Combining Deep Convolutional Feature Extraction with Hyperdimensional Computing for Visual Object Recognition

Piotr Łuczak
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland
pluczak@iis.p.lodz.pl

Krzysztof Ślot
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland
kslot@p.lodz.pl

Jacek Kucharski
Institute of Applied Computer Science
Lodz University of Technology
Łódź, Poland
jkuchars@iis.p.lodz.pl

***Abstract*—The presented paper proposes a novel, hybrid neuromorphic computational architecture for visual data classification aimed at implementation in energy-efficient application-specific, FPGA or ASIC-based edge computing devices. The architecture combines a convolutional neural extractor that produces comprehensive representations of input patterns with a Hyperdimensional Computing (HDC) module that enables complex data analyses, including vector and vector sequence classification. As the biologically inspired HDC paradigm operates on holistic representations of concepts, we accordingly design a convolutional extractor to summarize various aspects of objects' appearance. As low energy consumption is the key design constraint, we assume that input images are delivered by energy-efficient dynamic vision sensors (event cameras). The extractor is pretrained using a three-head Convolutional Neural Network (CNN). The different CNN heads: classifier, decoder, and clusterer implement optimization objectives essential for the "holographic" concept representation. Feature vectors produced by the extractor are projected onto hyperdimensional binary vectors using an encoding unit, and they are subject to classification in the HDC module. The neural extractor is trained in limited precision mode to account for ASIC/FPGA hardware constraints. We apply the proposed architecture to classify objects (pedestrians, cars, and cyclists) from two different traffic datasets: VIRAT and KAIST. We show that the proposed concept enables solving classification problems with an accuracy that matches the performance of deep neural classifiers while being feasible for implementation in energy-efficient application-specific hardware.**

***Index Terms*—neuromorphic architectures, convolutional neural networks, hyperdimensional computing, representation extraction.**

I. Introduction

Intelligent sensing devices, intended to locally solve complex, real-world problems at the periphery of large computing systems, are of growing importance [1]. There are multiple reasons for striving to solve tasks autonomously. Local execution of tasks can be done in real-time without unexpected delays, which may be of critical importance. By processing data locally, one can not only avoid generating heavy network traffic and streaming large amounts of data but also operate in environments where the network connection could break or is not available. Among the essential requirements for the usability of autonomous intelligent systems is the low energy consumption, which determines the longevity of their operation.

To enable intelligent analysis of visual data in compact, energy-efficient hardware, one must adapt existing problem-solving methodologies to severe complexity constraints imposed by target computational environments. Unfortunately, the most successful visual scene analysis algorithms (visual object classification [2] or detection [3], visual localization [4], prediction [5], or action recognition [6]) are based on deep convolutional neural networks. These algorithms require significant computational resources to accommodate complex architectures, making their implementation in compact, energy-efficient edge computing devices (such as FPGAs or custom, Application-Specific Integrated Circuits - ASICs) infeasible. Therefore, simplifying existing data analysis architectures while maintaining their high performance is an important research direction that could enable intelligent edge computing, increase processing efficiency, and ensure better explainability of underlying AI algorithms [7], [8].

An attractive candidate for performing complex data analyses, including data classification, prediction, or inference, which recently gained popularity, is the concept of *Hyperdimensional Computing* (HDC) [9]. HDC, inspired by findings in neurophysiology, lays down a hardware-friendly framework for performing high-level data analysis. HDC has proved successful in a variety of complex tasks, including clustering and recognizing the language of written texts [10], sensorimotor control [11], visual localization [12] or speech recognition [13]. The remarkable feature of all of these solutions is the simplicity of data processing that is necessary to do the task: the bulk of operations are carried out on binary variables using simple, spatially local operations. Hyperdimensional computing is, therefore, ideally suited to in-memory com-

This work was funded by European Union's Horizon 2020 research and innovation programme under grant agreement no 101016734. This work has been completed while the 1st author was the Doctoral Candidate in the Interdisciplinary Doctoral School at the Lodz University of Technology, Poland. Software from https://comet.ml accelerated this work. Our code is available at https://github.com/TUL-IIS-MISEL/CNN-HDC.

puting, which outperforms the conventional microprocessor architectures both in terms of energy efficiency (as no costly data transfers to and from CPU/GPU are required) and speed (due to computational locality) [14]–[16].

This paper proposes a novel computational architecture that integrates these two powerful data processing paradigms to enable energy-efficient implementation of intelligent data analysis in edge computing devices. The hybrid architecture, where the bulk of the operations can be executed asynchronously, combines the representation-extracting convolutional neural network (henceforth referred to as *Extractor*) with a Hyperdimensional Computing processor (*HDC Processor*). We propose to train the Extractor offline in such a way that it produces rich representations of the input data, thus complying with the "holographic" nature of HDC. Three different objectives drive Extractor's weights learning via a three-head deep neural network to accomplish this goal. The first training objective, enforced by the *classification head*, injects Extractor-generated latent vectors with information on interclass differences. The second objective, imposed by the *clustering head*, seeks characteristic structures in input patterns and encourages latent space vector grouping (this would enable compositional object representation, suggested to be an important attribute of cognitive information processing by [17] and [18]). Finally, the last objective, enforced by the *decoding head* (that together with the Extractor forms an Autoencoder), attempts to preserve all important information on objects' appearance. The derived composite latent space representations of input patterns are mapped onto *Hyperdimensional Binary Vectors* (HBV) in the encoding module (*Encoder*), which interfaces the Extractor and HDC Processor.

To validate the proposed concept, we consider two visual object classification problems: the simpler one is to discriminate between cars and pedestrians in the recordings provided by the VIRAT dataset [19], whereas the more challenging one is a three-category problem of discriminating among cyclists, persons, and groups of persons in recordings from KAIST database [20]. As our main objective is to develop an energy-efficient computational framework, instead of considering standard visual sensors, we assume that input data to be analyzed is produced by energy-efficient dynamic vision sensors, a.k.a. event-cameras [21]. It follows that the Extractor operates on differential information that corresponds to motion patterns, which is typically encoded using ternary data ("+1" or "-1" encode increments or decrements in pixels' intensities and "0" means no change). Although this encoding reduces the amount of input information, it increases the robustness of processing against noise. We show that using the proposed, hardware-friendly approach, one can achieve high recognition accuracy, which surpasses the performance of deep neural networks of comparable complexity.

We begin the remaining presentation with a brief discussion of the related work, which focuses on highlighting relevant concepts of hyperdimensional computing. The proposed computational architecture, along with an explanation of the adopted data processing pipeline, is provided in Section III. Description of the datasets used for experimental evaluations of the proposed concept, together with details on data preprocessing are presented in Section IV. Finally, the results of the proposed hybrid architecture evaluation are summarized in Section V.

II. Related work

Hyperdimensional computing, which is a high-level information processing framework inspired by findings in neurophysiology and formulated in the seminal work by Pentti Kanerva [9], assumes that concepts are represented using random, hyperdimensional binary vectors (HBVs). The two simple operations at the core of HDC-based concept processing are binding and bundling. Binding provides a means for combining two hyperdimensional vectors, e.g., to form a value - label pair, while bundling enables the combination of a number of hyperdimensional values into a single aggregate that resembles its constituents. It has been shown that supplementing these operations with distance-based similarity assessment of HBVs, enables executing a variety of complex data analysis and sequence analysis tasks, including data retrieval, data classification [22], as well as prediction and reasoning [23]. As bundling and binding require only binary, local operators, the HDC concept poses very low requirements on the underlying computational hardware and is suitable for in-memory computing [16], enabling development of intelligent, and energy-efficient edge computing devices.

To perform HDC-based data analysis, one needs to map the information extracted from the supplied raw data, such as images or image sequences, onto HBVs. The procedure involves two operations: the objective of the first one is to encode the input data descriptors (features) using random hyperdimensional binary vectors, whereas the second phase is concerned with aggregating the obtained HBVs ($\boldsymbol{h}_j$) into a single, composite HBV ($\boldsymbol{H}_j$), thus providing a holistic encoding of the input contents. The former operation can be realized in a variety of ways - individual features can be bound to corresponding HBVs using e.g. encoding scheme offered by Sparse Distributed Memory (SDM) [24] or by other, simpler approaches (e.g. by element-wise products of feature-specific HBVs and feature values encoded using the thermometric code [25]). In addition, high-dimensional feature vectors extracted from input data (such as, for example, NetVLAD image descriptor) have also been considered as hyperdimensional representations, without any further encoding, as it was shown in [12]. The two methods that are commonly used for producing holistic concept-wise HBVs [23] are record-based encoding and N-gram-based encoding. The former strategy assumes a "role-filler" encoding scenario, where individual HBVs that encode particular features are bundled together via element-wise majority voting. The latter strategy applies to sequence encoding, where HBVs derived for subsequent sequence components, are accumulated using a rotate-and-bind scheme [23].

HDC data analysis algorithm is set up by deriving concept-wise HBVs that form "holographic" representations of target

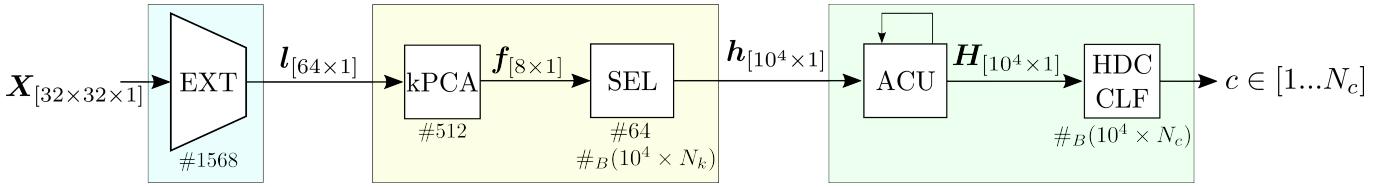


Fig. 1: The proposed hybrid architecture: the Extractor EXT (highlighted in blue), the Encoder comprising kPCA block and HBV selector SEL (in yellow) and the HDC Processor, composed of sequence accumulating module ACU and the classifier HDC CLF (in green). N_k denotes the number of clusters (and also the size of of atomic HBVs vocabulary) , N_c denotes the number of classes, $\#$ - the number of model floating-point parameters, $\#_B$ - number of binary parameters.

concepts (for example, categories considered in some particular recognition task). Once the algorithm has been "trained", the analysis begins with HBV-encoding of the information extracted from the input data to be analyzed. Next, the derived "challenge" HBV is confronted with all concept-wise HBVs. A concept represented by a HBV which is the most similar to the challenge HBV is considered to be an outcome of the analysis, where hyperdimensional vector similarity is assessed e.g. by the Hamming distance.

III. Methods

The proposed architecture, depicted schematically in Fig.1, comprises three different functional components: the convolutional Extractor, followed by the Encoder (a hypervector encoding unit) and the HDC Processor (a hyperdimensional processing module). The pretrained Extractor transforms images ($\boldsymbol{X}$) produced by an event-camera onto latent vectors ($\boldsymbol{l}$), which are mapped onto hypervectors ($\boldsymbol{h}$) by the Encoder. A sequence of hypervectors corresponding to subsequent input patterns are then appropriately combined and classified in the HDC Processor, producing the predicted class labels (c).

A. Extractor derivation

The Extractor is trained to provide a multi-aspect, holistic representation of the input image contents. We assume that the images delivered by an event camera are of size 32-by-32 pixels and each of Extractor's four convolutional layers comprises eight 3-by-3 filters. To train the Extractor we simultaneously apply three different criteria that drive learning by means of three different network's heads, as shown in Fig.2.

The classification head is trained in a supervised manner based on labeled dataset examples. A loss function component that is associated with a classifier-generated output is a commonly used categorical cross-entropy: $L_{CCE}(\boldsymbol{y_c}, \boldsymbol{c})$, where $\boldsymbol{c} \in \mathbb{R}^{N_c}$ denotes a label vector, N_c is a number of classes and $\boldsymbol{y_c} \in \mathbb{R}^{N_c}$ denotes a vector of actual classifier outputs.

The clustering head is based on the DeepCluster concept [26] and it aims to arrange latent space samples into a set of distinctive clusters that are expected to emphasize the compositional structure of input patterns. To adapt the method to our specific objectives, we introduced a few modifications to the original DeepCluster concept. The most important one is replacement of the linear PCA transformation, which was used for reducing latent space dimensionality, with the nonlinear kernel PCA [27], where we adopted the cosine kernel function. Adoption of the more powerful, nonlinear transformation enabled good clustering of input patterns in a compact space (with as few as 8-dimensions, instead of 256 as reported in [26]. Additionally, as the chosen ternary input patterns are much simpler than the originally considered real image patterns, we significantly reduced the target number of clusters (from $\sim 10^4$ to just ~ 10). The loss function component that is associated with the clustering result is also the categorical cross-entropy: $L_{CCE}(\boldsymbol{y_k}, \boldsymbol{k})$, where $\boldsymbol{k} \in \mathbb{R}^{N_k}$ denotes a "pseudo-label" vector (see [26] for details) of length corresponding to the assumed number of clusters N_k, and $\boldsymbol{y_k}$ denotes a vector of actual clusterer's outputs.

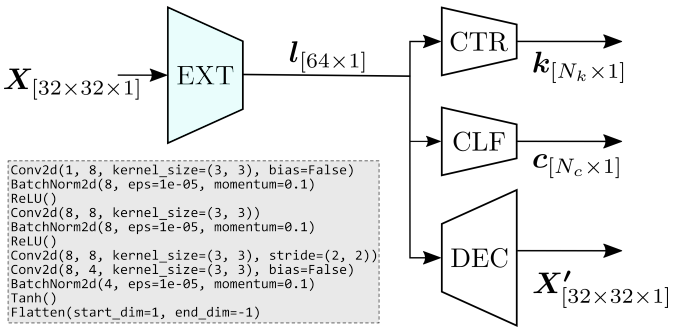


Fig. 2: The neural architecture used for Extractor training (of architecture summarized within the gray box) with clustering (CTR), classification (CLF) and decoding (DEC) heads. N_k and N_c are the number of clusters and classes, respectively.

The role of the last, decoding head is to enforce incorporation into the latent representation, all information useful in reconstructing input concepts, i.e., information summarizing the input patterns' appearance. The loss function component that is associated with the decoding head is the L1 norm: $L_{L1}(\boldsymbol{X}, \boldsymbol{X'})$ evaluated for input image $\boldsymbol{X}$ and the reconstruction result $\boldsymbol{X'}$.

Training of the Extractor is therefore driven by a loss function of the form:

$$\begin{aligned} L = \lambda_1 L_q(\boldsymbol{l}) + \lambda_2 L_{CCE}(\boldsymbol{y_k}, \boldsymbol{k}) \\ + \lambda_3 L_{CCE}(\boldsymbol{y_c}, \boldsymbol{c}) + \lambda_4 L_{L1}(\boldsymbol{X}, \boldsymbol{X'}) \end{aligned} \tag{1}$$

where: $L_{CCE}(\boldsymbol{y_c}, \boldsymbol{c}), L_{CCE}(\boldsymbol{y_k}, \boldsymbol{k}), L_{L1}(\boldsymbol{X}, \boldsymbol{X'})$ are the aforementioned loss components produced by the three considered heads and $\boldsymbol{\lambda} \in \mathbb{R}^4$ is the weight hyperparameter vector.

The fourth component of the loss function - $L_q(\boldsymbol{l})$ is introduced to enforce "soft quantization" of the latent space. The loss function component has the form:

$$L_q(\boldsymbol{l}) = 1 + cos((2^{m+1} - 1)\pi \boldsymbol{l}) \quad (2)$$

and its objective is to reduce the resolution of numbers that can be produced at the Extractor's output to m-bit signed values. This way, we can reduce the data processing sensitivity to reduced number precision of FPGA-based implementations or inevitable inaccuracies of ASIC-based hardware realization.

To determine the impact of each head on the final system performance, the model was trained and evaluated not only in the considered three-head version but also in all possible two-head and single-head configurations.

B. Latent vectors encoding

The objective of the Extractor training was to produce a latent space that captures a holistic description of input patterns and, additionally, arranges the samples into distinguishable clusters. Distinct locations in this space correspond to different patterns, so they are likely to represent distinct "concepts". Therefore, the tessellation' of the space, induced by a latent vector distribution, would produce concept-specific regions and each of these regions could be bound to a separate hypervector. The proposed method adopted for interfacing the Extractor and the HDC Processor implements this idea (see Fig.3). However, following the strategy used by the clustering head during Extractor training, an intermediate step of a kernel-PCA transformation is also applied. Kernel-PCA transforms the latent space to a low-dimensional *feature space* (we use the same architecture as during the training). This space gets split into N_k regions determined by locations of centers of N_k clusters, formed by the projected training latent vectors, and each region is assigned a unique random HBV.

Each input image is thus, converted by the Extractor to the latent space and then to the feature space. The resulting feature vector is matched to the closest of N_k cluster centers. The Encoder outputs an HBV that is bound to the selected cluster center. We assume that all hypervectors ($\boldsymbol{h}_1 ... \boldsymbol{h}_{N_k}$) comprise of 10^4 elements (the commonly adopted value).

C. Hyperdimensional computing module

The HDC Processor is composed of two functional blocks. The first one (the ACU block in Fig.1) accumulates a sequence of hypervectors ($\boldsymbol{h}_1 ... \boldsymbol{h}_{N_s}$) corresponding to N_s-subsequent input patterns into a single, composite HBV ($\boldsymbol{H}$) that represents the whole sequence. Encoding of the sequence of HBVs onto the composite hypervector is accomplished using the permute-and-sum approach [24]:

$$\boldsymbol{H}_j := \Pi \boldsymbol{H}_{j-1} + \boldsymbol{h}_j \quad \text{for } j \in [1...N_s]; \quad \boldsymbol{H}_0 = \mathbf{0}_{10^4 \times 1} \quad (3)$$

where Π denotes the permutation operator, $\boldsymbol{h}_j$ is a hypervector assigned to a j-th input image, $\boldsymbol{H}_j$ denotes accumulation result at the step j, and N_s is the sequence length.

As the input HBVs are snapshots of some object's motion sequence, the resulting hypervector $\boldsymbol{H}$ summarizes the way this motion evolves. Object classification, which is performed in the second block of the HDC Processor (HDC CLF), is done by assessing proximity of the currently observed motion pattern p, encoded by $\boldsymbol{H}^p$, to a set of HBVs that correspond to the motion patterns derived for the considered classes. As the hypervectors are binary, selection of the winning class can be efficiently done by means of Hamming distance calculation.

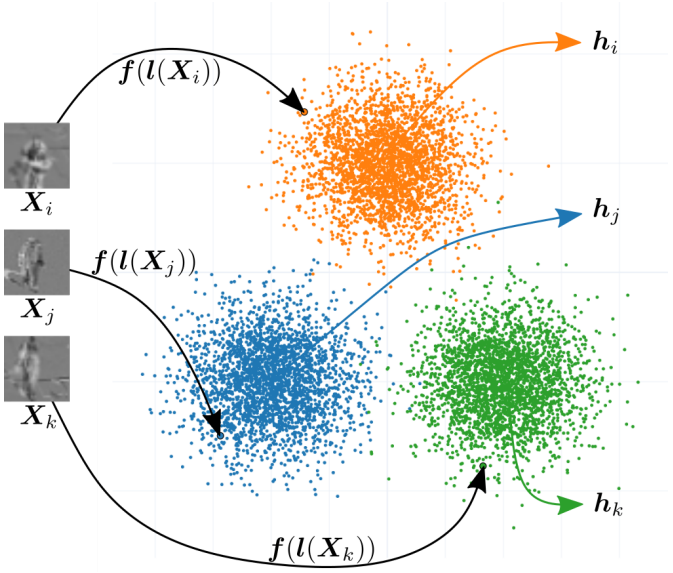


Fig. 3: Mapping of input patterns ($\boldsymbol{X_i}, \boldsymbol{X_j}, \boldsymbol{X_k}$) onto hypervectors ($\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{h}_k$) assigned to different regions of feature space.

Observe that the proposed method for image object classification fits the framework proposed in [9] for language recognition by means of the HDC paradigm. Hypervectors that are assigned to each cluster by the Encoder can be seen as an N_k-element alphabet of symbols (letters) and motion pattern identification can be seen as a detection of symbol sequences that are language-specific.

We assume that the process of building composite vectors that represent classes (class prototypes), as it was the case for Extractor derivation, is done offline. Therefore, the trained hybrid architecture is preset with both Extractor and Encoder parameters, as well as with all necessary HBVs.

D. Reference architecture

To ensure the reference for performance comparison purposes, a convolutional neural network, comprising both a convolutional feature extractor and a set of fully connected layers, has been used (Fig.4). We decided to use the same convolutional part (the Extractor) and design the dense part to have roughly the same number of parameters as has been used for implementing the Encoder and the HDC Processor. To provide information on complexity of the proposed architecture, all of its functional blocks in Fig.1 are assigned with values that denote the number of their parameters. As can be seen in Fig.1, a total of only 2144 floating-point parameters are necessary to implement the Extractor and the Encoder. In

addition, one also needs to store two sets of hypervectors (the first comprises alphabet symbols, whereas the other - HBVs that correspond to class prototypes). Despite the considerable length of hypervectors, they are binary, so they do not pose severe storage requirements. For the alphabet length N_k and the number of classes N_c that were considered throughout the experiments, assuming two-byte representation of floating-point values, Encoder and HDC Processor use approximately the equivalent to 6250 parameters.

To enable a fair comparison, we assume that the reference architecture operates not on a single latent vector but on a set of N_s latent vectors that correspond to s subsequent input images (as the proposed hybrid architecture analyses s-element motion patterns). This means that the input to MLP is a two-dimensional stack of s latent vectors. Exact count of reference architecture parameters has been presented in Fig.4, where different values for the MLP head correspond to two exemplary sequence lengths ($s = 3$ and $s = 8$).

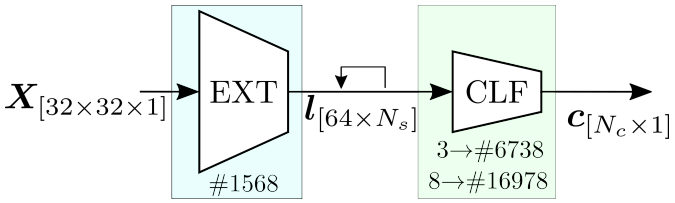


Fig. 4: Reference architecture: EXT - convolutional extractor module, CLF - neural classifier, # - number of model parameters.

E. Metrics

Performance of the considered classifiers was measured using the adjusted balanced accuracy score [28], which eliminates class imbalance effects. The balanced accuracy score A_B is defined as:

$$A_B = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}\right) \tag{4}$$

where TP, TN, FP and FN denote True Positive, True Negative, False Positive and False negative, respectively. The adjusted balanced accuracy score is given by:

$$A_{AB} = \frac{A_B - \frac{1}{1-N_c}}{1 - \frac{1}{1-N_c}} \tag{5}$$

The value of zero can be interpreted as a random performance, while the value of one would describe a "perfect" classifier. For a problem with only two classes, this measure can be considered as equivalent to J statistics (also known as J score) proposed by [29].

IV. Datasets

As it has been pointed out, to maintain energy efficiency, input images are assumed to be produced by event cameras, rather than by commonly used synchronous visible-spectrum cameras [30], [31]. This simplifies the image preprocessing procedure: since stationary event cameras ignore static objects, region of interest detection can be done by simple search for blobs, rather than by executing complex pattern analysis procedures necessary for grayscale content analysis [21]. Additionally, due to the extremely short response time of asynchronous event cameras, even rapidly moving objects can be observed without blurring or other artifacts caused by undersampling.

Due to the relative novelty of dynamic vision sensors, there exist only a few publicly available datasets acquired using event cameras and the majority of these datasets were acquired using a moving camera, such as e.g. Prophesee gen1 produced by [32]. Unfortunately, moving cameras introduce the problem of egomotion, which needs to be compensated using complex algorithms, thus, eliminating the aforementioned object detection simplicity.

An alternative source of event-based data is to extract them from conventional video datasets (this applies to both visual and infrared recordings), using, for example, the method proposed by [33]. This approach has been adopted to gather the experimental material used in evaluation of the proposed concept. The two datasets we use: VIRAT and KAIST are concerned with traffic monitoring and provide material for the detection of vehicles and vulnerable road users (VRUs), such as pedestrians or cyclists (note that the problem of pedestrian detection has been the subject of extensive research for over a decade [34]).

A. VIRAT

VIRAT Ground Video 2.0 dataset created by [19], consists of 25 hours of HD video shot at 30 FPS by stationary cameras in 11 different locations. It was intended to be used in visual event recognition tasks and contains bounding box annotations for the objects identified as "people", "cars", "bikes", "vehicles" or (other) "objects".

Due to their high prevalence, only objects belonging to "people" and "cars" classes were selected for our first set of experiments. Video frames were transformed into event-like frames by first converting them into greyscale images and then by computing the differences using a four-frame interval. The resulting differential images were subsequently converted to ternary ones by thresholding (to indicate the locations of strong "positive" and "negative" intensity changes incurred by motion). For the purpose of region of interest extraction, the ternary images are converted to the binary form (by merging the "negative" and "positive" classes). Finally, a set of morphological operations intended to clean up noise was executed, followed by the blob detection procedure, which indicated the target regions of interest. These regions were subsequently rescaled to a fixed size (32x32) and labeled based on the bounding box labels provided in the dataset.

B. KAIST

The dataset produced by [20] consists of 95 000 VGA video frames acquired using a combination of RGB and thermal cameras at 20 FPS. The dataset contains annotations for

three classes: “person”, “people” and “cyclist” and covers 11 different scenarios filmed in different locations and at different times of a day. The acquisition was carried out using two cameras (one for each spectrum) mounted on a moving car. A notable feature of this dataset is the severe class imbalance as samples of the class “person” make up almost 85% of all annotated objects.

Due to egomotion, KAIST dataset could not be converted to event-camera form using the same method as in the case of VIRAT, so the approach proposed by [33] was used instead and regions of interest were identified based on the provided bounding boxes.

V. Results

The main objective of the experiments was to assess the object classification performance offered by the proposed hybrid computing concept and compare it with performance of the adopted reference method. For both datasets, we experimentally selected the optimum number of feature space clusters (to get a reasonable compromise between performance and complexity): $N_k = 8$ for the VIRAT dataset and $N_k = 16$ for KAIST. For all experiments, performance of the reference architecture is calculated as the best value achieved over 10 experiment runs.

A. VIRAT

When faced with the simpler problem of distinguishing between “people” and “cars”, the reference algorithm managed to achieve J-scores of 0.76 (for processing of three-element sequences) and 0.80 (for eight-element sequences), with a comparable number of type 1 (False Positive) and type 2 (False Negative) errors. Performance of the proposed method has been summarized in table I and Fig.5.

TABLE I: Hybrid classifier performance on VIRAT

Model heads	N_s	AUC	J score
CLF	3	0.87± 0.03	0.58± 0.03
	8	0.92± 0.03	0.68± 0.03
CTR	3	0.90± 0.01	0.66± 0.01
	8	0.94± 0.02	0.73± 0.02
DEC	3	0.50± 0.00	0.00± 0.00
	8	0.50± 0.00	0.00± 0.00
CLF+CTR	3	0.91± 0.01	0.67± 0.01
	8	0.96± 0.01	0.77± 0.01
CLF+DEC	3	0.93± 0.02	0.70± 0.02
	8	0.96± 0.02	0.79± 0.02
CTR+DEC	3	0.89± 0.02	0.62± 0.02
	8	0.92± 0.01	0.69± 0.01
CLF+CTR+DEC	3	0.93± 0.02	0.72± 0.02
	8	**0.96± 0.01**	**0.81± 0.01**

As it can be observed, the proposed approach achieves the best performance when all three heads are applied during training. Over the course of several runs, the proposed method is capable of repeatedly reaching 0.81 J-score, with a standard deviation of only 0.01. The HDC classifier performs better when trained using longer sequences, which reveal more object-specific information on its motion pattern. As could be expected for a balanced dataset, classification in the hyperdimensional space produces a comparable number of type 1 and type 2 errors.

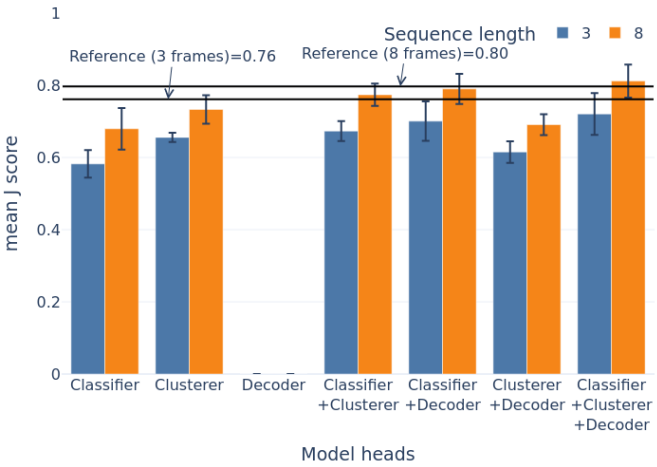


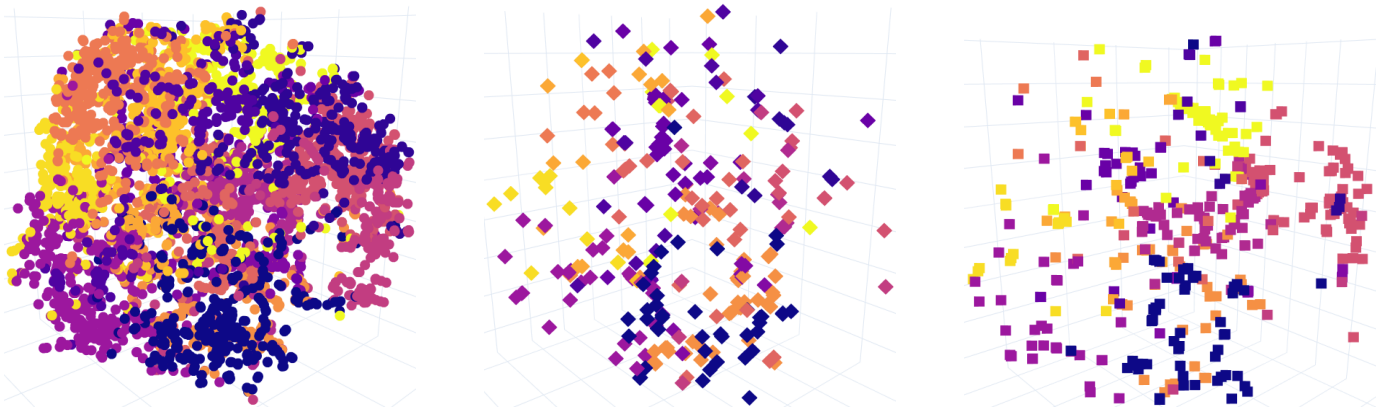
Fig. 5: Hybrid classifier performance on VIRAT

B. KAIST

When faced with the more complex problem of distinguishing between highly imbalanced sets of “person”, “people” and “cyclist”, the reference was not able to get any better than random-guess performance neither on visible nor on thermal spectra, severely overfitting to the most populous class. It is noteworthy that feature space clustering produced evenly sized clusters (each symbol contains 0.06±0.02 of the dataset samples) with the average GINI cluster impurity of 0.34. This implies that a single symbol could not be used to correctly predict the label. Clusters formed in a feature space and the distribution of class labels among a random 500-element sample from the dataset have been shown in Fig.6.

The proposed hybrid classifier was capable of achieving only slightly better than random performance on data from the visible spectrum. However, in contrast to the visible spectrum, the classifier trained on thermal images performed significantly better - thermal data are more resilient to egomotion noise, as inanimate objects typically exhibit very limited IR emission.

As it can be seen from Fig.7, which summarizes the classification results for the proposed hybrid architecture, where the *Extractor* was trained using different two-headed and three-headed configurations, a few observations can be made. The first one is a clear increase in label prediction accuracy for longer sequences considered in the hyperdimensional analysis. One needs to note that an increase in sequence length does not imply any modification of the computational architecture, so any hardware implementation of the algorithm can easily be tuned to handle the analysis of variable-length sequences. Comparing the classification accuracy for different extractor training scenarios, there is no clear advantage offered by the



(a) Images with “person” label (b) Images with “people” label (c) Images with “cyclist” label

Fig. 6: Symbol assignment in t-SNE space for a small subset of the dataset. Samples of each class are spread over all available symbols. The drastic disproportion of samples between classes can be clearly seen.

three-headed configuration, although one of the reasons for this to happen is the relatively small amount of training data, which might cause problems with training of the most complex, architecture.

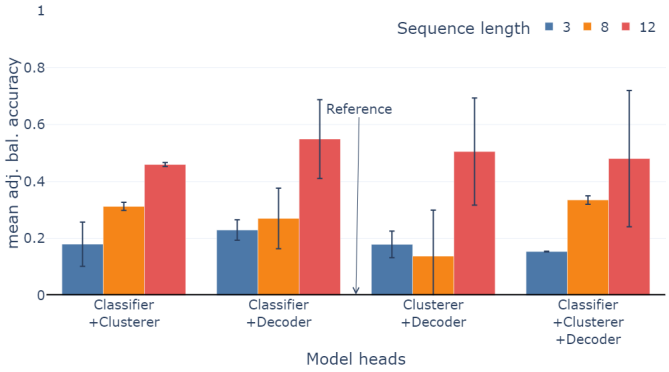
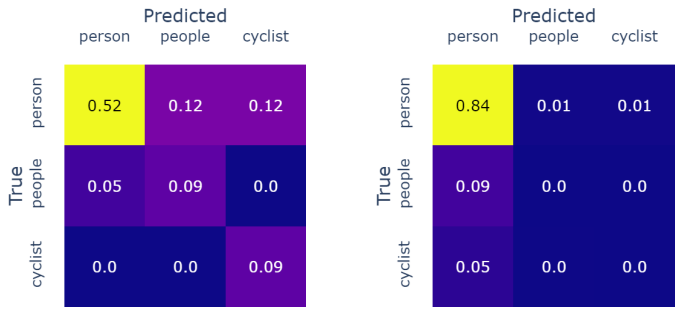


Fig. 7: Performance of the hybrid classifier on thermal images from the KAIST Dataset



(a) HDC classifier (b) Reference model

Fig. 8: Confusion matrices for the proposed hybrid and reference models.

Confusion matrices for data classification results by means of the proposed hybrid classifier and by means of the reference neural network are presented in Fig.8. As it can be seen, the neural classifier overfits in favor of the most populous class. The proposed hybrid architecture produces no false negatives for the “cyclist” class and mistakes “people” only with the highly similar “person” class.

VI. Conclusion

The main contribution of the presented paper is the proposal of a novel, hybrid computational architecture that combines a convolutional neural representation extractor with a Hyperdimensional Computing module. The proposed mixture of the state-of-the-art method for information extraction with a versatile information analysis tool proved promising in solving hard real-world problems of visual object classification. However, the most important aspect of the proposed concept is that it offers a viable candidate for an intelligent information computing framework for energy-efficient, resource-limited devices.

As both a convolutional approach to information extraction and hyperdimensional computing-based information analysis utilize only local data processing patterns, the algorithm is well-suited for its implementation in the form of either ASIC or FPGA devices. Therefore, hardware implementation of the proposed architecture will be the primary objective of our future work.

Another planned direction of further research is concerned with improving the learning performance of the presented multiple-headed setup used for extractor derivation. Given the high cost (in both time and labor) of labeling data, one might consider a scheme in which the classification head is used only sparingly, while the bulk of representation conditioning is done through the other two heads. As unsupervised and supervised approaches can yield very similar representations, our scheme could enable successful learning even when faced with a highly limited volume of labeled examples.

REFERENCES

[1] F. V. Paulovich, M. C. F. De Oliveira, and O. N. Oliveira, "A Future with Ubiquitous Sensing and Intelligent Systems," *ACS Sensors*, vol. 3, no. 8, pp. 1433–1438, Aug. 2018.

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

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv*, Apr. 2018.

[4] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5297–5307.

[5] Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, and J. Malik, "Long-Term Human Motion Prediction with Scene Context," in *Computer Vision – ECCV 2020*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 387–404.

[6] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? a New Model and the Kinetics Dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 4724–4733.

[7] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[8] P. Łuczak, P. Kucharski, T. Jaworski, I. Perenc, K. Ślot, and J. Kucharski, "Boosting Intelligent Data Analysis in Smart Sensors by Integrating Knowledge and Machine Learning," *Sensors*, vol. 21, no. 18, p. 6168, Jan. 2021.

[9] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, Jun. 2009.

[10] A. Joshi, J. T. Halseth, and P. Kanerva, "Language Geometry Using Random Indexing," in *Quantum Interaction*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 265–274.

[11] A. Mitrokhin, P. Sutor, C. Fermüller, and Y. Aloimonos, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, May 2019.

[12] P. Neubert and S. Schubert, "Hyperdimensional computing as a framework for systematic aggregation of image descriptors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 16 938–16 947.

[13] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "VoiceHD: Hyperdimensional Computing for Efficient Speech Recognition," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, Nov. 2017, pp. 1–8.

[14] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A Robust and Energy-Efficient Classifier Using Brain-Inspired Hyperdimensional Computing," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ser. ISLPED '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 64–69.

[15] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, "High-Dimensional Computing as a Nanoscalable Paradigm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2508–2521, Sep. 2017.

[16] D. Garcıa-Lesta, F. Pardo, O. Pereira-Rial, V. M. Brea, and P. Lopez, "HDC8192: A General Purpose Mixed-Signal CMOS Architecture for Massively Parallel Hyperdimensional Computing," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, to be published., May 2022.

[17] K. Greff, S. van Steenkiste, and J. Schmidhuber, "On the Binding Problem in Artificial Neural Networks," *arXiv:2012.05208 [cs]*, Dec. 2020.

[18] G. Hinton, "How to represent part-whole hierarchies in a neural network," *arXiv:2102.12627 [cs]*, Feb. 2021.

[19] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR 2011*, Jun. 2011, pp. 3153–3160.

[20] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1037–1045.

[21] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[22] A. Rahimi, A. Tchouprina, P. Kanerva, J. d. R. Millán, and J. M. Rabaey, "Hyperdimensional Computing for Blind and One-Shot Classification of EEG Error-Related Potentials," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1958–1969, Oct. 2020.

[23] L. Ge and K. K. Parhi, "Classification Using Hyperdimensional Computing: A Review," *IEEE Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, 2020.

[24] P. Kanerva, "Sparse Distributed Memory and Related Models," in *Associative Neural Memories: Theory and Implementation*. New York: Oxford University Press, 1993, pp. 50–76.

[25] A. Rahimi, P. Kanerva, L. Benini, and J. M. Rabaey, "Efficient Biosignal Processing Using Hyperdimensional Computing: Network Templates for Combined Learning and Classification of ExG Signals," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 123–143, Jan. 2019.

[26] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep Clustering for Unsupervised Learning of Visual Features," in *Computer Vision – ECCV 2018*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 139–156.

[27] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[28] L. Mosley, "A balanced approach to the multi-class imbalance problem," *Graduate Theses and Dissertations*, Jan. 2013.

[29] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016-Decem. IEEE, Jun. 2016, pp. 779–788.

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37.

[32] P. de Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A Large Scale Event-based Detection Dataset for Automotive," *arXiv:2001.08499 [cs, eess]*, Jan. 2020.

[33] D. Gehrig, M. Gehrig, J. Hidalgo-Carrio, and D. Scaramuzza, "Video to Events: Recycling Video Datasets for Event Cameras," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 3583–3592.

[34] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten Years of Pedestrian Detection, What Have We Learned?" in *Computer Vision - ECCV 2014 Workshops*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 613–627.