reduced resolution experiment, does not occur in the increased resolution experiment. This suggests that the network finds it easier to adjust to a reduction in information within the data than to adapt to an increase in information. What also points in that direction is that the final performance in the target domain is lower in the increased resolution experiment (68.2%) than in the reduced resolution experiment (71.9%).

Lastly, it is noteworthy that the accuracy of the target test set in the reduced resolution experiment (71.9%) surpasses the accuracy of the source test set in the increased experiment (67.5%), while the temporal resolution in both datasets is 20 ms. Although these accuracies are not directly comparable due to different datasets, the contrast remains interesting to observe, as it could suggest ways to increase the performance on low-resolution data.

5.2 Different shift in temporal resolution

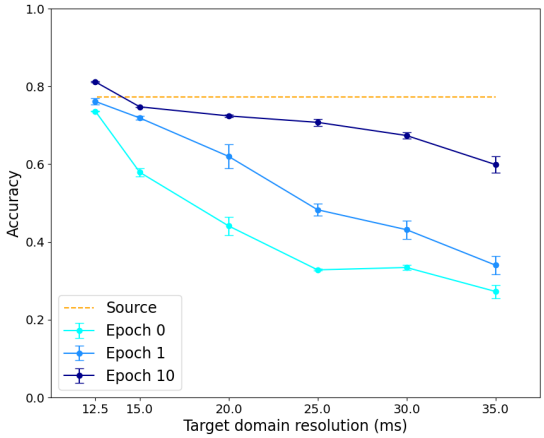
In these experiments, we explored the influence of different shifts in temporal resolution on the effectivity of the transfer learning scheme, both in the scenario where the resolution reduces and increases when shifting to the target domain. Both experiments involved a single source domain, and we examined the performance across diverse target domains.

Reduced resolution In this experiment, the model underwent initial training in the source domain with a 10 ms accuracy, achieving an accuracy of 78.0%. Subsequently, it was copied to multiple target domains, each characterized by a distinct temporal resolution.

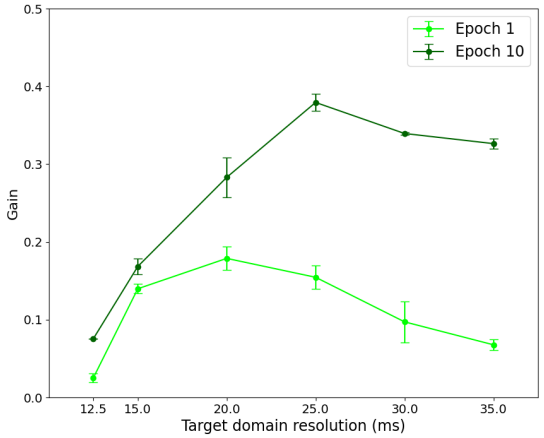
The results show that, without any training in the target domain, averaged of three runs, the accuracy on the target domain test set is between $27.3\% \pm 1.7\%$ for a resolution of 35 ms and $73.6\% \pm 0.2\%$ for a resolution of 12.5 ms (see Figure 9). The findings suggest that a smaller difference between the resolution in the target domain and the source domain leads to a higher initial test accuracy in the target. This can be caused by a smaller shift in data distribution between the two domains. During the target domain, the target test accuracy increases to a value between $81.2\% \pm 0.2\%$ for a resolution of 12.5 ms and $59.9\% \pm 2.1\%$ for a resolution of 35 ms. Interestingly, the final target test accuracy for a 12.5 ms resolution is consistently higher than the accuracy on the source test set. Notably, the accuracy for a 12.5 ms target domain resolution consistently surpasses the performance in the source domain, suggesting that the data at this resolution might contain less noise and more informative features.

Examining accuracy gains during target domain training, resolutions above 25 ms show an increasing trend in gain with higher resolution (see Figure 9b). Conversely, at lower resolutions, this trend diminishes. This suggests a limit to the adaptation achievable during training in the target domain.

All exact gain and accuracy values can be found in Table 5 and Table 6 in Appendix B.



(a) Accuracy on the target test sets with different resolutions at the end of training in the source domain (Epoch 0 in light blue), after one epoch of training in the target domain (Epoch 1 in medium blue), and the maximum target test accuracy during the training in the target domain (Maximum in dark blue). Furthermore, the dashed orange line gives the accuracy on the source domain test set at the end of the training in the source domain.



(b) The gain of accuracy on the target test set obtained with the target domain training after one epoch of training in the target domain (Epoch 1 in green), and the final gain in accuracy (Maximum in dark green).

Figure 9: Plots of the performances of the model for different shifts in temporal resolution in the reduced resolution experiment. The accuracy of the source target test set (Figure 9a) and the gained increase of accuracy of the target test set (Figure 9b) at different times during the training in the target domain for the different resolutions of the target dataset are displayed. Furthermore, each measuring point represents the average over the runs, with error bars representing the standard deviation.

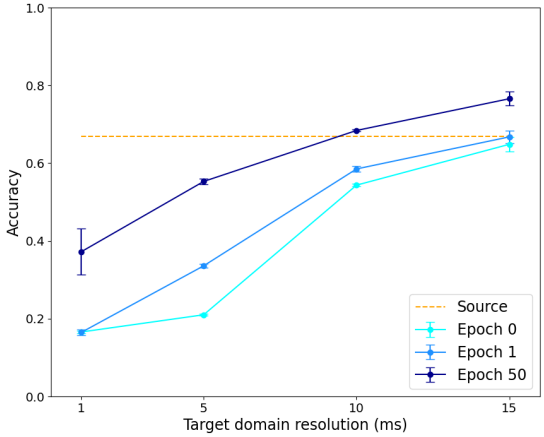
Increased resolution The results show that an accuracy of 67.5% on the source test set with a resolution of 20 ms is obtained with the source domain training.

Without any target domain training, the accuracy on the target test set is between $1.65\% \pm 0.3\%$ for a 1 ms resolution and $64.8\% \pm 1.8\%$ for a resolution of 15 ms (see Figure 10a). Similar to the reduced resolution experiment, the results suggest that a smaller shift in resolution leads to a higher initial performance in the target domain. At the end of the target domain training, the target test accuracy ranges between $37.2\% \pm 5.9\%$ for a resolution of 1 ms and $76.6\% \pm 1.8\%$ for a 15 ms resolution. Notably, the target test accuracy when the resolution is 10 ms or 15 ms is higher than the source test accuracy, suggesting that the additional information introduced in the target domain data contributes to improved performance.

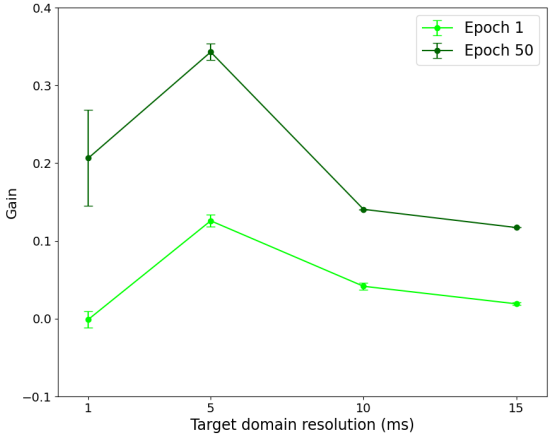
Furthermore, the maximum target test accuracy is consistently obtained after at least 35 epochs of training in the target domain, suggesting that the model requires substantial training to adapt to the new data.

Examining the gain in accuracy in the target domain, the highest gain is obtained for a resolution of 5 ms, both in the first epoch and the final gain. As can be seen in Figure 10b, the substantial improvement in accuracy in the first epoch of target domain training is absent for most of the resolutions. In this experiment, the large increase only occurs when the target domain resolution is 5 ms.

All gain and accuracy values of this experiment can be found in Table 7 and Table 8 in Appendix B.



(a) Accuracy on target test sets with different temporal resolutions at the end of source domain training (Epoch 0 in light blue), after one epoch of target domain training (Epoch 1 in medium blue), and at the end of the target domain training (Epoch 50 in dark blue). Furthermore, the orange dashed line shows the accuracy of the source test set at the end of the source domain training.



(b) The gain obtained with training in the target domain, for the different temporal resolutions, after one epoch of target domain training (Epoch 1 in green), and at the end of the target domain training (Epoch 50 in dark green).

Figure 10: Plots of the performances for different shifts in temporal resolution in the lower-to-higher experiment. The accuracy of the different target test sets and the gain in accuracy at different times during training can be found in, respectively, Figure 10a and Figure 10b. Each measuring point is an average of runs with error bars showing standard deviation.

Comparison Comparing the results of the experiments above, a prominent observation emerges: transitioning to an increased resolution results in less adaptation to the target domain compared to the reverse direction despite involving the same difference in temporal resolution. For instance, shifting from a 10 ms resolution in the source domain to a 20 ms resolution in the target domain leads to a target test accuracy of $72.4\% \pm 0.3\%$. In contrast, transfer from a 10 ms source domain resolution to a 20 ms target domain resolution gives a target test accuracy of only $68.4\% \pm 0.3\%$. This observation suggests that transitioning from a domain with higher resolution and, consequently, more detailed data to a domain with less detailed information appears to be more straightforward than the reverse.

However, this effect does not so clearly appear in the experiments where the resolution in the target domain is 15 ms. In the reduced resolution experiment, the target test accuracy is $74.7\% \pm 0.0\%$, while it is $76.6\% \pm 1.8\%$ in

the increased resolution experiment. Furthermore, there is an interesting difference in target domain performance when the model only underwent source domain training. In the reduced resolution experiment, this accuracy is $57.9\% \pm 2.2\%$, while it is already $64.8\% \pm 1.8\%$ in the increased resolution experiment. This suggests that when target domain training is not an option, training the model on lower-resolution data could be more beneficial. As illustrated in Figure 11, the dynamics during the target domain training also look different. In the reduced resolution experiment, the performance in the target domain increases quickly in the first epochs of target domain training, while it takes the model longer to adjust to the new domain in the increased resolution experiment.

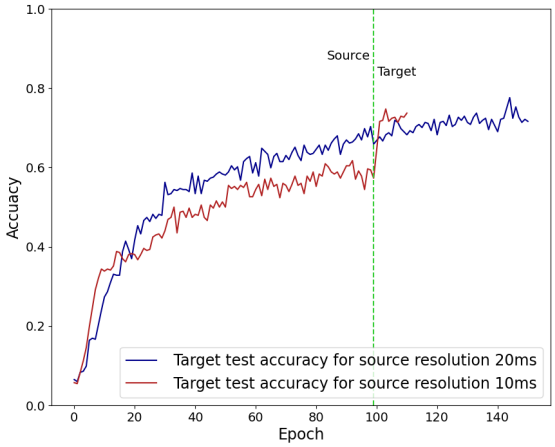


Figure 11: A comparison of the target test accuracy with a 15 ms resolution for the different source domains. The experiment when the source domain resolution is 10 ms is displayed in red. The experiment with a 20 ms source domain resolution is shown in blue.

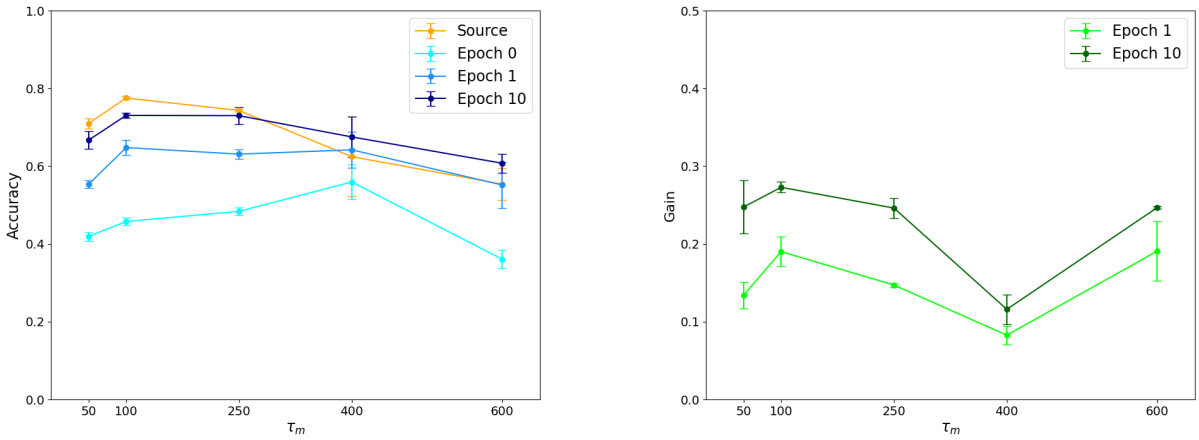
5.3 Different membrane potential time constants

To explore the influence of the membrane potential time constant τ_m on the transfer learning, we performed several experiments where the resolution reduced (10 ms to 20 ms) and where the resolution increased (20 ms to 10 ms) for different τ_m-values.

Reduced resolution As can be seen in Figure 12a, varying the membrane time constant τ_m introduces variability in the source domain performance. Notably, higher τ_m-values, and thus a slower leakage in the membrane potential, appear to cause poorer performance and greater variability across different runs.

The same trend can be observed for the target domain performance after one epoch and at the end of the target domain training. However, for $\tau_m = 400$ ms, the highest target domain accuracy obtained is $55.9\% \pm 4.5\%$ without any target domain adaptation. This suggests that when target domain training is not possible, setting τ_m to 400 ms might lead to optimal performance. Conversely, if target domain adaptation is possible, it is advisable to opt for a lower τ_m-value. Another notable observation is that the gain in accuracy obtained with target domain training for $\tau_m = 400$ ms is substantially lower than for the other τ_m-values, both the gain in the first and last epoch, which could be caused by the higher initial performance in the target domain (see Figure 12b).

All in all, the results of this experiment show that the choice of the τ_m-value impacts the performance in both domains substantially. Therefore, spending some time optimizing this parameter might be beneficial. Lastly, the gain and accuracy values of this experiment can be found in Table 9 and Table 10 in the Appendix B.

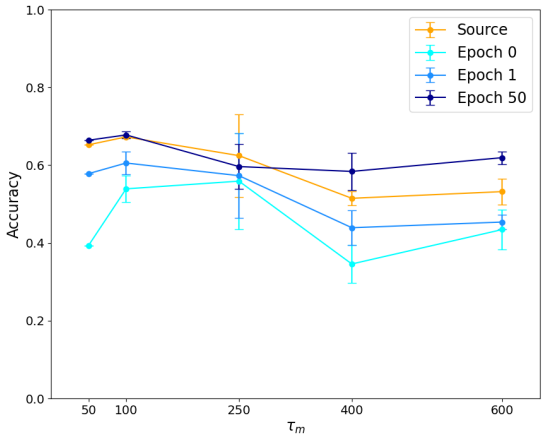


(a) Accuracy on the target test set at the end of the training in the source domain (Epoch 0 in light blue), after one epoch of training in the target domain (Epoch 1 in medium blue), and the final target test accuracy (Epoch 10 in dark blue) for the different τ_m-values. Furthermore, the source test accuracy is shown in orange

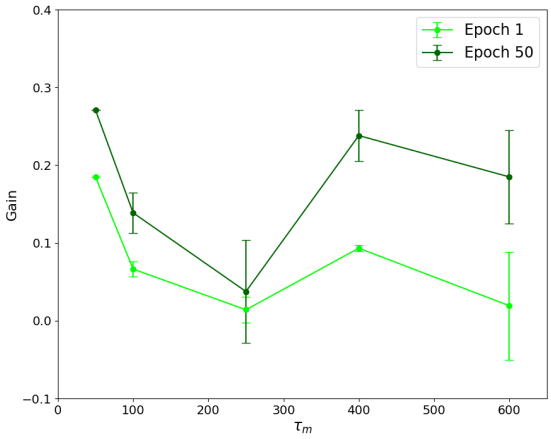
(b) The gain in accuracy of the target test set obtained with training in the target domain for the different τ_m-values. The gain after one epoch of training in the target domain is displayed in green, and the gain at the end of the target domain training is in dark green.

Figure 12: Plots of the performance for different membrane potential time constants τ_m when shifting from a higher to a lower resolution. The accuracy of the target test set (Figure 12a) and the gained increase of accuracy of the target test set (Figure 12b) are shown at different times during the training in the target domain for the different membrane time constants τ_m. Furthermore, the accuracy on the source test set is displayed in Figure 12b. The value obtained at each measuring point is an average of multiple runs, and the error bars show the standard deviation.

Increased resolution In this experiment, a similar trend in the source domain performances can be observed. Namely, the higher τ_m-values generally lead to a worse performance. Furthermore, the performances in the target domain broadly follow a similar trend. The large increase in accuracy in the first epoch of training in the target domain only occurs for the smallest τ_m-value, namely $\tau_m = 50$ ms (see Figure 13b). Adapting to the new domain takes comparatively longer for the other τ_m-values.



(a) Accuracy on the target test sets at the end of training in the source domain (Epoch 0 in light blue), after one epoch of training in the target domain (Epoch 1 in medium blue), and at the end of the training in the target domain (Epoch 50 in dark blue) for the different τ_m-values. Furthermore, the orange line gives accuracy on the source test set.



(b) The gain in target test accuracy obtained with training in the target domain, after one epoch of training in the target domain (Epoch 1 in green), and the final gain in accuracy (Epoch 50 in dark green) for the different τ_m-values.

Figure 13: Plots of the performance for different membrane potential time constants τ_m when the resolution increases between the domains. The accuracy of the target test set is shown in Figure 12a and the gained increase of accuracy during the target domain training in Figure 13b. Lastly, the performance in the source domain is displayed in Figure 13b. The value of each measuring point is calculated by taking an average of multiple runs, and the error bars in the graph represent the standard deviation.

Notably, the results exhibit great variation between runs for $\tau_m = 250$ ms. This is caused by a negative transfer that occurs in some runs. The decline in target domain performance follows a consistent pattern, as illustrated in Figure 14. It is possible that the issue is related to overfitting, which occurs when the model is too tuned to the noisy features in the data. Moreover, this result suggests that adding more information to the data, such as increasing the temporal resolution, may actually have a negative impact on performance in some cases. The results of this experiment suggest that the performance of the model is best for the low τ_m-values, both in terms of accuracy and in terms of variation between runs. Furthermore, all gain and accuracy values of this experiment can be found in Table 11 and Table 12 in the Appendix B.

Comparison Generally, the higher τ-values in both experiments perform worse than the smaller ones. This suggests that the leakage of the membrane potential plays an important role in the model's ability to learn, which aligns with the literature (see, for instance, Bellec et al. [2]). However, both experiments' target domain performance exceeds the source domain performance for these larger τ_m-values. This suggests that holding on to information for a longer period may be beneficial when transferring it to another domain.

Furthermore, the increased resolution experiment shows more variation between runs, suggesting that the learning is more unstable. This underlines that transferring from a lower to a higher resolution is more difficult for the network than the other way around.

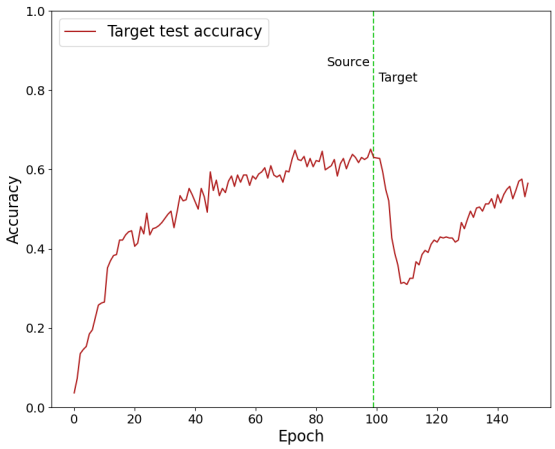


Figure 14: An example of the progression of the target test accuracy during the source (left of the green line) and target (right of the green line) domain training, when $\tau_m = 250$.

5.4 Different threshold factors

In these experiments, we explored the influence of the threshold factor β on transfer learning. Again, we performed several experiments where the resolution was reduced (10 ms to 20 ms) and where the resolution increased (20 ms to 10 ms). Importantly, a threshold factor of $\beta = 0$ represents the LIF neuron, which does not include an adaptive firing threshold.

Reduced resolution First and foremost, the results of this experiment demonstrate that incorporating an adaptive component in the neuron's firing threshold substantially enhances performance in both the source and target domain (see Figure 15a). The results suggest that the exact rate of change of the adaptive component does not influence the performance much, as there is not much difference in accuracies for the higher β-values. Interestingly, this trend is not reflected in the initial performance in the target domain when the model has not undergone any target domain training yet. Here, the target test accuracy is similar for all β-values.

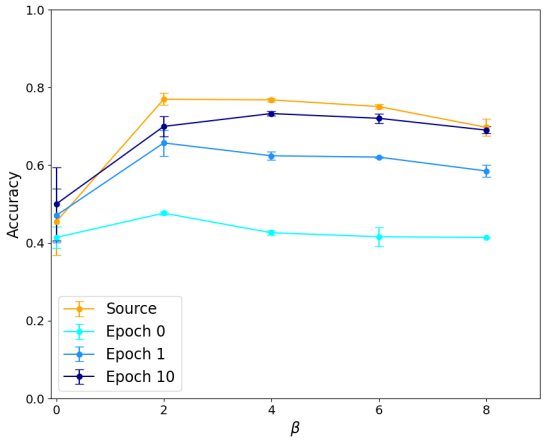
For all β-values, except for $\beta = 0$, the large increase in the accuracy on the target test set is obtained in the first epoch of training in the target domain (see Figure 15b.

Lastly, interesting to see is that for $\beta = 0$, the results show the most variation between runs both in the source and target domain, next to performing the worst. Together, these results all imply that adding the adaptive part to the neuron's firing threshold contributes greatly to the performance. The gain and accuracy values of this experiment can be found in Table 13 and Table 14 in the Appendix B.

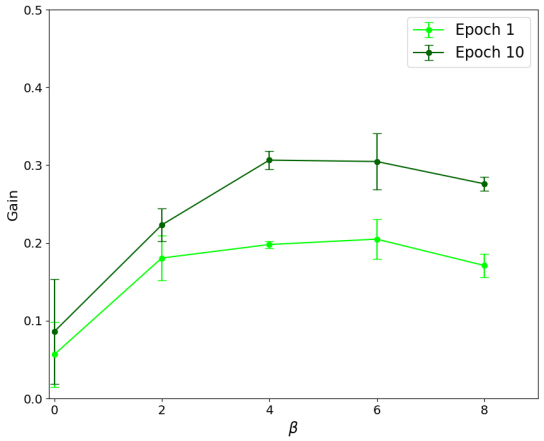
Increased resolution Similar to the reduced resolution experiment, employing a LIF neuron model ($\beta = 0$) shows the lowest accuracy on both the source and target test datasets (see Figure 16a). It is also similar that the exact value of β has little influence on performance.

The gain in accuracy during this experiment during this experiment remains relatively consistent across the β-values (see Figure 16b). The notable exception is observed for $\beta = 6$, where the accuracy gain displays substantial variation between runs ($7.3\% \pm 8.6\%$ in epoch 1 and $19.9\% \pm 6.9\%$ in epoch 50). This variability might be attributed to the target test accuracy at the end of source domain training, exhibiting considerable variation between runs ($45.9\% \pm 6.6\%$). However, the final target test accuracy stabilizes at $65.8\% \pm 0.3\%$. This suggests that the network can reach comparable performance, even when initiating target domain training from a less ideal starting point.

As in previous experiments with increased resolution, the results demonstrate that the model can exceed source domain performance with its final performance in the target domain. The gain and accuracy values of this experiment can be found in Table 15 and Table 16 in the Appendix B.

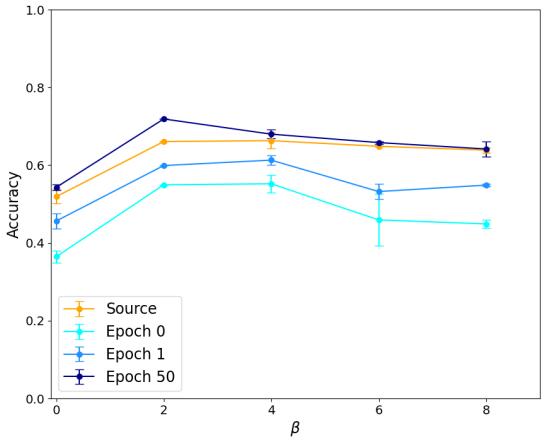


(a) Accuracy on the target test sets at the end of training in the source domain (Epoch 0 in light blue), after one epoch of training in the target domain (Epoch 1 in medium blue), and the final target domain accuracy (Epoch 10 in dark blue) for the different β-values. Furthermore, the source domain accuracy is displayed in orange.

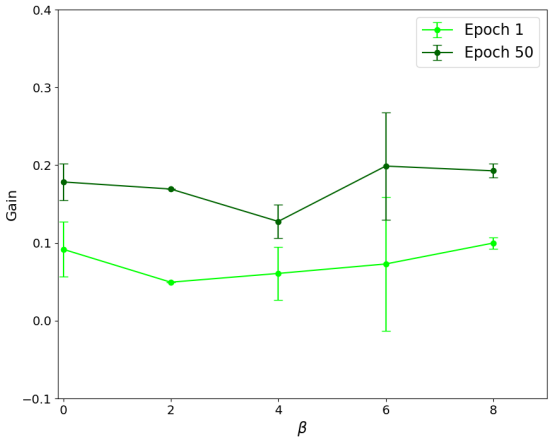


(b) The gain in target test accuracy after one epoch of training in the target domain (Epoch 1 in green) and at the end of the target domain training (Epoch 10 in dark green) for different β-values.

Figure 15: Plots of the accuracy and gain in accuracy of the reduced resolution experiment for different threshold factors β. The accuracy on the target test set is shown in Figure 15a and the gained increase of accuracy of the target test set in Figure 15b. The accuracy on the source test set is displayed in Figure 15a as well. Furthermore, each measuring point represents the average over the runs, with error bars representing the standard deviation.



(a) Accuracy on the target test sets at the end of the source domain training (Epoch 0 in light blue), after one epoch of target domain training (Epoch 1 in medium blue), and at the end of target domain training (Epoch 50 in dark blue) for the different β-values. Furthermore, the source test set accuracy is shown in orange.



(b) The gain in accuracy obtained with target domain training after one epoch of target domain training (Epoch 1 in green), and at the end of the target domain training (Epoch 50 in dark green) for the different β-values.

Figure 16: Plots of the performance of the model in an increased accuracy experiment for different threshold factors β. The accuracy of the target test set is shown in Figure 16a and the gained increase of accuracy of the target test set in Figure 16b. The accuracy on the source test set is displayed in Figure 16a too. Each measuring point is an average of the runs, with error bars showing standard deviation.

Comparison The result that stands out in these experiments is the inferior performance of the LIF neuron ($\beta = 0$), which aligns with existing literature [2, 39]. Additionally, it is interesting to note that the LIF neuron performs better in the increased resolution experiment than in the reduced resolution experiment. For the larger β-values, the exact value appears not to have a tremendous influence on the performance of the model in both domains. This implies that while neurons in the SNN require an adaptive component in their firing threshold, the precise rate of change of this adaptive component does not substantially impact the final performance.

5.5 Number of epochs training in the source domain

The results of the experiments of different lengths of source domain training are presented below. The various durations of source domain training are tested for both a reduced and increased target domain resolution. Furthermore, these experiments include an Integrate-and-Fire (IF) neuron model($\tau_m = \infty$, which is comparable to a LIF neuron but without the leakage.

Reduced resolution In all cases, the more epochs of training in the source domain, the better the performance in the target domain (see Figure 17). Noteworthy, the model performs substantially worse when an IF neuron model is deployed, implying that introducing leakage in the membrane potential enhances performance. When the source domain training is only 40 epochs, $\tau_m = 250$ ms performs best with an accuracy of $60.7\% \pm 0.0\%$, but it is not among the best-performing τ_m values when the source domain training is longer. This could mean that this τ_m-value allows the network to quickly learn the relevant dynamics for adapting to the target domain. However, if the source training is longer, the model may adapt too much to source-domain specific dynamics. The accuracy values of this experiment can be found in Table 17 in the Appendix B.

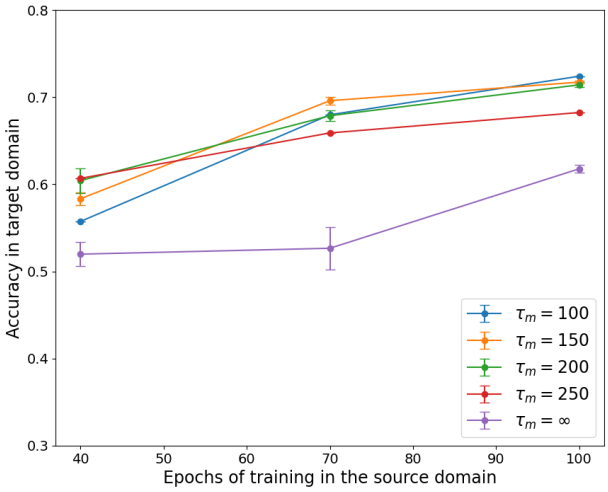


Figure 17: The accuracy on the target-test sets obtained during the target domain training for different numbers of epochs of training in the source domain. The different lines represent different membrane potential time constants τ_m. Additionally, each data point represents the mean of multiple trials, with error bars indicating one standard deviation.

Increased resolution Upon examination of the domain accuracies, it appears that the correlation between training more epochs in the source domain leading to improved performance in the target domain does not hold true for all τ_m-values in this experiment. While this relationship persists for the lower three τ_m-values, for $\tau_m = 250$, the model performs better with 40 or 70 epochs of source domain training and performs poorest when the source domain training was 100 epochs. Upon closer examination of the performance trajectory during target domain training, it becomes evident that this decline can be attributed to a pattern similar to that illustrated in Figure 14. In the case of the IF-neuron ($\tau_m = \infty$), the duration of source domain training appears to have minimal influence on target domain performance, with the model consistently exhibiting suboptimal results with great variation between runs. The accuracy values of this experiment can be found in Table 18 in the Appendix B.

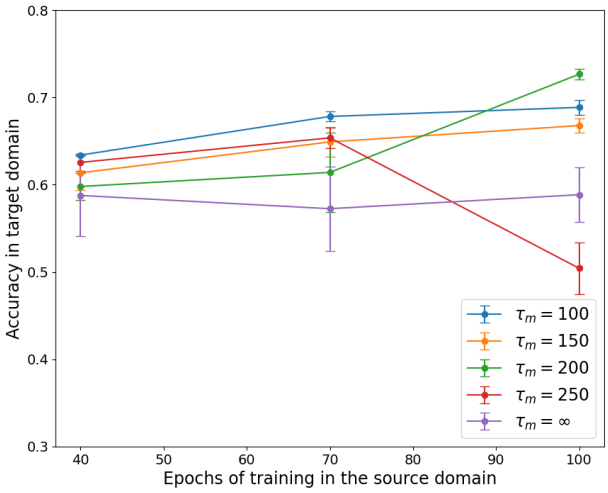


Figure 18: The accuracy on the target test sets obtained during the target domain training for different numbers of epochs of training in the source domain. The different lines represent different membrane time constants τ_m. Moreover, each data point represents an average value calculated from multiple runs, and the error bars indicate the standard deviation.

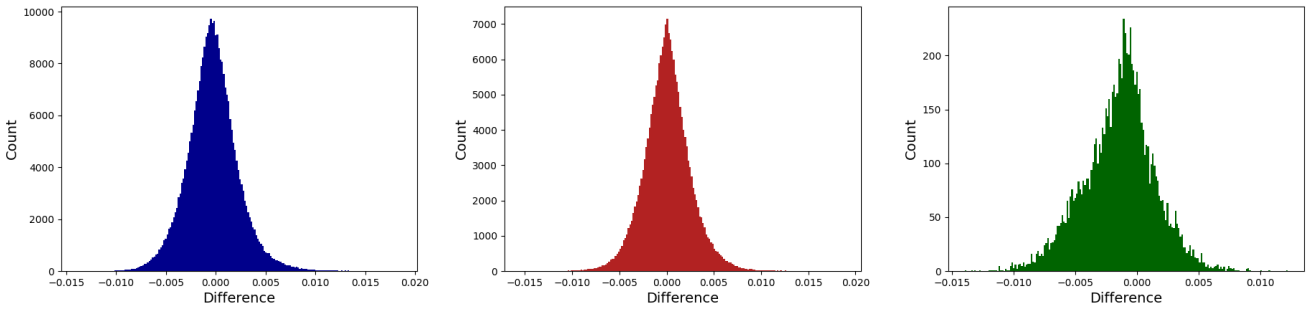
Comparison In general, the results show that a higher number of epochs of source domain training tends to yield better results. However, an exception arises in the case of the increased resolution transition with $\tau_m = 250$, which followed a comparable pattern as the results in Section 5.3. When considering the IF-neuron, it consistently demonstrates suboptimal performance. This aligns with findings in existing literature existing literature [6, 8, 2], which indicate that the leakage in the membrane potential is important when both temporal information is present in the data and there is recurrence in the network topology.

5.6 Weights analysis

When comparing the weights at the end of the source domain training, which has a 10 ms resolution, with those at the end of target domain training, where the resolution was 20 ms, it can be observed that, on average, the weights from the input to the hidden layer ($\mathbf{W_{in}}$) and the recurrent weights in the hidden layer ($\mathbf{W_{rec}}$) exhibit minimal alterations. In contrast, the mean change in weights between the hidden and output layer ($\mathbf{W_{out}}$) is roughly ten times greater, but the changes are still not very large. (see Table 3 and Figure 19). The limited changes in the weights may stem from the similarity in features in the two domains, as the only variation is a different temporal resolution.

Table 3: The **Mean** and standard deviation (**SD**) of the weights in the **Source Domain**, **Target Domain**, and the **Difference** between these weights.

	Source Domain			**Target Domain**			**Difference**		
	$\mathbf{W_{in}}$	$\mathbf{W_{rec}}$	$\mathbf{W_{out}}$	$\mathbf{W_{in}}$	$\mathbf{W_{rec}}$	$\mathbf{W_{out}}$	$\mathbf{W_{in}}$	$\mathbf{W_{rec}}$	$\mathbf{W_{out}}$
Mean	0.0028	-0.0228	-0.1521	0.0028	-0.0230	-0.1533	-0.0003	0.0001	-0.0014
SD	0.0764	0.1496	0.4025	0.0762	0.1499	0.4036	0.0026	0.0025	0.0029



(a) Histogram of the change in $\mathbf{W_{in}}$. **(b)** Histogram of the change in $\mathbf{W_{rec}}$. **(c)** Histogram of the change in $\mathbf{W_{out}}$.

Figure 19: Histograms of the weight change during the target domain training. The difference in the weights between the input and hidden layers ($\mathbf{W_{in}}$) is displayed in Figure 19a. Figure 19b shows the differences in weights of the recurrent connections in the hidden layer ($\mathbf{W_{rec}}$). Lastly, Figure 19b illustrated the change in the weights to the output layer ($\mathbf{W_{out}}$).

6 Discussion

In this study, we proposed a transfer learning framework incorporating source domain training, followed by an adaptation phase in the target domain, where both domains are characterized by a different temporal resolution. Additionally, we investigated the influence of several SNN parameters on the effectiveness of the transfer learning. The results show that the performance of the model is substantially improved during the target domain training for most of the parameter settings. Moreover, the smaller the shift in temporal resolution, the better the performance target domain performance was. Additionally, employing an ALIF neuron model resulted in superior performance compared to employing a LIF or IF neuron model. Interestingly, the rate of change of the adaptive component of the firing threshold, set by the parameter β, had minimal influence on the performance. On the other hand, the rate of leakage in the membrane potential had a notable impact, with both a very low and high decay having a negative influence on performance. Lastly, a longer source domain training duration generally corresponded with an enhanced final performance in the target domain.

All in all, the primary insight from our experiments is the positive impact of the domain adaptation to mitigate performance drop when transferring to the target domain. Given the limited existing research on transfer learning for SNNs, our study serves as a valuable addition, showcasing how target domain adaptation of the model can enhance the performance of an SNN across domains with varying temporal resolution. Moreover, exploring the impact of different SNN parameters yields noteworthy insights. For example, the value of the neurons' membrane potential time constant emerges as an important factor for the performance. This suggests that investing time in optimizing this parameter can be rewarding in practical implementations. In contrast, the precise rate of change of the adaptive component of the firing threshold does not have a big impact on performance, although this adaptive component is an important dynamic to have in the neurons to extract more complex temporal dynamics.

Next to this, the training length required for optimal performance in the target domain yields valuable insights. In the increased resolution experiments, optimal performance was often achieved towards the end of the target domain training, typically around forty epochs. Thus, a substantial amount of training in the target domain is still needed to reach an adequate performance. Consequently, whether or not this configuration sufficiently reduces the cost of training the model may depend on the particular use case. Contrarily, substantial accuracy gain is frequently obtained within the first epochs of target domain training in the reduced resolution experiments. Performing just one or two epochs of target domain training can yield considerable improvement. This is advantageous for practical implementation, especially as post-deployment training on deployed hardware can prove costly [40].

Nevertheless, there are many interesting directions that might improve the performance, which were beyond this study's scope. For example, in this study, all membrane potential time constants were homogeneously initialized, which could already lead to accuracies as high as 80%. However, several studies showed that heterogenous initialization and time-constant training can improve the performance of SNNs, especially when the data has a rich temporal structure [28, 39, 6]. In heterogeneous time-constant training, neurons' time constants are optimized during the training along the synaptic weights. In our situation, where the temporal information changes between the source and target domain, optimizing the membrane potentials to the new temporal situation might be extra beneficial.

Moreover, we attempted to implement BPTT as the source domain learning rule. However, similar to Quintana et al. [30], we encountered difficulties in achieving satisfactory performance in our setting with homogeneous time constants and ALIF neurons. We attempted this approach because Bittar et al. [4] achieved a significantly higher accuracy (around 90%) on the SHD dataset by incorporating heterogeneous time constant training, utilizing a more complex neuron model, and employing BPTT. Additionally, the limitations associated with the offline nature are not a concern, given that the model is typically not utilized for inference in the source domain. Therefore, exploring if and how source domain training with BPTT might improve performance could still be an interesting direction.

Another avenue worth exploring is to dive into which layers of the network need retraining during the target domain training. In our study, we chose to retrain all layers, but the weight analysis experiment showed that the most significantly modified weights are the ones to the output layer (see Section 5.6). It is plausible that only retraining this layer might suffice, which would make the post-deployment less computationally demanding.

Further, it may be worth exploring how adapting to a target domain affects the performance of the source domain. Namely, a potential risk of the domain adaptation is catastrophic forgetting [18], which, in our case, refers to a decline in performance in the source domain due to target domain training. Therefore, another potential benefit of not retraining all layers during the target domain adaptation is the potential reduction in the risk of catastrophic forgetting. Nevertheless, further exploration is required to study the extent of catastrophic forgetting and possible solutions for it.

Another aspect to investigate is the approach to binning employed to modify the temporal resolution. Currently, we utilize binarizing binning, in which each channel can emit only one spike in each binning window. Despite maintaining a highly efficient binary event data communication, this approach results in some loss of information (see Section 4.1). An alternative method is utilizing graded spikes with sum-binning [26], where spikes are represented as integers, and the spikes are summed in each binning window. This encoding method inherently contains more information. Although using this method has demonstrated improved performance in the study of Orchard et al. [26], it leads to increased computational costs. All in all, employing graded spikes throughout the whole network and utilizing sum-binning on the data would be interesting to explore.

Although this study does not aim to optimize the transfer learning performances for a given setup, it would be valuable to consider doing so. For instance, a potentially fruitful investigation would be examining the seemingly prominent role of overfitting in the worse performance of the increased resolution experiments. Various regularization methods, including dropout and diverse forms of weight regularization that have proven effective for e-prop and deep learning, could be explored within this framework [3, 21].

Lastly, as this study online includes the SHD dataset, it would be interesting to see how our transfer learning scheme would perform on other datasets. Particular of interest would be to see how it performs on data encoded by the Speech2Spikes audio processing pipeline, which can encode spikes for event-based speech recognition systems [32].

In addition to the numerous potential improvements within the framework, there are also interesting avenues beyond it. The first of them is the many other transfer learning methods that could be interesting in further research. For example, the source domain can be extended to multiple source domains [38]. There are several examples where multiple source training leads to good results on speech recognition tasks in the field of deep learning [37, 29]. It would be interesting to investigate how this approach influences the performance of SNNs with varying temporal resolutions across multiple source domains.

An even more advanced approach involves meta-learning. In meta-learning, the network learns how to learn and has already shown promising results in SNN for few-shot learning [31]. Although few-shot learning is about learning new classes, delving into how meta-learning performs for domain adaptation could be interesting to explore nonetheless.

Lastly, in numerous real-life scenarios, labelled data may not be available. There exist several transfer learning methods designed for deep learning that operate without requiring labels in either the target or source domain [27, 38]. Considering practical applications, it is worthwhile to investigate the performance of these methods with SNN.

References

[1] Mohammad Abdollahi and Shih-Chii Liu. "Speaker-independent isolated digit recognition using an aer silicon cochlea". In: *2011 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2011, pages 269–272.

[2] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. "Long short-term memory and learning-to-learn in networks of spiking neurons". In: *Advances in neural information processing systems* 31 (2018).

[3] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. "A solution to the learning dilemma for recurrent networks of spiking neurons". In: *Nature communications* 11.1 (2020), page 3625.

[4] Alexandre Bittar and Philip N. Garner. "A surrogate gradient spiking baseline for speech command recognition". In: *Frontiers in Neuroscience* 16 (2022), page 865897.

[5] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. "SpikeProp: backpropagation for networks of spiking neurons." In: *ESANN*. Volume 48. Bruges. 2000, pages 419–424.

[6] Mohamed Sadek Bouanane, Dalila Cherifi, Elisabetta Chicca, and Lyes Khacef. "Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition". In: *arXiv preprint arXiv:2211.07761* (2022).

[7] Vincent Chan, Shih-Chii Liu, and Andr van Schaik. "AER EAR: A matched silicon cochlea pair with address event representation interface". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1 (2007), pages 48–59.

[8] Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.7 (2022), pages 2744–2757.

[9] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning". In: *IEEE Micro* 38.1 (2018), pages 82–99.

[10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pages 249–256.

[11] Stephen Grossberg. "Competitive learning: From interactive activation to adaptive resonance". In: *Cognitive science* 11.1 (1987), pages 23–63.

[12] Mark Horowitz. "1.1 computing's energy problem (and what we can do about it)". In: *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*. IEEE. 2014, pages 10–14.

[13] Dongsung Huh and Terrence J Sejnowski. "Gradient descent for spiking neural networks". In: *Advances in neural information processing systems* 31 (2018).

[14] Aren Jansen and Partha Niyogi. "Point process models for event-based speech recognition". In: *Speech Communication* 51.12 (2009), pages 1155–1168.

[15] Angel Jiménez-Fernández, Elena Cerezuela-Escudero, Lourdes Miró-Amarante, Manuel Jesus Domınguez-Morales, Francisco de Asıs Gómez-Rodrıguez, Alejandro Linares-Barranco, and Gabriel Jiménez-Moreno. "A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach". In: *IEEE transactions on neural networks and learning systems* 28.4 (2016), pages 804–818.

[16] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. "Spiking-yolo: spiking neural network for energy-efficient object detection". In: *Proceedings of the AAAI conference on artificial intelligence*. Volume 34. 07. 2020, pages 11270–11277.

[17] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, and Agnieszka Grabska-Barwinska. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pages 3521–3526.

[19] Thomas Jacob Koickal, Rhonira Latif, Luiz Gouveia, Enrico Mastropaolo, Shiwei Wang, Alister Hamilton, Rebecca Cheung, Michael Newton, and Leslie Smith. "Design of a spike event coded RGT microphone for neuromorphic auditory systems". In: *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE. 2011, pages 2465–2468.

[20] Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. "Transfer learning for speech recognition on a budget". In: *arXiv preprint arXiv:1706.00290* (2017).

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pages 436–444.

[22] Qianli Liao, Joel Leibo, and Tomaso Poggio. "How important is weight symmetry in backpropagation?" In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 30. 1. 2016.

[23] Wolfgang Maass. "Networks of spiking neurons: the third generation of neural network models". In: *Neural networks* 10.9 (1997), pages 1659–1671.

[24] Misha A. Mahowald and Carver Mead. "The Silicon Retina". In: *Scientific American* 264.5 (1991), pages 76–83.

[25] Arild Nøkland. "Direct feedback alignment provides learning in deep neural networks". In: *Advances in neural information processing systems* 29 (2016).

[26] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. "Efficient neuromorphic signal processing with loihi 2". In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE. 2021, pages 254–259.

[27] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pages 1345–1359.

[28] Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. "Neural heterogeneity promotes robust learning". In: *Nature communications* 12.1 (2021), page 5791.

[29] Chu-Xiong Qin, Dan Qu, and Lian-Hai Zhang. "Towards end-to-end speech recognition with transfer learning". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2018.1 (2018), pages 1–9.

[30] Fernando M Quintana, Fernando Perez-Peña, Pedro L Galindo, Emre O Netfci, Elisabetta Chicca, and Lyes Khacef. "ETLP: Event-based Three-factor Local Plasticity for online learning with neuromorphic hardware". In: *arXiv preprint arXiv:2301.08281* (2023).

[31] Kenneth M Stewart and Emre O Neftci. "Meta-learning spiking neural networks with surrogate gradient descent". In: *Neuromorphic Computing and Engineering* 2.4 (2022), page 044002.

[32] Kenneth Michael Stewart, Timothy Shea, Noah Pacik-Nelson, Eric Gallo, and Andreea Danielescu. "Speech2Spikes: Efficient Audio Encoding Pipeline for Real-time Neuromorphic Systems". In: *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference*. 2023, pages 71–78.

[33] Mohammad-Hassan Tayarani-Najaran and Michael Schmuker. "Event-based sensing and signal processing in the visual, auditory, and olfactory domain: A review". In: *Frontiers in Neural Circuits* 15 (2021), page 610446.

[34] Volodymyr Vasyutynskyy and Klaus Kabitzsch. "Event-based control: Overview and generic model". In: *2010 IEEE International Workshop on Factory Communication Systems Proceedings*. IEEE. 2010, pages 271–279.

[35] Dong Wang and Thomas Fang Zheng. "Transfer learning for speech and language processing". In: *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE. 2015, pages 1225–1237.

[36] Shiwei Wang, Thomas Jacob Koickal, Godwin Enemali, Luiz Gouveia, Lei Wang, and Alister Hamilton. "Design of a silicon cochlea system with biologically faithful response". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2015, pages 1–7.

[37] Zhenyu Wang and John H. L. Hansen. "Multi-Source Domain Adaptation for Text-Independent Forensic Speaker Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), pages 60–75.

[38] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big data* 3.1 (2016), pages 1–40.

[39] Bojian Yin, Federico Corradi, and Sander M Bohté. "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks". In: *Nature Machine Intelligence* 3.10 (2021), pages 905–913.

[40] Qiugang Zhan, Guisong Liu, Xiurui Xie, Guolin Sun, and Huajin Tang. "Effective transfer learning algorithm in spiking neural networks". In: *IEEE Transactions on Cybernetics* 52.12 (2021), pages 13323–13335.

[41] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. "A Comprehensive Survey on Transfer Learning". In: volume 109. 1. 2021, pages 43–76.

A Hyperparameter settings

Table 4: The hyperparameter setting in the different experiments.

	Experiment				
	Temporal resolution sweep	**τ_m sweep**	**β sweep**	**Epochs sweep**	**Weight analysis**
Batch size	128	128	128	128	128
Learning rate	0.0005	0.0005	0.0005	0.0005	0.0005
Epochs source training	100	100	100	sweep	100
Epochs target training	10 or 50	10 or 50	10 or 50	10 or 50	10
Source resolution	10 or 20	10 or 20	10 or 20	10 or 20	10
Target resolution	sweep	20 or 10	20 or 10	20 or 10	20
Refractory period	5.0 ms	5.0 ms	5.0 ms	5.0 ms	5.0 ms
$\boldsymbol{\tau}_m$	100 ms	sweep	100 ms	sweep	100 ms
$\boldsymbol{\tau}_a$	100 ms	100 ms	100 ms	100 ms	100 ms
$\boldsymbol{\tau}_o$	100 ms	100 ms	100 ms	100 ms	100 ms
$\boldsymbol{\beta}$	5.0	5.0	sweep	5.0	5.0

B Tables of results

B.1 Different temporal resolution

Reduced resolution Below, the tables with the accuracies (Table 5) and gains (Table 6) can be found. The highest values per metric are highlighted.

Table 5: The accuracy on the target test set, which is binned with a resolution of **Target resolution**, at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the target domain (**Epoch 1**) and the maximum accuracy during the training in the target domain (**Epoch 10**). Lastly, the accuracy on the source test set, which has a resolution of 10 ms, is displayed in the **Source test** row.

	Target resolution (ms)					
	12.5	**15**	**20**	**25**	**30**	**35**
Epoch 0	**0.736±0.002**	0.579±0.022	0.441±0.023	0.328±0.003	0.334±0.006	0.273±0.017
Epoch 1	**0.761±0.008**	0.719±0.005	0.620±0.031	0.483±0.025	0.431±0.024	0.340±0.023
Epoch 10	**0.812±0.002**	0.747±0.000	0.724±0.003	0.707±0.0009	0.674±0.008	0.599±0.021
Source	0.780	0.780	0.780	0.780	0.780	0.780

Table 6: The gain in accuracy on the target test set, which is binned with a resolution of **Target resolution** after one epoch of training in the target domain (**Epoch 1**) and the maximum gain during the training in the target domain (**Epoch 10**).

	Target resolution (ms)					
	12.5	**15**	**20**	**25**	**30**	**35**
Epoch 1	0.025±0.006	0.140±0.006	**0.179±0.015**	0.155±0.015	0.097±0.026	0.068±0.007
Epoch 10	0.076±0.000	0.168±0.011	0.283±0.026	**0.379±0.011**	0.339±0.002	0.326±0.007

Increased resolution Below are the tables with the accuracies (Table 7) and gains (Table 8) displayed. The highest values per metric are highlighted.

Table 7: The accuracy on the target test set, which is binned with a resolution of **Target resolution (ms)**, at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the target domain (**Epoch 1**) and the final target test accuracy during the training in the target domain (**Epoch 50**). Lastly, the accuracy on the source test set, with a resolution of 20 ms, is displayed in the **Source** row.

	Target resolution (ms)			
	1	**5**	**10**	**15**
Epoch 0	0.165±0.003	0.210±0.003	0.543±0.003	**0.648±0.018**
Epoch 1	0.165±0.008	0.336±0.005	0.585±0.008	**0.668±0.017**
Epoch 50	0.372±0.059	0.553±0.008	0.684±0.003	**0.766±0.018**
Source	0.675	0.675	0.675	0.675

Table 8: The gain in accuracy on the target test set, which is binned with a resolution of **Target resolution (ms)** after one epoch of training in the target domain (**Epoch 1**) and the final target test accuracy during the training in the target domain (**Epoch 50**).

	Target resolution (ms)			
	1	**5**	**10**	**15**
Epoch 1	-0.001±0.011	**0.126±0.008**	0.042±0.005	0.019±0.002
Epoch 50	0.207±0.062	**0.343±0.011**	0.141±0.000	0.117±0.000

B.2 Different membrane time constants

Reduced resolution Table 9 shows the accuracy for the membrane time constant sweep and Table 10 the gains of this experiment. The highest values per metric are highlighted.

Table 9: The accuracy on the target test set, with a resolution of 20 ms, at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the source domain (**Epoch 1**) and the maximum during the training in the source domain (**Epoch 10**), for the different membrane time constants. Lastly, the accuracy on the source test set, with a resolution of 10 ms, is displayed in the **Source test** row.

	Membrane time constant (ms)				
	50	**100**	**250**	**400**	**600**
Epoch 0	0.419±0.011	0.458±0.009	0.484±0.009	**0.559±0.045**	0.361±0.023
Epoch 1	0.578±0.000	**0.605±0.029**	0.573±0.109	0.439±0.045	0.454±0.019
Epoch 10	0.667±0.023	**0.730±0.007**	**0.730±0.022**	0.675±0.052	0.607±0.024
Source test	0.710±0.013	**0.775±0.004**	0.743±0.003	0.624±0.102	0.554±0.041

Table 10: The gain in accuracy on the target test set after one epoch of training in the target domain (**Epoch 1**) and final gain during the training in the target domain (**Epoch 10**), for the different membrane time constants.

	Membrane time constant (ms)				
	50	**100**	**250**	**400**	**600**
Epoch 1	0.134±0.017	0.190±0.019	0.147±0.003	0.083±0.012	**0.191±0.038**
Epoch 10	0.247±0.034	**0.273±0.007**	**0.246±0.013**	0.116±0.019	0.247±0.001

Increased resolution Below, the tables with the accuracies (Table 11) and gains (Table 12) can be found. The highest values per metric are highlighted.

Table 11: The accuracy on the target test set, with a resolution of 10 ms, at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the target domain (**Epoch 1**) and final target test accuracy obtained during the target domain training (**Epoch 50**), for the different membrane time constants. Lastly, the accuracy on the source test set with a 20 ms resolution is displayed in the **Source test**.

	Membrane time constant (ms)				
	50	**100**	**250**	**400**	**600**
Epoch 0	0.393±0.000	0.539±0.035	**0.559±0.123**	0.346±0.048	0.434±0.051
Epoch 1	0.553±0.010	**0.648±0.019**	0.631±0.012	0.642±0.047	0.551±0.059
Epoch 10	0.664±0.000	**0.678±0.009**	0.596±0.057	0.584±0.048	0.619±0.016
Source test	0.652±0.000	**0.673±0.003**	0.624±0.106	0.515±0.017	0.532±0.033

Table 12: The gain in accuracy of the target test set, after one epoch of training in the target domain (**Epoch 1**) and at the end of the training in the target domain (**Epoch 50**), for the different membrane time constants.

	Membrane time constant (ms)				
	50	**100**	**250**	**400**	**600**
Epoch 1	**0.185±0.000**	0.066±0.010	0.014±0.017	0.093±0.004	0.019±0.069
Epoch 10	**0.271±0.000**	0.139±0.026	0.037±0.066	0.238±0.033	0.185±0.060

B.3 Different threshold factors

Reduced resolution Table 13 shows the accuracy for the threshold factor sweep and Table 14 the gain in accuracy of this experiment. The highest values per metric are highlighted.

Table 13: The accuracy on the target test set with a 20 ms at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the target domain (**Epoch 1**) and the final accuracy (**Epoch 10**), for the different threshold factors. Lastly, the accuracy on the source test set, where the resolution is 10 ms, is displayed in the **Source** row.

	Threshold factor				
	0	**2**	**4**	**6**	**8**
Epoch 0	0.414±0.027	**0.477±0.005**	0.426±0.006	0.416±0.024	0.414±0.000
Epoch 1	0.470±0.069	**0.657±0.033**	0.624±0.011	0.621±0.002	0.585±0.015
Epoch 10	0.500±0.095	0.700±0.026	**0.733±0.006**	0.720±0.012	0.690±0.009
Source	0.454±0.086	**0.770±0.015**	0.768±0.005	0.751±0.006	0.697±0.022

Table 14: The gain in accuracy on the target test set obtained in the first epoch of training in the target domain (**Epoch 1**) and the maximum gain obtained during the training in the target domain (**Epoch 10**), for the different threshold factors.

	Threshold factor				
	0	**2**	**4**	**6**	**8**
Epoch 1	0.056±0.042	0.181±0.029	0.198±0.005	**0.205±0.026**	0.171±0.015
Epoch 10	0.086±0.068	0.223±0.021	0.306±0.012	0.305±0.036	**0.376±0.009**

Increased resolution Below are the tables with the accuracies (Table 15) and gains (Table 16) displayed. The highest values per metric are highlighted.

Table 15: The accuracy on the target test set, where the resolution is 20 ms, at the end of the training in the source domain (**Epoch 0**), after one epoch of training in the target domain (**Epoch 1**) and the final accuracy (**Epoch 50**), for the different threshold factors. Lastly, the accuracy on the source test set (10 ms resolution) is displayed in the **Source** row.

	Threshold factor				
	0	**2**	**4**	**6**	**8**
Epoch 0	0.365±0.016	0.549±0.000	**0.552±0.023**	0.459±0.066	0.449±0.011
Epoch 1	0.456±0.020	0.599±0.000	**0.613±0.012**	0.532±0.020	0.549±0.003
Epoch 50	0.543±0.008	**0.719±0.000**	0.680±0.011	0.658±0.003	0.641±0.020
Source	0.519±0.086	0.661±0.015	**0.663±0.005**	0.648±0.006	0.639±0.022

Table 16: The gain in accuracy on the target test set obtained in the first epoch of training in the target domain (**Epoch 1**) and the maximum gain obtained during the training in the target domain (**Epoch 50**), for the different threshold factors.

	Threshold factor				
	0	**2**	**4**	**6**	**8**
Epoch 1	0.092±0.035	0.049±0.000	0.061±0.034	0.073±0.086	**0.100±0.008**
Epoch 50	0.178±0.023	0.169±0.000	0.128±0.021	**0.199±0.069**	0.193±0.009

B.4 Different number of epochs of source domain training

Reduced resolution Table 13 shows the accuracy for the different number of epochs of source domain training per membrane potential time constants τ_m-value. The highest accuracy per τ_m value is highlighted.

Table 17: The final accuracy on the target test set with a 20 ms resolution (**Epoch 10**) for the different number of epochs source domain training (**Epochs training**) and different membrane potential time constants ($\boldsymbol{\tau_m}$**(ms)**) .

$\boldsymbol{\tau_m}$**(ms)**	**Epochs training**	**Epoch 10**
100	40	0.547±0.026
	70	0.661±0.045
	100	**0.705±0.047**
150	40	0.583±0.007
	70	0.696±0.004
	100	**0.717±0.001**
200	40	0.604±0.014
	70	0.679±0.014
	100	**0.714±0.006**
250	40	0.607±0.000
	70	0.659±0.000
	100	**0.683±0.000**
$\boldsymbol{\infty}$	40	0.520±0.014
	70	0.527±0.024
	100	**0.618±0.005**

Increased resolution Table 15 shows the accuracy for the different number of epochs of source domain training per membrane potential time constants τ_m-value. The highest accuracy per τ_m value is highlighted.

Table 18: The final accuracy on the target test set, where the resolution is 10 ms (**Epoch 50**) for the different number of epochs source domain training (**Epochs training**) and different membrane potential time constants ($\boldsymbol{\tau_m}$**(ms)**) .

$\boldsymbol{\tau_m}$**(ms)**	**Epochs training**	**Epoch 50**
100	40	0.634±0.001
	70	0.678±0.006
	100	**0.689±0.009**
150	40	0.614±0.020
	70	0.649±0.017
	100	**0.668±0.008**
200	40	0.598±0.015
	70	0.614±0.024
	100	**0.727±0.006**
250	40	0.626±0.010
	70	**0.654±0.012**
	100	0.594±0.030
$\boldsymbol{\infty}$	40	0.588±0.046
	70	0.527±0.049
	100	**0.589±0.032**