From Pixels to Spikes: Efficient Multimodal Learning in the Presence of Domain Shift

Aaron Chadha

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Electronic and Electrical Engineering

University College London

January 14, 2019

I, Aaron Chadha, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Aaron Chadha

Abstract

Computer vision aims to provide computers with a conceptual understanding of images or video by learning a high-level representation. This representation is typically derived from the pixel domain (i.e., RGB channels) for tasks such as image classification or action recognition. In this thesis, we explore how RGB inputs can either be pre-processed or supplemented with other compressed visual modalities, in order to improve the accuracy-complexity tradeoff for various computer vision tasks.

Beginning with RGB-domain data only, we propose a multi-level, Voronoi based spatial partitioning of images, which are individually processed by a convolutional neural network (CNN), to improve the scale invariance of the embedding. We combine this with a novel and efficient approach for optimal bit allocation within the quantized cell representations. We evaluate this proposal on the content-based image retrieval task, which constitutes finding similar images in a dataset to a given query.

We then move to the more challenging domain of action recognition, where a video sequence is classified according to its constituent action. In this case, we demonstrate how the RGB modality can be supplemented with a flow modality, comprising motion vectors extracted directly from the video codec. The motion vectors (MVs) are used both as input to a CNN and as an activity sensor for providing selective macroblock (MB) decoding of RGB frames instead of full-frame decoding. We independently train two CNNs on RGB and MV correspondences and then fuse their scores during inference, demonstrating faster end-to-end processing and competitive classification accuracy to recent work.

In order to explore the use of more efficient sensing modalities, we replace the MV stream with a neuromorphic vision sensing (NVS) stream for action recognition. NVS hardware mimics the biological retina and operates with substantially lower power and at significantly higher sampling rates than conventional active pixel sensing (APS) cameras. Due to the lack of training data in this domain, we generate emulated NVS frames directly from consecutive RGB frames and use these to train a teacher-student framework that additionally leverages on the abundance of optical flow training data. In the final part of this thesis, we introduce a novel unsupervised domain adaptation method for further minimizing the domain shift between emulated (source) and real (target) NVS data domains.

Impact Statement

This PhD thesis was sponsored in part by an EPSRC CASE award, co-sponsored by BAFTA, the British Academy of Film and Television Arts. During the interactions with the sponsor, the work of Chapter 2 of this thesis had impact and contributed to research developments within the VideoClarity project, a BAFTA research project sponsored by Innovate UK (101932). Namely, we designed and implemented a content-based retrieval system for finding matches to a query from a large corpus of images and videos, based on our findings in Chapter 2. Therefore, elements of the proposals of Chapter 2 may be developed within a commercial context in the future.

Beyond this specific aspect, the proposals of Chapters 3 and 4 on efficient sensing and action recognition may influence future commercial products in these areas. For example, efficient spike-domain sensing or efficient ingestion of codec motion vectors may accelerate recognition tasks within Internet-of-Things oriented applications, such as robotics, visual sensing on-board drones or other mobile visual sensing and analysis contexts. Finally, some of the mathematical techniques that arose from Chapter 5 may find applicability in a number of applications where domain shift is prevalent between one set of labelled data and another set of unlabelled data. There is interest for such techniques where a lack of labelled training data exists in the domain of interest and knowledge has to be transferred from synthetic data, such as in medical imaging.

Acknowledgements

I would like to thank my academic supervisor, Professor Yiannis Andreopoulos, my family and peers for their continuous guidance and support over the course of this PhD. I would also like to thank the Royal Commission for the Exhibition of 1851 for providing additional funding, support and networking opportunities to further my research. Finally, I would like to thank my industrial sponsor, British Academy of Film and Television Arts (BAFTA) for partly funding my PhD.

Contents

List of Figures

List of Tables

Notation

a, A	A scalar
$\boldsymbol{a}$	A vector
$\mathbf{A}$	A matrix
$\boldsymbol{A}$	A set of elements
$\emptyset$	Empty set
$\{0,1\}$	A set containing 0 and 1
$\{0,\dots,n\}$	The set of all integers between 0 and n
$\mathrm{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
$f(\boldsymbol{x};\theta), F(\boldsymbol{x};\theta)$	A function of $\boldsymbol{x}$ parameterized by θ
$\lfloor a \rfloor$	The floor of integer a
$\|\|\boldsymbol{x}\|\|$	L_2 norm of $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	L_1 norm of $\boldsymbol{x}$
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	Inner product between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$\langle x, y \rangle$	Tuple of scalars x and y
$\boldsymbol{x}\|\|\boldsymbol{y}$	Concatenation of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$
$[\boldsymbol{x}_1,\dots,\boldsymbol{x}_N]$	Concatenation of sub-vectors $\boldsymbol{x}_1$ to $\boldsymbol{x}_N$
$\boldsymbol{C}_1 \times \cdots \times \boldsymbol{C}_M$	Cartesian product of sets $\boldsymbol{C}_1$ to $\boldsymbol{C}_M$
$\log(x)$	Natural logarithm of x
$\exp(x)$	Exponential of x
$\mathscr{F}$	A feature space or domain
$\mathbb{R}$	The set of real numbers
$x \sim \mathbb{D}$	A random variable x drawn from distribution $\mathbb{D}$
$\mathbb{E}_{x\sim\mathbb{D}}$	Expectation over x with respect to distribution $\mathbb{D}$

Chapter 1

Introduction and Literature Review

Humans associate with objects in their environment through their sensory modalities, such as vision or sound. In the case of sensory impairment (e.g., blindness), a human can still identify objects using their remaining senses; however, the task becomes more difficult, as they can not rely on the mutual information present between their sensory modalities. For example, both oranges and apples feel spherical in shape but only oranges are also visibly orange in colour; the combination of both information sources facilitates classification of the fruit. Many computer vision systems typically rely on learning features only within a single modality (e.g., text, images or audio). Multimodal machine learning aims to build models that can jointly process and relate information across multiple input modalities, with the intention of leveraging on the different statistical properties in order to improve performance on a given task, such as classification.

In this thesis, we focus primarily on two computer vision tasks, content based image retrieval (CBIR) and action recognition. Conventionally, both tasks ingest whole RGB images as inputs to a learned embedding function. The objective of this thesis is to explore how the RGB inputs can either be additionally processed or supplemented with compressed *visual* modalities (with minimal associated sensing and acquisition complexity), in order to improve the accuracy-complexity tradeoff.

CBIR based on visual queries is a topic of intensive research interest since it finds many applications in visual search [1], detection of copyright violations [2], recommendation services [3] and object or person identification [4]. The task constitutes finding matches to an image query in a large database, based on the content (raw pixel data), rather than associated metadata such as tags or descriptions. There are a number of retrieval engines commercially available that utilize CBIR; one such example is Google Image Search, which allows the user to upload an image or image URL as a query. The system combines reverse image search with available metadata to find matches over the billions of images in Google's databases. In this thesis, we investigate whether the RGB images can be additionally processed for CBIR by adaptively partitioning the input image, in order to extract more invariant features. As such, an overarching retrieval system should be able to better handle both small region-of-interest (ROI) queries and whole image queries. We aim for our approach to be detector/descriptor agnostic, such that it can be easily slotted into existing frameworks, with the intention of adding to the scale invariance of existing shallow and deep-learned descriptor proposals. The end goal is to improve overall retrieval performance on image datasets and subsequently extend this to efficient video indexing and retrieval, to the point of deployment in a commercial application.

Action recognition refers to the task of classifying video sequences based on their constituent human action (e.g., 'running' or 'playing tennis'). Examples where this task plays a pivotal commercial role are video surveillance [5], human-computer interaction [6] or robot learning [7]. Conventional methods for action recognition typically employ a combination of RGB frames and dense optical flow modalities for modelling both the spatial (static) and inter-frame motion dependencies respectively. The problem with optical flow is it is expensive to compute; the flow field is generated by computing partial derivatives at every voxel position. In this thesis, we explore how to design competitive recognition systems with less expensive motion based alternatives to optical flow. We first consider how motion vectors, which constitute a key component of standard video compression algorithms, can be extracted directly from the video codec and employed as the second modality to supplement selectively decoded RGB frames. We design an efficient two-stream framework for processing the RGB frames and motion vector frames per video in a synchronous fashion.

The problem with optical flow and motion vector inputs for multimodal action recognition is that they rely on a conventional active pixel sensing (APS) camera for producing the video recording. APS cameras are limited by framerate, power consumption and shutter speed and thus are not practical for certain applications such as drone surveillance. Instead, we investigate whether we can substitute conventional APS-based sensing with asynchronous neuromorphic vision sensing (NVS) in order to provide a motion based input at low power and storage cost. NVS hardware like the iniLabs DAVIS and the Pixium Vision ATIS cameras [8, 9, 10, 11] emulate the photoreceptor-bipolar-ganglion cell information flow and their output consists of asynchronous ON/OFF address events (a.k.a., spike events) that indicate the changes in scene reflectance. Existing NVS cameras can produce spike representations that can be rendered into frame representations comprising up to 2000 frames-per-second (fps), whilst operating with robustness to changes in lighting and at low power, on the order of 10mW. However, the events generated by NVS cameras are typically sparse and substantially more difficult to train on than APS flow based variants and there is currently very little data available for training. Therefore, in this thesis we propose improvements to NVS emulators that generate emulated NVS events directly from RGB images for the purpose of training action recognition systems with labelled NVS data generated from RGB video datasets. We consider how such emulators can be embedded into a multimodal transfer learning framework that carries out heterogeneous transfer from optical flow (of which there is an abundance of dense frames available) to emulated NVS during training. At inference, assuming the domain shift is small between the emulated NVS data and the real NVS data generated from a camera, we can directly infer on the real data and sparsely selected RGB frames.

It is worth noting that the emulator parameters have to be manually fine-tuned in order to minimize the domain shift between the emulated and real NVS data. Furthermore, in order to accurately quantify the domain shift, this requires a sufficient amount of real NVS data. Given the scarcity of real NVS data and that any recorded data would typically be unlabelled, we can pose this as an unsupervised domain adaptation problem. Domain adaptation refers to methods in which we leverage on a source data distribution in order to learn a different (but related) target data distribution, with the intention of improving accuracy on a given computer vision task. This finds application in numerous fields ranging from medical imaging [12] to spam filtering [13], where there is a scarcity of (labelled) data in the target domain and synthetic versions of the target data are required in order to improve performance on classification based tasks. Likewise, in this thesis, we wish to learn a model that minimizes the distance between the labelled (source) distribution of emulated NVS instances and unlabelled (target) distribution of real NVS instances. We design a framework that builds on existing adversarial domain adaptation methods by embedding task knowledge directly into the discriminator that tries to distinguish between the real and emulated domains. Whereas existing domain adaptation methods are typically evaluated on RGB image datasets only, we additionally aim to demonstrate that in the case of NVS based classification, we are able to substantially minimize the domain shift between emulated and real NVS events.

1.1 Literature Review

In this section, we review current work published in the field of large scale content based retrieval, unimodal and multimodal action recognition and domain adaptation, which constitute the three core vision components of this thesis. Before detailing the current state-of-the-art deep learning methods in each domain, we briefly discuss methods prior to deep learning.

1.1.1 Content Based Image Retrieval

A generic retrieval system should typically have an offline and online component:

- **Offline**: For a given database, the system learns and subsequently encodes the comprising images, such that similar images have similar encodings and dissimilar images have dissimilar encodings.

- **Online**: For a given query, the system finds relevant matches by mirroring the encoding of the database images on the query. We can then search the database and retrieve images with encodings similar to the query.

The encodings need to carry over enough information about their associated image, such that matching images can be found reliably. The inherent difficulty in retrieval is that database images deemed as 'matching' may be subject to viewpoint, lighting or scale changes, or the object-of-interest may be occluded. Therefore, any functional encoding has to be invariant to these changes, so corresponding images are viewed by the system as matches. Furthermore, there is an inevitable tradeoff between retrieval performance and efficiency. Retrieval performance is typically measured as a combination of precision (the fraction of retrieved images that are matches) and recall (the fraction of matching images that are retrieved) and reported in terms of mean average precision (mAP), defined as the mean of average precision (mAP) scores over all queries. Efficiency can be broken down into storage, (offline) computation and (online) processing. For fast retrieval, we are restricted to storing the entire encoding set for the database on random access memory, thus enabling efficient database access. The computation and processing must be fast enough to allow for real-time comparison between the query and database images.

1.1.1.1 'Shallow' Global Descriptors

A global descriptor is a single vector encoding that is representative of the entire image. The motivation behind global descriptors is that we can sacrifice some discriminatory power achieved by local descriptors for a more compact representation with concomitant lower complexity. That is, the structural information in an image can be condensed into a single vector, then image matching can be reduced to computing some distance metric between two vectors, rather than an exhaustive search over matrices.

Bag of Words: Most notably, the Bag-of-Words (BoW) representation [14] draws parallels to text retrieval implementations by employing vector quantization (K-means) to learn a vocabulary of 'visual words' by grouping local descriptors. The local descriptors extracted over image patches are hard assigned to a visual word and the image is thus represented by a histogram of visual word frequencies. Again, borrowing from text retrieval, visual words that possess higher entropy (i.e. rare words) are deemed more discriminative. As such, the histogram tends to be weighted using term frequency (TF) and inverse document frequency (IDF), with the final BoW image representation being a TF-IDF vector. Assuming document (image) d is represented by K-dimensional vector $\boldsymbol{t}$, where K is the vocabulary size, for the i-th component:

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i} \tag{1.1}$$

where n_{id} is the number of occurrences of word i in document d, n_d is the total number of words in document d, n_i is the number of occurrences of term i in the whole database and N is the number of documents in the whole database. Similarity between two images can be computed as the Euclidean distance between BoW vectors in K-dimensional space, with a smaller distance equating to higher similarity. The BoW vector is inherently sparse, and the search space can be reduced further with dimensionality reduction via PCA and whitening [15, 16]. This can be coupled with sparse coding, spatial pooling [17] or learning over multiple vocabularies [15], in order to encode a more discriminative representation (at the cost of n-times training computation for n vocabularies).

A number of successful extensions have been applied to BoW, that go beyond K-means clustering and hard-assignment. One approach is to replace hard-assignment with a ’descriptor space soft-assignment’ [18]. Weights are assigned to each cluster as a function of the standard deviation σ and distance r from local descriptors to the cluster center, $\exp -\frac{r^2}{2\sigma^2}$. In essence, this applies a ’smoothing’ effect to the visual word histogram, diffusing a bin count to neighboring bins. This soft assignment slots into the BoW encoding process; a TF-IDF weighting (and spatial re-ranking) can still be applied, and the resulting mAP gain over hard-assignment is roughly 10-20% on standard image datasets [18]. Another variant of soft-assignment [19] is to create a ‘kernel codebook’ using radius-based clustering instead of K-means.

Vector of Locally Aggregated Descriptors (VLAD): Vector of Locally Aggregated Descriptors (VLAD) is a fixed-size compact image representation that stores first-order information associated with clusters of image salient points [20, 21]. In the offline part of the VLAD encoding, based on a training set of D_{SIFT}-dimensional SIFT descriptors [22] derived from Y training images, a visual word vocabulary is first learned using K-means clustering. This vocabulary comprises K clusters with D_{SIFT}-dimensional centroids $\boldsymbol{M} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K\}$.

For each new test image I (out of a test dataset comprising W images), N interest points are detected (using an affine invariant detector) and described using D_{SIFT}-dimensional SIFT descriptors, thus forming a descriptor ensemble $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. The descriptors $\boldsymbol{x}_n$, $1 \leq n \leq N$, are assigned to the nearest cluster in the vocabulary via a cluster assignment function $f(\boldsymbol{x}_n)$. VLAD then stores the residuals of the SIFT assignments from their associated centroids. The VLAD D_{SIFT}-dimensional encoding $\boldsymbol{v}_k$ for the k-th cluster, $1 \leq k \leq K$, is given by [20, 21]:

$$\boldsymbol{v}_k = \sum_{\forall x_n : f(x_n) = k} (\boldsymbol{x}_n - \boldsymbol{\mu}_k). \tag{1.2}$$

The VLAD encodings for each cluster are concatenated into a single descriptor $\boldsymbol{\phi}(\mathrm{I}) = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K]^{\mathrm{T}}$ with fixed dimension KD_{SIFT}, which is independent of the number of the SIFT descriptors found in the image. The VLAD vectors are then sign square-rooted and L_2-normalized [21]. The intuition behind sign-square rooting (power normalizing) the vector is that it effectively regularizes the vector by removing burstiness in dimensions [15].

In a practical system, the SIFT descriptor length D_{SIFT} is typically 128; if the feature space is coarsely quantized with K set to 64, then the VLAD image descriptor has 8192 dimensions. As with the BoW vector, further dimensionality reduction is achieved with PCA (learned on an independent image dataset), which further minimises the memory footprint associated with each image. The KD_{SIFT} $\times D$ projection matrix $\mathbf{R}$ used by VLAD comprises only the D largest eigenvectors of the covariance matrix [15, 16]. The projected VLAD, $\widetilde{\boldsymbol{\phi}}_{\text{test}}$, of each image in the test dataset is then L_2-normalized, thereby completing the offline part of the VLAD generation.

During online retrieval between a ROI query and test image (for example), after the VLAD encoding and projection of the ROI query has been carried out, the similarity between that and the (projected) VLAD of a test dataset image, $\widetilde{\boldsymbol{\phi}}_{\text{ROI}}$ and $\widetilde{\boldsymbol{\phi}}_{\text{test}}$, can be simply measured using the squared Euclidean distance [21]. With L_2 normalized vectors, this is a monotonic function of the inner product, such that:

$$S_{\text{ROI,test}} = \left\langle \widetilde{\boldsymbol{\phi}}_{\text{ROI}}, \widetilde{\boldsymbol{\phi}}_{\text{test}} \right\rangle, \tag{1.3}$$

where the similarity score $S_{\text{ROI,test}}$ ranges between -1 (completely dissimilar) to 1 (perfect match).

As with BoW, there have been a number of extensions proposed to enhance the performance of the VLAD descriptor. For example, intra-normalization [23] is a normalization scheme that is presented as an improvement to sign-squared rooting, in removing burstiness from visual elements. The sum of residuals is L_2-normalized within each cluster k, and the entire vector is re-normalized, with the result being consistent improvement in retrieval performance. This can be coupled with a cluster center adaptation [23], which resolves the problem of inconsistent vocabularies by refining the center position $\boldsymbol{\mu}_k$ using the assigned local descriptors, as new images are added to the database online. More recent work has also looked at providing a more discriminative VLAD representation by assigning local descriptors to multiple bins based on their dominant orientation intra-cluster and computing residuals for each bin separately [24] - thus incorporating weak geometric information. The method exhibits improved retrieval performance on the standard Holidays dataset.

However, for ROI-based retrieval, VLAD and the similarity measure of (1.3) will produce sub-optimal results for small ROI, because information encoded from the remaining parts of the dataset image will distort the similarity scoring [23]. In this thesis, we propose a method for directly improving ROI-based retrieval that also generalizes to whole images.

Fisher vectors: The Fisher Vector [25, 26] is a generalised case of VLAD that can be extended to convey both first and second order information. A Gaussian Mixture Model (GMM) with parameters $\Theta = (\mu_k, \Sigma_k, \pi_k : k = 1, \ldots, K)$ is trained offline on a training set of SIFT descriptors, $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, where μ, Σ and π represent the mean, covariance matrix and component prior respectively. The mixture parameters are optimised as per convention, using Expectation Maximization (EM). The first order Fisher vector $\boldsymbol{u}$ is the derivative of the log-likelihood with respect to the mean:

$$u_{jk} = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \frac{x_{ji} - \mu_{ik}}{\sigma_i} \tag{1.4}$$

To derive the second order Fisher vector $\boldsymbol{v}$, the derivative of the log-likelihood with respect to the covariance is computed as:

$$v_{jk} = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} \left[\left(\frac{x_{ji} - \mu_{ik}}{\sigma_j} \right)^2 - 1 \right] \tag{1.5}$$

Compared to the VLAD vector, the first order Fisher vector essentially weights the SIFT assignments to clusters by the posterior probability; this is analogous to the soft assignment described, as a BoW extension. The dimensionality of the second order Fisher vector is $2KD_{\text{SIFT}}$. Generally, a second order Fisher vector presents a more discriminative representation for the same vocabulary size than, at the cost of greater training and assignment computation.

Indeed, the same tricks used to improve the performance of BoW and VLAD, such as power normalization (sign-squared rooting) are transferable to the Fisher vector.

1.1.1.2 Deep Learning for Content Based Image Retrieval

CNNs are feed-forward neural networks comprising multiple layers, and typically trained for the classification task on large *a-priori* labelled datasets, such as ImageNet [27]. Following their superior performance in image classification [28], it has been shown that extracted features from intermediate layers are transferable to other visual recognition tasks, including image retrieval [29, 30]. Descriptors derived on these extracted features are competitive or superior to “shallow” learned alternatives, based on VLAD [20] or Fisher vectors [31].

Recent work [32, 33, 34, 35] has been aimed at improving the geometric invariance of the CNN-based descriptors with region or patch-based methods. Earlier methods, such as the work of Gong *et al.* [32], extract image blocks from a grid over multiple scales, which are subsequently fed into the CNN in order to derive features to be aggregated with a VLAD pooling. However this is an exhaustive approach and many blocks may not be informative. Paulin *et al.* [34] improve on this type of approach by selectively extracting patches at keypoints using a Hessian-Affine detector [36], which is coupled with a convolutional kernel network. While these approaches have been shown to achieve competitive performance to the state-of-the-art, they are heavily influenced by traditional methods that use hand-crafted features and require external components such as feature detectors.

To avoid the use of such external (or hand-crafted) components, Tolias *et al.* [37] proposed the use of region-based max pooling directly on the activations as the means to derive a region-of-interest oriented descriptor. This takes advantage of the correspondence between the input semantics and the activations. In other related work, Babenko *et al.* [38] proposed the SPoC descriptor, which combines sum-pooling of the layer activations with a spatial weighting via a center prior. More recently, Kalantidis *et al.* [39] extended on the SPoC descriptor with a cross-dimensional spatial and channel weighting (CroW) and unresized inputs. Their weighting scheme normalizes the activations by the sum of the channel responses and weights the channels by considering the number of non-zero activations per channel. The general framework of all these approaches involves aggregation of the final pooling or convolutional layer and application of principal component analysis (PCA), whitening and truncation in order to derive the resulting global descriptor. It is worth noting that more recent work [40, 41] has looked at end-to-end training with siamese networks and achieved state-of-the-art of results; however, in this thesis we focus on ‘weakly supervised retrieval’ with pre-trained ImageNet models.

1.1.1.3 Efficient and Accurate Matching

In a practical retrieval system, there is an inevitable tradeoff between accuracy and efficiency in the online component of the system. When querying an image on a dataset, an exhaustive nearest neighbours search equates to computing the distance metric between the query and all members of the dataset. For a dataset of size W and with K dimensional vector representations, the order of complexity is $\mathcal{O}(K \times W)$; there is linear growth in search complexity as the dataset grows in size.

Borrowing again from text retrieval systems, an inverted index [14] can significantly improve efficiency by mapping from visual word to image index (rather than vice versa). This capitalises on the sparsity of BoW vectors, which means that each image contains only a subset of the visual word vocabulary. Thus, the search space is reduced, as it is only necessary to search lists that are indexed by the visual words in a query. Recent work [42] has also looked at improving the efficiency of inverted index traversal, by computing an impact score to weight visual words and 'prune' the query. This is complemented with early termination, which sorts index entries by their impact, such that the most promising entries are accessed first.

As is evident, an inverted index relies on sparsity in the descriptor to improve search efficiency, and therefore does not fit naturally into a system built on the denser VLAD and Fisher vector variants. Nonetheless, the general idea is transferable; in order to improve the search efficiency and memory footprint over large datasets, whilst maintaining accuracy, the search space must be compressed intelligently. Approximate Nearest Neighbor (ANN) algorithms [43, 44, 45] are readily employed for such compaction over large datasets.

Hashing and Embedding: Hashing schemes quantize the search space by mapping the data to a hash key via a hash function. Euclidean Locality Sensitivity Hashing (E2LSH) [43] is a popular extension of basic hashing for nearest neighbour search. In essence, the algorithm maps each vector to M distinct hash keys using a random projection, each with an associated hash table. This is conceptually similar to an inverted index, as only dataset vectors mapped to the same hash key as the query need to be searched and the Euclidean distance computed. The hash function for the random projection of a vector x is of the form:

$$\boldsymbol{h}(x) = \left\lfloor \frac{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} - b}{w} \right\rfloor \tag{1.6}$$

where $\boldsymbol{a}$ is a random projection vector drawn from a stable distribution and $b \in \mathbb{R}$ is drawn from 0 to w.

More recent work attempts to improve performance by using vector quantizers such as K-means or Hierarchical K-means [46], to learn a projection, rather than using a structured or random projection. Generally, this does improve nearest neighbour recall, albeit with an extra pre-processing step.

LSH has two fundamental problems. First, the multiple hash tables have to be stored in main memory to allow for fast access. Multi-probe LSH [47] attempts to rectify this, by taking a similar approach to soft assignment on BoW and searching multiple cells nearest to the query hash key, rather than using mutliple hash functions. The second problem relates to the fact that the exact Euclidean distance is computed for those image vectors in the relevant search space; as such, these vectors also need to be loaded into main memory, which means that LSH can lose efficiency very quickly as the dataset size grows. A practical solution is to binarize the vectors based on their hash keys [44, 48]. Assuming that the search space is partitioned by N random planes, a vector is now described with 2^N bits. The angular similarity between two binarized vectors $b(\boldsymbol{x})$ and $b(\boldsymbol{y})$ can then be expressed as a function of the Hamming distance h [49]:

$$\theta = \frac{\pi}{N} h(b(\boldsymbol{x}), b(\boldsymbol{y})) \tag{1.7}$$

Essentially, the binarization forgoes some potentially discriminative information for heavy compaction and efficiency (bit comparisons).

Product Quantization: A popular method for ANN search is product quantization (PQ) [50]. In order to further reduce the search complexity and the required memory footprint when handling large datasets, D-dimensional vectors are typically quantized to produce compact B-bit representations [50]. Consider a D-dimensional query vector $\boldsymbol{x}$. K-means maps the vector $\boldsymbol{x} \in \mathbb{R}^D$ to a vector $q(\boldsymbol{x})$ in codebook $\boldsymbol{C} = \{\boldsymbol{c}_i, 1 \leq i \leq k\}$. For quantizer q with k centroids, the total number of bits used to encode $\boldsymbol{x}$ is $B = \log_2(k)$. For a 64 bit encoding of a 128-D vector, (0.5 bits per dimension) $k = 2^{64}$ centroids must be learned using K-means, which is clearly not feasible. Product quantization (PQ) is an effective solution that reduces the learning and storage requirement by using multiple sub-quantizers rather than a single global quantizer. Importantly, we can couple product quantization with an inverted index for a non-exhaustive search [50]. In short, each vector $\boldsymbol{x}$ is coarsely quantized by quantizer Q and the residuals, $r_Q(\boldsymbol{x})$ are product quantized and stored in inverted lists. The residual vector is thus defined as:

$$r_Q(\boldsymbol{x}) = \boldsymbol{x} - Q(\boldsymbol{x}) \tag{1.8}$$

As such, the query is coarsely quantized and assigned to its w nearest clusters. Distances are computed between residuals only within the corresponding w inverted lists.

1.1.2 Action Recognition

Action recognition or classification is the task of assigning an action label to a video clip. The recognition process can be preceded by action localization to pinpoint the start and end of the relevant action in a longer video clip; however, in this thesis we consider recognition on short video snippets of about 10s that only contain the relevant action. Recognition performance is typically quantified in terms of classification accuracy; i.e., the proportion of correctly classified videos from the test dataset. Prior to deep learning methods, the state-of-the-art for video classification systems relied on a combination of hand-crafted feature detection, description and aggregation. Wang *et. al* [51, 52] were able to achieve promising accuracy for action recognition using dense trajectories. Essentially, given a set of extract frames from a video, feature points are densely sampled at multiple spatial scales and tracked over time at these points using dense optical flow [53, 54]. Thus, for each trajectory neighbourhood, several local descriptors can be computed (HOG, HOF and MBH), and outliers removed with RANSAC for homography estimation. Finally, these descriptors can be aggregated into a global representation using BoW or Fisher Vectors and classification performed with an SVM. Importantly, the core principles behind dense trajectories are transferable to deep learning system design.

1.1.2.1 Deep Learning for Action Recognition

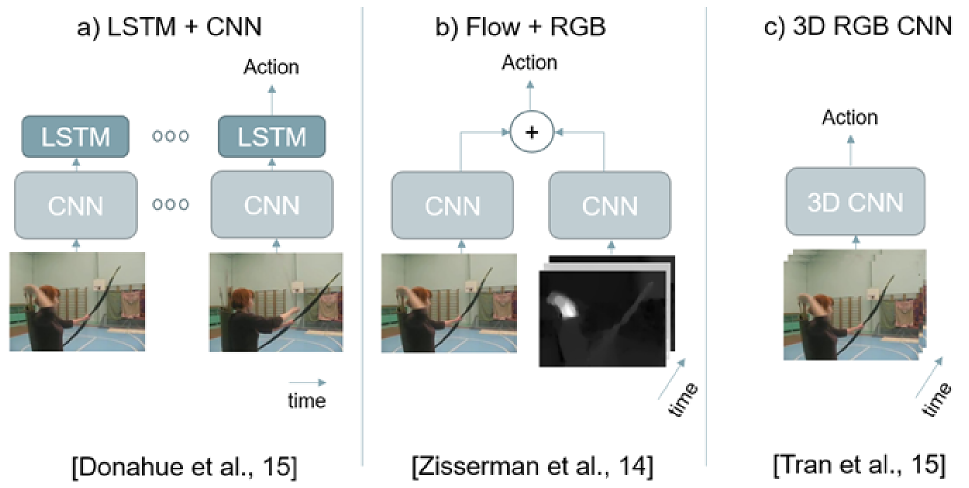


Figure 1.1: State-of-the-art methods for action recognition typically fall into one of three categories, or a combination of categories: a) fusing an LSTM with a CNN by using an LSTM to model the CNN encoded frames over time; b) two-stream CNN that synchronously ingests an RGB frame and flow frames into their respective CNNs, with the encoded logits being fused with averaging/SVM to generate the final prediction; c) 3D RGB CNN that encodes multiple RGB frames with 3D convolutions and spatio-temporal activations.

With increasing dataset sizes and complexity in classification and retrieval, there is a need for deeper and scalable models to learn more complex representations and with larger learning capacity. Subsequently, due to their stand-out performance in image classification [28], deep convolutional neural networks (CNNs) have recently come to the forefront in video classification. Karpathy *et. al* [55] proposed extending the CNN architecture from image to video by performing spatio-temporal convolutions in the first convolutional layers over a 4D video chunk $\mathbf{V} \in \mathbb{R}^{W\times H\times C\times T}$, where W,H are the spatial dimensions, C is the number of channels and T is the number of frames in the chunk. This is the premise behind their slow-fusion architecture, which uses 3D convolutions on RGB frame chunks in the first 3 layers, thus encompassing the full spatio-temporal extent of the input. Notably, experiments demonstrated that feeding a single RGB frame versus multiple frames into this architecture did not have any significant effect on accuracy; in other words, the CNN was not effectively learning on the motion information. Tran *et. al* [56] attempt to improve accuracy by using a deep 3D CNN architecture (resembling VGGnet [57]) together with spatio-temporal convolutions and pooling in all layers. They combine this with a bagging approach over 3 nets and improved dense trajectories, to achieve state-of-the-art accuracy of 90.4% on UCF-101 [58], albeit with heavy computational cost. 3D CNNs typically demonstrate better performance as they model spatio-temporal abstractions of data unlike than their 2D counterparts, albeit at the cost of *(i)* higher complexity due to the extra kernel dimension, and *(ii)* subsequent difficulty in training. Additionally, it has been argued [59] that modelling temporal dependencies only constitutes a fraction of the recognition accuracy, and the majority of performance gain actually comes from the spatial invariance inherent in motion based inputs. Therefore, 2D CNNs still remain a viable option.

Simonyan *et. al* [60] argue that the problem is not the depth or spatio-temporal extent of the architecture but rather the nature of the RGB input that does not effectively present motion information to the CNN. They propose using a 2D architecture with dense optical flow to represent the temporal component of the video. Notably, this temporal CNN outperforms an equivalent 2D spatial stream ingesting RGB frames. However, performance can be improved further by fusing the temporal and spatial streams using a simple score averaging. This two stream architecture achieves 88.0% on UCF-101; however, the computational cost is still relatively high due to the requirement to extract optical flow for the temporal stream (reported as 0.06s for an image pair).

One of the main issues with the above approaches is the short temporal extent of the inputs; each input is a small chunk of frames that only encapsulates a second or so of the video. However, this does not account for cases where temporal dependencies extend over longer durations of video. Feichtenhofer *et. al* [61] attempt to resolve this issue by using multiple copies of the two stream network above. The copies are spread over a coarse temporal scale, thus encompassing both coarse and fine motion information with an optical flow input. The architecture is spatially and then temporally fused using a 3D convolution and pooling. Whilst this does surpass performance of ensemble averaging, it can incur a substantial increase in the number of weights when the number of channels in the preceding convolutional layer is large. Additionally, only the fusion layer learns a true spatio-temporal feature map[1]. Wang *et. al* [63] present a similar approach with temporal segment networks. Essentially, the video is divided into snippets and each snippet is fed into a two-stream network. The temporal and spatial components are independently fused over snippets using a segmental consensus function over their outputs, prior to class fusion. This is combined with warped optical flow input to the temporal streams, where camera motion is compensated for by estimating the homography matrix (analogous to improved dense trajectories). Alternatively, Laptev *et. al* [64] argue that increasing the temporal extent is simply a case of taking the optical flow component over a larger temporal extent. In order to minimize the complexity of the network, they downsize the frames, thus reducing the spatial dimensions. Combining their two stream architecture with improved dense trajectories yields 92.7% on UCF-101 action recognition dataset.

Another method of generating CNNs with a long temporal extent is to integrate an RNN into the architecture. In principle, an RNN would give you infinite temporal context up to the present frame. The idea of appending a 3D CNN with an LSTM was originally introduced by Baccouche *et. al* [65] in 2011; the LSTM appended to the fully connected layers effectively models global motion in the video, whereas the 3D CNN models the fine temporal cues. Donahue *et. al* [66] follow a similar approach but use a 2D CNN to extract features from individual frames. These are subsequently fed into a stack of LSTMs for sequence learning over the input. Due to parameter sharing over time, this model scales to arbitrary sequence lengths.

[1]Alternatively, it is worth noting that the recently proposed ActionVLAD [62] follows a method more in line with trajectory based methods, by aggregating the feature maps generated per time step into a VLAD representation of the action.

Ng *et. al* [67] extend this by considering the effects of appending the CNN with feature pooling versus an LSTM, prior to class fusion. Their results demonstrate that pooling is a good alternative to using an LSTM and achieves competitive accuracy (88.2% vs 88.6% on UCF-101). They also note that simply appending a 2D CNN with an LSTM stack has its limitations. For one, the LSTM is likely to only focus on global temporal motion, such as shot detection and not the fine temporal cues inherent in groups of consecutive frames. Ballas *et. al* [68] propose integrating the RNN units directly into the CNN. Essentially, CNN layers from different time-steps are passed through a GRU to generate a new layer that contains temporal information from both layers. In this way all activations are temporally connected. With an optical flow input, this can achieve 85.7% on UCF-101 (split 1).

Finally, there has been some recent work into unsupervised learning for video representations. The advantage of unsupervised methods are that they do not require pre-labelled data, which can be expensive to collate. Srivastava *et. al* [69] use an LSTM to map the input to a feature descriptor, which can then be passed to decoder LSTMs to perform various tasks, and finetuned for supervised learning.

1.1.2.2 Choice of Modality

The choice of modality constitutes a significant point of contention between recent work, with most methods opting for a combination of an RGB and optical flow-based input, as evident above. The problem with optical flow as an input is that it represents a substantial bottleneck in the processing pipeline. In order to generate optical flow, the video needs to be fully decoded and the flow field generated from consecutive frames. Recent work [70, 71, 72] has experimented with extracting the motion vector (MV) fields directly from the video codec. Whilst performance is usually diminished when training on motion vectors compared to optical flow (as MV fields comprise lower-resolution and noisier representations), no additional computation is required as motion vectors are directly extractable from the compressed video bitstream at very high speed. Kantorov *et al*. [73] initially introduced the MPEG flow model for motion vector based action recognition; however their method was limited to the shallow global descriptors described previously. Recently, Zhang *et al.* [71] utilized codec MVs as an input to a 2D CNN in their action recognition, termed EMV-CNN; however their approach requires upsampled MV fields which diminishes the complexity saving. In this thesis, we show how the video codec can be leveraged to additionally allow for selective decoding of the RGB frames and we directly process low resolution MV frames, thus improving the complexity saving. We also demonstrate how emulated and real neuromorphic vision sensing (NVS) based frames can be leveraged to replace expensive optical flow whilst maintaining competitive performance.

1.1.2.3 Benchmarks

Benchmarking for action recognition is typically performed on UCF-101 and HMDB-51 datasets, which respectively contain about 7000 and 13000 video clips for training and inference. These datasets are relatively small, particularly when compared to benchmarks in the image classification domain. Carreira *et al.* [74] recently introduced the Kinetics video dataset, consisting of 400 action classes and over 400 video clips per class. The larger video dataset translates to more discriminative features learnt during training. Pre-training with Kinetics is coupled with an inflated Inception-V1 architecture for both RGB and motion streams, where the Inception kernels and pooling are extended in the temporal dimension and initialized with the scaled pre-trained ImageNet weights. The result is a significant improvement in performance when subsequently fine-tuning on UCF-101 [58] or HMDB-51 [75]. However, the sheer size of the input volume (64 frames for both streams) is extremely complex to train on, requiring 16 GPUs for fine-tuning on UCF-101 with synchronous training [74]. Due to lack of resources, in this thesis we do not report results with this dataset for the action recognition task and instead restrict our investigations to the well-established UCF-101 and HMDB datasets.

1.1.3 Unsupervised Domain Adaptation

In transfer learning nomenclature, unsupervised domain adaptation falls under the category of transductive transfer learning, where labels are available in the source domain but not in the target domain but the task is shared between domains. The domain D has an associated marginal probability distribution $P(\boldsymbol{X})$ over image samples $\boldsymbol{X} = \{\boldsymbol{x}^i\}_{i=0}^N \in \mathscr{X}$ and a conditional distribution $P(\boldsymbol{Y}|\boldsymbol{X})$ between source images and labels, where $\boldsymbol{Y} = \{\boldsymbol{y}^i\}_{i=0}^N \in \mathscr{Y}$, where $\mathscr{X}$ is the feature space, $\mathscr{Y}$ is the label space and N is the set cardinality. Let us denote the source domain as $\mathscr{D}_S$ and target domain as $\mathscr{D}_T$. The domain adaptation scenario arises when there is a domain shift between the source and target; i.e., $\mathscr{D}_S \neq \mathscr{D}_T$ and $P(\boldsymbol{X}_T) \neq P(\boldsymbol{X}_S)$. For homogenous domain adaptation, we assume that the feature representations are the same; i.e., $\mathscr{X}_S = \mathscr{X}_T$. In addition, for the transductive setting we assume that the label sets are shared between the source and target domains; i.e., $\mathscr{Y}_S = \mathscr{Y}_T$. As the source domain is labelled, we can learn $P(\boldsymbol{Y}_S|\boldsymbol{X}_S)$; the aim is then to learn $P(\boldsymbol{Y}_T|\boldsymbol{X}_T)$ and classify the target instances by leveraging on the source distributions.

Prior to deep learning, methods for homogenous domain adaptation included instance reweighting [76], which typically assumes that the conditional distributions match, but there is a covariate shift between the source and target domains. Other methods that remove the assumption of matching conditional distributions were based on feature augmentation or feature space alignment; for example, by minimizing the Bregman divergence between the source and target PCA subspaces [77].

1.1.3.1 Deep Learning for Domain Adaptation

We briefly discuss recent developments in deep learning for unsupervised domain adaptation. In general, we can segment recent work into discrepancy based, adversarial based or reconstruction based methods.

Discrepancy based methods: Discrepancy based methods minimize the domain distribution discrepancy directly, typically using an IPM based metric such as maximum mean discrepancy (MMD) [78] loss for this purpose. MMD maps the original data to a reproducing kernel Hilbert space (RKHS), where the source and target distributions are assumed separable. Notably, the MMD is commonly used with a Gaussian kernel, which from the Taylor expansion enables matching between all moments of distributions, albeit with some cost in processing. For example, [79] proposed the deep domain confusion (DDC) method which applied a joint classification and linear MMD loss on an intermediate adaptation layer. [80] extended on DDC by adding multiple task-specific adaptation layers and minimizing the domain shift with a multiple-kernel maximum mean discrepancy. Rather than matching the marginal distributions, the joint adaptation network (JAN) [81] aligns the domain shift between the joint distributions of input features and output labels, $P(\boldsymbol{X}, \boldsymbol{Y})$. Alternatively, CORAL [82] matches only the mean and covariance between distributions,which despite its simplicity in only matching second order moments, still maintains competitive performance. More recently, Haeusser *et al.* [83] proposed associative domain adaptation that replaces the MMD with an efficient discrepancy-based alternative that reinforces association between source and target embeddings. The basis of associativity is the two-step round-trip probability of a random walker starting from a labelled source feature and ending at another source feature via transition to a target feature. Associative cycle probabilities are encouraged to be close to a uniform distribution. Effectively, associative domain adaptation uses the clustering assumption, where target and source samples from the same class should be located in high density regions of the feature space, with low density regions between classes. Similarly, this assumption is adopted by Shu *et al.* [84], who add an additional penalty loss function to their adversarial learning framework, in order to punish violation of the clustering assumption.

Adversarial based methods. Adversarial based methods opt for an adversarial loss function in order to minimize the domain shift. The domain adversarial neural network (DANN) [85] first introduced a gradient reversal layer that reversed the gradients of a binary classifier predicting the domain in order to train for domain confusion. Training is performed jointly with a cross entropy loss that classifies the source examples, in order to learn a shared task-based embedding. Other recent proposals [86, 87, 88] have explored generative models such as GANs [89, 90] to learn from synthetic source and target data. These approaches typically train two GANs on the source and target input data with tied parameters. In order to circumvent the need to generate images, Adversarial Discriminative Domain Adaptation (ADDA) [91] was recently proposed as an adversarial framework for directly minimizing the distance between the source and target encoded representations. A discriminator and target encoder are iteratively optimized in a two-player game akin to the original GAN setting, where the goal of the discriminator is to distinguish the target representation from the source domain and the goal of target encoder is to confuse the discriminator. This implicitly aligns the target distribution to the (fixed) source distribution. The simplicity and power of ADDA has been demonstrated in visual adaptation tasks like MNIST, USPS and SVHN digits datasets. The recently proposed DIFA [92] extends this discriminative adversarial framework further by training a generator to generate source-like features that can be used to supplement the source examples during target adversarial training. Finally, Saito *et al.* [93] propose an interesting yet simple proposal of training a feature generator and two classifiers in an adversarial fashion. Their proposal is based on the assumption that target examples that fall outside the support of the source will be misclassified by two different classifiers. They alternately maximize a discrepancy based loss function to train the two classifiers and minimize the same function to train the generator, such that the generator will eventually generate target features that fall inside the support of the source and thus are more easily classified.

Reconstruction based methods: Reconstruction based methods use the reconstruction of source and target examples as an auxiliary task for learn a shared encoding between the source and target domains. This is exemplified by the deep reconstruction network (DRCN) [94], which is comprised of a shared encoder over source and target examples and a two component loss function; a cross entropy loss that classifies the source examples over the shared encoder and a reconstruction loss over a target decoder and shared encoder that reconstructs target examples, in order to ensure that the shared encoder generalizes to the target features Domain separation networks (DSN) [95] also adopt a similar strategy but instead use a combination of private and shared encoders to learn private and shared domain features and a shared decoder that aids generalization between domains. The reconstruction based methods introduced above use a traditional convolutional autoencoder configuration; however, more recently there has been growing interest in adversarial-based reconstruction methods that translate images between domains with a combination of multiple generators and discriminators. As this is typically an under-constrained problem, adversarial reconstruction methods adopt a cycle consistency loss that ensures that an image can be translated back to it's original domain. One such example is CycleGAN [96] which uses two generators F and G to translate from source to target and target to source domains. A source and target discriminator is employed to distinguish between translated and real images for each domain and train each generator adversarially. A cycle-consistency loss is used to ensure that we are able to invert the translation; i.e., for source and target images x_s and x_t, $x_s \rightarrow G(x_s) \rightarrow F(G(x_s)) \approx x_s$ and $x_t \rightarrow F(x_t) \rightarrow G(F(x_t)) \approx x_s$. Whilst CycleGAN learns a separate latent space for the target and source domain, UNIT [97] enforces a shared latent space by splitting each generator into encoder and decoder components and tying weights around the encoder output and decoder input layers. More recently, Lee *et al.* [98] proposed to disentangle the latent spaces between encoder-decoder pairs into a shared content space $\mathscr{C}$ and an attribute space $\mathscr{A}$ for each domain. The attribute space contains domain-specific information, analogous to the private and shared encoder of the DSN.

1.2 Thesis Outline and Research Outcomes

The remaining chapters of the thesis can be divided into the three core components as discussed in the literature review; content based image retrieval (CBIR), action recognition and unsupervised domain adaptation. Inline with our main thesis objective of improving the accuracy-complexity tradeoff for CBIR and action recognition, we begin by exploring improvements to unimodal pixel domain CBIR systems and extend to compressed multimodal representations for action recognition. We ultimately transition from the pixel domain only to the spike domain with neuromorphic vision sensing (NVS) based recognition.

In Chapter 2, we consider the CBIR task and RGB modality only, where we investigate whether we can improve the geometric invariance of our embedding. The objective is to design a retrieval system that can handle queries comprising arbitrary regions-of-interest (ROI) rather than entire images. Our proposal is a compact image descriptor that combines the state-of-the-art in content-based descriptor extraction with a multi-level, Voronoi-based spatial partitioning of each dataset image. The proposed multi-level Voronoi-based encoding uses a spatial hierarchical K-means over interest-point locations, and computes a content-based descriptor over each cell. In order to reduce the matching complexity with marginal or no sacrifice in retrieval performance:

- we utilize the tree structure of the spatial hierarchical K-means to perform a top-to-bottom pruning for local similarity maxima;
- we propose a new image similarity score that combines relevant information from all partition levels into a single measure for similarity;
- we combine our proposal with a novel and efficient approach for optimal bit allocation within quantized descriptor representations.

By deriving both a Voronoi-based VLAD descriptor (termed as Fast-VVLAD) and a Voronoi-based deep convolutional neural network (CNN) descriptor (termed as Fast-VDCNN), we demonstrate that our Voronoi-based framework is agnostic to the descriptor basis, and can easily be slotted into existing frameworks. We demonstrate this capability by designing and implementing a large scale retrieval system for image or video frame retrieval.

In Chapter 3, we consider how the RGB modality can be supplemented with an additional compressed flow based modality in order to improve performance for video classification. Specifically, we investigate action recognition via a two-stream convolutional neural network (CNN) design that directly ingests information extracted from compressed video bitstreams. Our approach begins with the observation that all modern video codecs divide the input frames into macroblocks (MBs). We demonstrate that selective access to MB motion vector (MV) information within compressed video bitstreams can also provide for selective, motion-adaptive, MB pixel decoding (a.k.a., MB texture decoding). This in turn allows for the derivation of spatio-temporal video activity regions at extremely high speed in comparison to conventional full-frame decoding followed by dense optical flow estimation. In order to evaluate the accuracy of a video classification framework based on such activity data, we independently train two CNN architectures on MB texture and MV correspondences and then fuse their scores to derive the final classification of each test video. We find that:

- a CPU-based realization of our MV extraction is over 977 times faster than GPU-based optical flow methods;
- selective decoding is up to 12 times faster than full-frame decoding;
- our proposed spatial and temporal CNNs perform inference at 5 to 57 times lower cloud computing cost than the fastest methods from the literature.

In Chapter 4, in order to improve on the sensing efficiency, we circumvent the limitations of conventional active pixel sensing (APS) cameras by replacing motion vectors with neuromorphic vision sensing (NVS) events. Neuromorphic vision sensing (NVS) hardware is now gaining traction as a low-power/high-speed visual sensing technology that circumvents the limitations of cameras. While object detection and tracking models have been investigated in conjunction with NVS, there is currently little work on NVS for higher-level semantic tasks, such as action recognition. This is partly due to the lack of available labelled NVS training data. We attempt to fill this gap by introducing new options and associated parameters on a recently proposed NVS emulator framework, in order to minimize the domain shift between real NVS events and emulated events generated from APS video. We then embed the improved emulator into a multimodal transfer learning framework that, contrary to recent work that considers homogeneous transfer between flow domains (optical flow to motion vectors), carries out heterogeneous transfer from optical flow to NVS. The potential of our framework is showcased by the fact that:

- for the first time, our NVS-based results achieve comparable action recognition performance to motion-vector or optical-flow based methods;
- the improved NVS emulator and NVS camera hardware offers 3 to 6 orders of magnitude faster frame generation (respectively) compared to standard Brox optical flow.

In Chapter 5, we complete our transition from pixels to spikes by further addressing the inevitable domain shift that arises when training on emulated NVS events and inferring on real NVS events. The scarcity of real labelled NVS data means we can present his as an unsupervised domain adaptation problem. Adversarial discriminative domain adaptation (ADDA) [91] is an efficient framework for unsupervised domain adaptation, where the source and target domains are assumed to have the same classes, but no labels are available for the target domain. While ADDA has already achieved better training efficiency and competitive accuracy in comparison to other adversarial based methods, we investigate whether we can improve performance by incorporating task knowledge into the adversarial loss functions. We achieve this by extending the discriminator output over the source classes and leverage on the distribution over the source encoder posteriors, which is fixed during adversarial training, in order to align a target encoder distribution to the source domain. We additionally consider how the extended discriminator can be regularized in order to further improve performance, by treating the discriminator as a denoising autoencoder and corrupting its input. Our final design employs maximum mean discrepancy and reconstruction-based loss functions for adversarial training. We first demonstrate on standard pixel domain datasets that our proposal is able to compete or outperform the state-of-the-art in unsupervised domain adaptation, whilst offering lower complexity. Finally, we show that the improved performance generalizes to the spike domain by introducing and evaluating on a neuromorphic vision sensing (NVS) sign language recognition dataset, where the source domain constitutes emulated neuromorphic spike events converted from APS video and the target domain is experimental (real) spike events from an NVS camera.

1.2.1 Research Outcomes

The work completed during the PhD has resulted in two journal publications and three conference publications. There is also a conference paper and journal paper currently under review. We only present part of the work in this thesis.

Conference papers:

- Aaron Chadha and Yiannis Andreopoulos. ”Region-of-interest retrieval in large image datasets with Voronoi VLAD.” in International Conference on Computer Vision Systems (ICVS), 2015.
- Aaron Chadha, Alhabib Abbas, and Yiannis Andreopoulos. ”Compressed-domain video classification with deep neural networks: There’s way too much information to decode the matrix.” in IEEE International Conference on Image Processing (ICIP), 2017.
- Abbas, Alhabib, Aaron Chadha, Yiannis Andreopoulos, and Mohammad Jubrani. “Rate-Accuracy Trade-Off In Video Classification With Deep Convolutional Neural Networks.” in IEEE International Conference on Image Processing (ICIP), 2018.
- Aaron Chadha, Yiannis Andreopoulos. “Improving Adversarial Discriminative Domain Adaptation.” (arXiv preprint arXiv:1809.03625).
- Aaron Chadha, Yin Bi, Alhabib Abbas, Yiannis Andreopoulos. ‘Neuromorphic Vision Sensing for CNN-based Action Recognition‘” submitted to IEEE International Conference on Acoustics and Signals Processing (ICASSP), 2019.

Journal papers:

- Aaron Chadha, Yiannis Andreopoulos. “Voronoi-based Compact Image Descriptors: Efficient Region-of-Interest Retrieval With VLAD And Deep-

Learning–based Descriptors.” in IEEE Transactions on Multimedia (TMM), 2017.

- Aaron Chadha, Alhabib Abbas, Yiannis Andreopoulos. “Video Classification With CNNs: Using The Codec As A Spatio-Temporal Activity Sensor” in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 2017.
- Mohammad Jubran, Alhabib Abbas, Aaron Chadha, Yiannis Andreopoulos. “Rate-Accuracy Trade-Off In Video Classification With Deep Convolutional Neural Networks” in IEEE Transactions on Circuits and Systems for Video Technology (TCSVT).

Awards and other accomplishments:

- Awarded Industrial Fellowship from the Royal Commission for the Exhibition of 1851 for work on “High-Speed Analysis of Big Video Data”.
- Google Cloud & YouTube-8M Video Understanding Challenge, Kaggle, 2017. Task: Develop a classification algorithm which accurately assigns video-level labels using the YT-8M V2 dataset released by Google (7 million videos, 450,000 hours). Ranked in top 7% (out of 650 teams) with 7 submissions (username ‘DelveVideo’ on leaderboard).
- Designed and implemented a retrieval system for industrial sponsor BAFTA within the VideoClarity project, a BAFTA research project sponsored by Innovate UK (101932), with potential commercial application in logo detection/de-duplication services; the retrieval system is discussed further in Appendix C.

Chapter 2

Voronoi-based Compact Image Descriptors: Efficient Region of Interest Retrieval with VLAD and Deep-Learning-based Descriptors

In this chapter, we focus on the content based image retrieval (CBIR) task, which is conventionally treated as a unimodal task on pixel domain (RGB) images. Therefore, rather than supplementing with an additional modality, we instead consider how to additionally process the images in order to extract more invariant features that can potentially improve matching capabilities. Specifically, we are interested in the problem of designing a visual-query based retrieval system that is capable of handling both small and large-size "object", or, more broadly, region-of-interest (ROI) queries over image datasets. Given a ROI representing a visual query, the proposed system should return all images from the database containing this query, with matching complexity and storage requirements that remain of the order of standard encodings. This is considerably more challenging than whole-image retrieval systems, as the query object may be occluded or distorted, or be seen from different viewpoints and distances in relevant images [99]. This is also the reason why the original VLAD proposal does not perform as well for this problem [23].

There have been a number of VLAD-derived approaches that tackle ROI based retrieval. Lazebnik *et al.* [100] introduced the concept of spatially partitioning an image into a rectangular grid over multiple scales and encoding per block, as a method of incorporating spatial information; this has found application in both image classification [100, 101] and retrieval [102]. Similarly, the Multi-VLAD descriptor [23] attempts to improve VLAD performance for small ROI by spatially partitioning the dataset images into a rectangular grid over three scales and computing a VLAD descriptor per block. For ROI queries occupying about 11% of image real estate, the Multi-VLAD descriptor has been shown to outperform the single (128×14)-D VLAD (computed over the whole image) in terms of mAP. However, Multi-VLAD achieves 20% lower mAP than the (128×14)-D VLAD when queries occupy a sizeable proportion of the image [23]. In addition, it incurs a 14-fold penalty in storage and matching complexity in comparison to the baseline 128-D VLAD.

We instead propose a new adaptive Voronoi-based encoding (VE), in which we spatially partition the image, using a hierarchical K-means, into Voronoi cells and thus compute multiple descriptors over cells. We couple this with an adaptive search algorithm that minimizes the overall computation for similarity identification by first finding the cells most representative to the query and then deriving a novel single-score metric for the image over these cells. In order for our proposal to be further amenable to big data processing, we additionally propose a novel product quantization framework (based on symmetric distance computation) for our proposal. We show that our proposed framework is agnostic to the descriptor basis by testing on both a Voronoi-based VLAD descriptor and Voronoi-based deep CNN feature descriptor and assessing performance against their respective state-of-the-art variants. Our proposal is inline with grid based spatial search CNN derived methods of Carlsson *et al.* [103, 104], as we are not explicitly computing a global descriptor over extracted features from multiple patches like CNN+VLAD [32] and CKN-mix [34] (which require additional computational pre-processing, e.g., for learning encoding centers).

The Voronoi-based encoding is discussed in Section 2.1.1 and constitutes the offline component of our system. Section 2.1.2 describes the proposed acceleration for online Voronoi-based ROI query search and possibilities for memory compaction to reduce storage requirements. Section 2.2 considers integration of a modified product quantization approach, both for efficient Voronoi-based search and for memory compaction (to reduce storage requirements). Finally, Section 2.3 presents experimental results for the accuracy-complexity tradeoff on the Holidays [105] and Caltech Cars (Rear) [106] datasets, and extends the Voronoi-based partitioning to whole images, demonstrating that we can achieve competitive accuracy compared to state-of-the-art methods at modest computational overhead.

2.1 Proposed Voronoi-based Encoding and its Fast Online Implementation

In this section, we present our proposed Voronoi-based encoding and its fast variant for efficient online Voronoi-based ROI query search. Table 2.1 summarizes the important nomenclature to be used in the remainder of this chapter.

2.1.1 Voronoi-based Encoding and Compact Descriptors

Instead of spatially partitioning the images into a rectangular grid, we propose to partition the image into Voronoi cells over L levels (scales), using hierarchical spatial K-means clustering. The key intuition is that objects that may constitute ROI queries tend to appear as clusters of salient points, potentially interspersed with featureless regions in the image. Therefore, a ROI-oriented partitioning must attempt to adaptively isolate these spatial clusters at multiple levels.

Initially, the entire image is encoded; this comprises level 0 of the Voronoi-based encoding. For level 1, a spatial K-means is computed over the interest point locations in the whole image, which effectively partitions the image into V_1 Voronoi cells. Next, for level 2, a spatial K-means is computed over the interest point locations within each level-1 Voronoi cell, thus partitioning each cell into V_2 constituent cells. In general, each of the V_{l-1} cells in level $l-1$ are partitioned into V_l cells in level l, with $1 \leq l < L$ and $V_0 \triangleq 1$. A base descriptor, whether this be VLAD or

Table 2.1: Nomenclature Table.

Symbol	**Definition**
U	dimensions of unprojected descriptor
D, D'	dimensions of PCA-projected and truncated descriptor and descriptor blocks (resp.)
Y	number of training-set images
W	number of test-dataset images
$\mathbf{R}$	$U \times D$ PCA projection matrix
$\mathbf{\Lambda}, \mathbf{\Lambda}_\mathrm{S}$	diag. eigenvalue matrix, diag. eigenvalue sub-matrix
$\widetilde{\boldsymbol{\phi}}_{\mathrm{ROI}}, \widetilde{\boldsymbol{\phi}}_{\mathrm{test}}$, and $\widehat{\boldsymbol{\phi}}_{\mathrm{ROI}}, \widehat{\boldsymbol{\phi}}_{\mathrm{test}}$	PCA-projected descriptor of a query ROI and a test image (resp.), and whitening-and-normalization based product quantization (WNPQ) descriptor of the same
B, B'	number of bits for quantized descriptor & constituent block
Z, Z'	number of quantization centroids per descriptor and descriptor block
$S_{\mathrm{des1,des2}}$	similarity score between descriptor “des1” and “des2”
M	number of quantization sub-spaces (blocks) for Product Quantization (PQ)
$\boldsymbol{C}_1, \ldots \boldsymbol{C}_M$	PQ codebook per quantization block m, $1 \leq m \leq M$
L	number of levels (scales) used for Voronoi-based encoding (VE)
$V_1, ..., V_L$	number of Voronoi cells per level l, $1 \leq l \leq L$
V_{tot}	number of Voronoi cells in VE
l_{ph1}	level that Phase 1 exits in Fast-VE adaptive search, $0 \leq l_{\mathrm{ph1}} < L$
$S_0^*, \ldots, S_{l_{\mathrm{ph1}}}^*$	similarity score for cell with maximum similarity to the query per level l, $0 \leq l \leq l_{\mathrm{ph1}}$
$v_0, \ldots, v_{l_{\mathrm{ph1}}}$	difference between number of interest points in query and cell corresponding to S_l^*, $0 \leq l \leq l_{\mathrm{ph1}}$
$\widehat{w}_0, \ldots, \widehat{w}_{l_{\mathrm{ph1}}}$	L1-normalized weighting per level l for Fast-VE, $0 \leq l \leq l_{\mathrm{ph1}}$
V_F	number of cells accessed in Fast-VE
$\mathbf{\Sigma}_1, \ldots \mathbf{\Sigma}_M$	covariance matrix per descriptor block m, $1 \leq m \leq M$

aggregated deep CNN features, is encoded over each cell following the description of Section 2.1, giving a total of

$$V_{\mathrm{tot}} = 1 + \sum_{l=1}^{L-1} \prod_{m=1}^{l} V_m \tag{2.1}$$

encodings per image. When PCA-projecting each cell descriptor, we aggregate each level l into a single matrix $\mathbf{\Phi}_l$.

A three-level Voronoi partitioning for an image from the Caltech Cars image dataset with $V_1 = V_2 = 3$ is illustrated in Figure 2.1. The detected points are shown in color in the left image of Figure 2.1, and the level-1 and level-2 Voronoi cells are superimposed with dashed lines on the middle and right image (resp.), with their corresponding descriptors appearing with different colors.

In essence, there are two variables to consider when implementing a Voronoi-based encoding; the number of levels L and the number of Voronoi cells V_l, $1 \leq l < L$, to encode. For the purposes of this chapter, we will consider V_l to be constant for all levels l. In addition, it is worth noting that for the Voronoi-based encoding, we construct a single PCA projection matrix using the entire images of the training set. This is because we found that there is very little gain in retrieval performance when learning separate PCA projection matrices for each Voronoi partition level, mostly due to sufficient variability in scale of ROI in the training images alone. Finally, given we are dealing with PCA on high dimensional data, for the case where the unprojected cell descriptor dimension, U, is greater than the training size Y, we use the manipulation described by Bishop [107]. In essence, we define the covariance matrix for the $U \times Y$ training descriptor matrix $\mathbf{\Phi}_{\mathrm{T}}$ as $\mathbf{\Phi}_{\mathrm{T}}^{T}\mathbf{\Phi}_{\mathrm{T}}$, which provides for a lower dimensional $Y \times Y$ matrix to work with. Following singular value decomposition, we then rotate the derived projection matrix, $\mathbf{R}_{\mathrm{ROT}}$, into the original covariance data space to obtain the $U \times D$ projection matrix, $\mathbf{R}$, using the equivalence:

$$\mathbf{R} = \mathbf{\Phi}_{\mathrm{T}}\mathbf{R}_{\mathrm{ROT}}\mathbf{\Lambda}^{-1}, \tag{2.2}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1^{-\frac{1}{2}}, \lambda_2^{-\frac{1}{2}}, \ldots, \lambda_Y^{-\frac{1}{2}})$ is the diagonal matrix of eigenvalues of $\mathbf{\Phi}_{\mathrm{T}}^T \mathbf{\Phi}_{\mathrm{T}}$.

We conclude this subsection by summarizing the VLAD and deep CNN descriptors utilized for each Voronoi partition.

2.1.1.1 Voronoi-based VLAD (VVLAD)

We require a detector that is robust to scale and viewpoint changes, while also detecting enough points in salient regions to allow for reliable partitioning. Therefore, for VVLAD, we use the Hessian Affine detector [36, 108], which is based on the multi-scale determinant of the Hessian matrix (computed locally), and detects affine covariant regions. SIFT descriptors are produced based on the detected points. It is worth noting that: *(i)* salient point detection is an implicit step in each VLAD computation and not additional processing; *(ii)* unlike Multi-VLAD, there is no need to preprocess the image and exclude featureless regions. As shown in the example of Figure 2.1, smaller Voronoi cells are adaptively formed around regions of tight clusters of detected points.

2.1.1.2 Voronoi-based Deep CNN (VDCNN)

In this case, the salient point detection constitutes additional pre-processing. Nevertheless, this can be achieved efficiently by using the FAST corner detector [109], which classifies a pixel as a corner based on its relative intensity to a set of contiguous pixels. As for the case of VVLAD, the image is partitioned into Voronoi cells based on the location of detected points. Since the deep CNN must take a rectangular input image segment, we compute a bounding box over the constituent points of each cell, and then resize and subtract an average image, as per convention, before feeding into the pre-trained deep CNN. Given that the cells are treated independently, the feed-through can be done in parallel, using multiple copies of the network. In terms of the deep CNN descriptor specifics, we use the CNN-S architecture [110] pretrained on ILSVRC-2012 with batch normalization [111]. The network is sufficiently deep to provide a rich semantic representation of the image/image partitions without overfitting to the classification task. The conventional approach to generating a feature descriptor from the network is to simply extract one of the fully-connected layers [29, 28, 104]. Instead, we extract the last max-pooling layer (Layer 13) of the network, which precedes the fully-connected network and should be less tuned to the classification task. From this layer, we generate a 512-D feature descriptor by averaging the CNN activations over the spatial dimensions. We can also (optionally) apply PCA-projection and truncation to achieve further compaction to 128 dimensions.

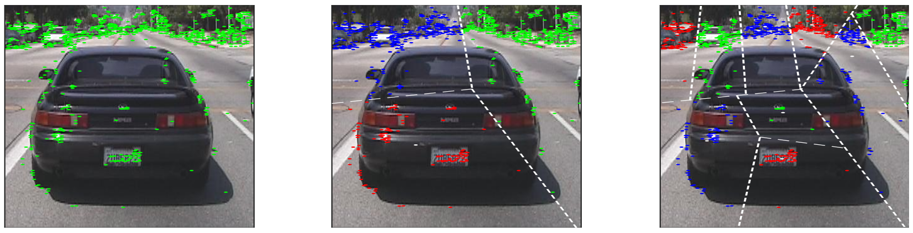


Figure 2.1: Three-level Voronoi partitioning for an image from Caltech Cars dataset. For illustration purposes, SIFT descriptors are color-differentiated for each cell.

2.1.2 Fast Online Implementation: Adaptive Search and Image Similarity Score

Conventionally, we could assign an image score as the global maximum similarity to a query over cells, using (1.3) for each cell. However, the proposed Voronoi partitioning essentially gives us a tree of spatial Voronoi cells where, for L levels, $\prod_{l=1}^{L-1} V_l$ "leaf" Voronoi cells exist at the bottom of the tree. Given that there is inherent mutual information between a cell and its constituent cells, rather than accessing data for all levels and measuring similarity over all V_{tot} cells of the tree indiscriminately, we can design an adaptive search with top-to-bottom tree pruning to find the most relevant Voronoi cells to the query. This reduces the overall execution time and memory accesses when performing a retrieval task, which makes our proposal applicable to very large image databases that would contain millions of images. The top-to-bottom search is carried out in two phases.

Phase-1: Considering the cell of level $l-1$ with maximum similarity to the query [measured via (1.3)], in Phase-1 of the search, we assume that either this cell or a constituent cell within it (at level l) will attain high similarity to the query. If the cell of level $l-1$ is found to attain the highest similarity to the query, we terminate the search for that image at level $l-1$ and proceed to Phase-2. On the other hand, if we find that a constituent cell of level l attains the maximum similarity, we repeat Phase-1 for that cell and its constituent cells at the next level $(l+1)$, until we reach the bottom of the tree, in which case we move to Phase-2.

Phase-2: Let us denote the maximum similarity found by Phase-1 for each level l as S_l^* and assume that Phase-1 exited at level l_{ph1}, $0 \leq l_{\text{ph1}} < L$. Rather than assigning $S_{l_{\text{ph1}}}^*$ as the similarity score between the ROI query and the test image I in the dataset, we perform a weighted sum over all $S_0^*, \ldots, S_{l_{\text{ph1}}}^*$. To this end, we first compute the difference v_l, $0 \leq l \leq l_{\text{ph1}}$, between the number of interest points in the query and the number of interest points in the image dataset cell corresponding to S_l^*. This difference is subsequently used within a scaled inverse function. The weight for S_l^* $(0 \leq l \leq l_{\text{ph1}})$ is thus defined as:

$$w_l = \frac{C}{\max\left(|v_l|, 1\right)}. \tag{2.3}$$

where C controls the order (set as the modal order of magnitude over all v_l). The weight vector over all levels is L_1-normalized so that the image score can be ranked independently of the level l_{ph1} at which Phase-1 terminated. Denoting the L_1-normalized weight as $\hat{w}_l$, the proposed similarity score between a ROI query and dataset image I after Phase-2 is:

$$S_{\text{ROI},I} = \sum_{l=0}^{l_{\text{ph1}}} \hat{w}_l S_l^*. \tag{2.4}$$

For example, for a three-level partition, if a query object is small relative to the image size, we expect that the total number of interest points over the query would be comparable to that of a level-2 cell. Hence, the level-2 maximum dot product S_2^* should receive the largest weighting $\hat{w}_2$ when computing the similarity score. This is expected to be a more robust similarity scoring than just taking a global maximum over all $S_0^*, \ldots, S_{l_{\text{ph1}}}^*$ (as in Multi-VLAD) as the similarity score, since we account for relevant information from all levels.

Summary: We term this two-phase search coupled with the Voronoi-partitioning as *Fast Voronoi-based encoding* (Fast-VE), because it reduces the expected number of cells that are accessed at runtime. The matching complexity is now:

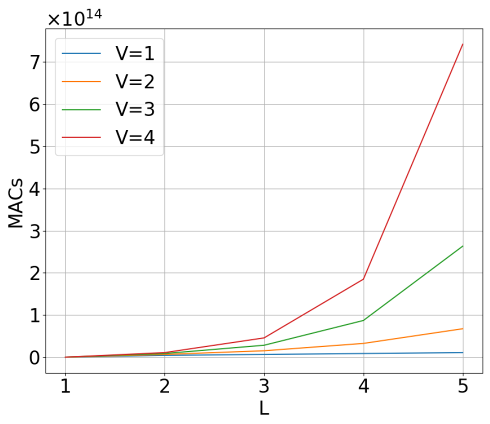
$$V_{\mathrm{F}} = 1 + \sum_{l=1}^{\min\{l_{\mathrm{ph1}}+1, L-1\}} V_l \qquad (2.5)$$

inner products per image instead of the V_{tot} inner products required using a global maximum similarity measure that considers all cells. Due to the weights of (2.4), per image I, along with the Voronoi-based encoding we also store the number of interest points per cell, comprising V_{tot} additional values.

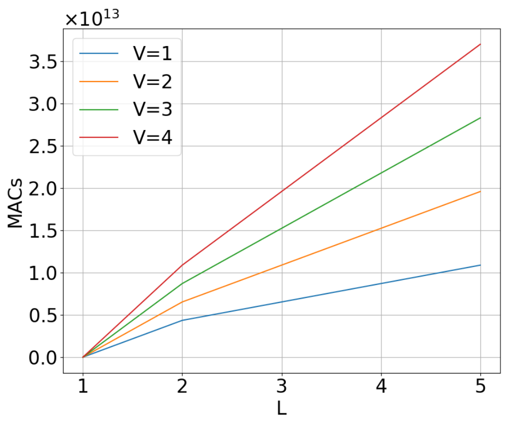
One of the main complexity savings that arises from employing Fast-VE over exhaustive VE is from the matching complexity; i.e., the number of multiply-accumulates (MACs) required to perform a query search over an image corpus. For exhaustive VE, the number of partitions V_{tot} is a function of the number of levels L and the number of partitions per level $V_l, \forall l \in \{1, \ldots, L-1\}$. For Fast-VE, the number of partitions V_{F} is additionally a function of the level l_{ph1} at which Phase-1 exits. In order to visualize the matching complexity saving for Fast-VE compared to exhaustive VE, we assume that V_l is independent of the level (i.e., $V = V_l, \forall l \in \{1, \ldots, L-1\}$ where V is a constant) and that $l_{\mathrm{ph1}} = L-1$, which gives an upper bound on the complexity for Fast-VE. We plot in Figure 2.2 the number of MACs required to match 128-D cell descriptors for exhaustive VE and Fast-VE on a typical commerical retrieval engine such as Tineye[1] (over 17 billion images), with varying number of Voronoi partitions per level V and number of levels L. As is evident, the complexity in matching all cell descriptors for exhaustive VE scales exponentially with the number of levels, whereas for Fast-VE this scales linearly and reduces the number of MACs by at least an order of magnitude.

There is also additional complexity in generating the cell descriptors, which scales accordingly to the matching complexity with number of levels and cells per level. A commercial CPU can typically process around 100GFLOPs per second. Assuming that $L = 4, V = 3$ and the base architecture is VGG-16, then 624GFLOPs

[1]www.tineye.com



(a) Exhaustive VE



(b) Fast-VE

Figure 2.2: Plots of number of multiply-accumulates (MACs) to perform a query search over Tineye image database a) Exhaustive VE and b) Fast-VE for number of levels L and varying number of Voronoi partitions per level $V_l = V, \forall l \in \{1, \dots, L-1\}$

are required for generating exhaustive VDCNN cell descriptors. Therefore,it would take roughly 6 seconds to generate a single image VDCNN with these settings. On the contrary, at most 156GFLOPs are needed for generating Fast-VDCNN cell descriptors with the same settings, which cuts the time for generation by a factor of 4. This saving becomes more meaningful as the number of images to query increases.

It is worth noting that further storage compaction of the Fast-VE is feasible using level projection. Via level projection, we can adhere to memory constraints of a practical deployment for very-large image datasets by only storing the PCA projected cell descriptors for the last level, $L-1$ and computing the cell descriptors for levels $0, \dots, L-2$ at runtime by aggregating smaller-cell descriptors. Given that such storage compaction is of secondary importance in the overall unquantized and quantized Fast-VE design, we include its details as supplementary information in Appendix A.

2.2 Product Quantization for Efficient VE Search

Given that quantized descriptor representations offer significantly-higher compaction than unquantized ones, we extend VE and Fast-VE to quantized representations via a specially-designed product quantization framework. We consider product quantization (PQ) based on SDC for the proposed VE approach[2], where both the query vector $\boldsymbol{y}$ and test dataset vector $\boldsymbol{t}$ are quantized [50].

2.2.1 Product Quantization based on Symmetric Distance Computation for Voronoi-based Encoding

To implement PQ, we consider each vector $\boldsymbol{x}$ as the concatenation of M sub-vectors, $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M]$, each with equal dimension $D' = D/M$. Each sub-vector $\boldsymbol{x}_m$ is encoded from its own sub-codebook $\boldsymbol{C}_m = \{\boldsymbol{c}_{m,i}, 1 \leq i \leq k'\}$, learned using K-means and considered to be of size k' for all m, $1 \leq m \leq M$. As such, the new codebook $\mathbf{C}$ is the Cartesian product of the sub-codebooks, with total size $k = (k')^m$:

$$\boldsymbol{C} = \boldsymbol{C}_1 \times \cdots \times \boldsymbol{C}_M \tag{2.6}$$

Crucially, the learning complexity and storage requirement have been reduced to $\mathscr{O}(Mk'D') = \mathscr{O}\left(k^{\frac{1}{m}}D\right)$, whilst maintaining an exponentially large codebook. The total number of bits used to encode a vector x is now given by $B = M \times B'$, where B' is the number of bits used to encode each sub-vector $\boldsymbol{x}_m$ $(B' = \log_2(k'))$.

Previous work proposed PQ with asymmetric distance computation (ADC) [20, 21], which only encodes the vectors of the test dataset, and PQ with symmetric distance computation (SDC) [50], where both query and test vectors are quantized. By not encoding the query vectors, ADC reduces the overall quantization distortion, thus enhancing the discriminatory power of the system. On the other hand, in SDC the distances between any two subcodewords in the m-th subspace are pre-computed and stored in a $Z' \times Z'$ lookup table, thus enabling efficient ANN search by simple lookup table accesses. Experimental results [20, 21], have shown that ADC and SDC variants of PQ-based VLAD achieve comparable retrieval performance to unquantized VLAD representations with four to ten-fold reduction in storage and search complexity. We opt for SDC-based rather than ADC-based PQ because ADC-based methods require the precomputation and storage of distances

[2]refer to Table 2.1 for nomenclature and symbol definitions

between VE query and test vectors, which is not feasible in a large-scale image retrieval system where potentially any image could form a query.

In SDC-based PQ, the nearest neighbour to $\boldsymbol{y}$ can be approximated by optimizing the distance function $d(q(\boldsymbol{y}), q(\boldsymbol{t}))$. The distance function is typically the squared Euclidean distance [50]:

$$\begin{aligned} d(q(\boldsymbol{y}), q(\boldsymbol{t})) &= \|q(\boldsymbol{y}) - q(\boldsymbol{t})\|^2 \\ &= \sum_{m=1,\ldots,M} d(q_m(\boldsymbol{y}_m), q_m(\boldsymbol{t}_m)) \\ &= \sum_{m=1,\ldots,M} \|q_m(\boldsymbol{y}_m) - q_m(\boldsymbol{t}_m)\|^2, \end{aligned} \tag{2.7}$$

with M the number of subquantizer blocks of (2.6).

The key intuition behind modifying PQ for VE is to treat the constituent Voronoi cells as images and apply PQ on each query and test cell. A single PQ codebook $\mathbf{C}$ is learned using K-means clustering on a training set. Each cell descriptor from the test dataset is thus considered as a concatenation of M subvectors $\widetilde{\boldsymbol{\phi}} = \left[\widetilde{\boldsymbol{\phi}}_1, \ldots, \widetilde{\boldsymbol{\phi}}_M\right]$ of $D' = D/M$ elements each, with each subvector being encoded from its corresponding subcodebook $\boldsymbol{C}_m$. In this way, there is also no dependency on the level l, as we quantize the cell descriptors from a single PQ codebook. All possible distance values between the ith and jth subcodebook vectors in the m-th subspace, $d(\boldsymbol{c}_{m,i}, \boldsymbol{c}_{m,j})$, are pre-computed and stored in a $Z' \times Z'$ lookup table, thus enabling efficient ANN search by simple lookup table accesses.

As the subspaces are orthogonal, we L_2-normalize the product quantization of each cell descriptor's subquantizer block m, $1 \leq m \leq M$, by normalizing the columns of the PQ subcodebooks $\boldsymbol{C}_m$ individually before computing and storing the distance values. For the ith subcodebook vector in the m-th subspace, the normalization term is given by $\sqrt{M}\,\|\boldsymbol{c}_{m,i}\|$. As such, the distance value to be stored between the ith and jth subcodebook vectors is

$$d(\boldsymbol{c}_{m,i}, \boldsymbol{c}_{m,j}) = \frac{\left\langle \boldsymbol{c}_{m,i}, \boldsymbol{c}_{m,j} \right\rangle}{M\,\|\boldsymbol{c}_{m,i}\|\,\left\|\boldsymbol{c}_{m,j}\right\|}. \tag{2.8}$$

Quantizing from the normalized PQ subcodebooks, the distance function between a subspace normalized and quantized query cell descriptor $q(\boldsymbol{y})$ and a subspace normalized and quantized test cell descriptor $q(\boldsymbol{t})$ is now simply

$$d(q(\boldsymbol{y}),q(\boldsymbol{t})) = \sum_{m=1,\ldots,M} \langle q_m(\boldsymbol{y}_m), q_m(\boldsymbol{t}_m)\rangle, \tag{2.9}$$

which is analogous to the squared Euclidean distance of normalized vectors. Importantly, this bounds the similarity score between -1 and 1, which facilitates performance comparisons.

Figure 2.3 illustrates two indicative examples of SDC-based PQ on the $D = 128$ and $D = 2048$ dimensional VLAD, both with and without subspace normalization. The retrieval performance is measured in terms of mean average precision (mAP) on the Holidays dataset [105], using whole-image queries. It is evident from the results of the figure that, for given dimension $D' = D/M$, subspace normalization actually improves retrieval performance, effectively peaking close to $M = D/4$. At this block size, the subspace dimensionality is sufficient such that each subspace is optimally regularized. In addition, we observe that the performance margin between the VLAD descriptor and its subspace normalized counterpart increases significantly with dimension D.

Essentially, we want the block size to be large enough that we encode over a sufficient number of bits; however, beyond a certain block size, we end up normalizing over too few dimensions. In this regard, it is interesting to consider the limit case, where $M = D$, i.e., $D' = 1$. There, subspace normalization results in storing just the sign per cell descriptor. In this extreme case, the similarity between cells can be computed very efficiently by using the Hamming distance, i.e., without accessing any lookup tables.

Concerning storage requirements, assuming that the components of the unquantized cell representation are kept as 32-bit floating-point numbers, their offline storage requirement is $D \times 32$ bits per test image. On the other hand, our product-quantized cell descriptor requires $V_{\text{tot}} \times B$ bits per test image, which is independent of the dimension, D. In addition, for the entire test dataset, the total storage requirement for the quantization lookup tables is $Z' \times Z' \times M$. As the test dataset grows in

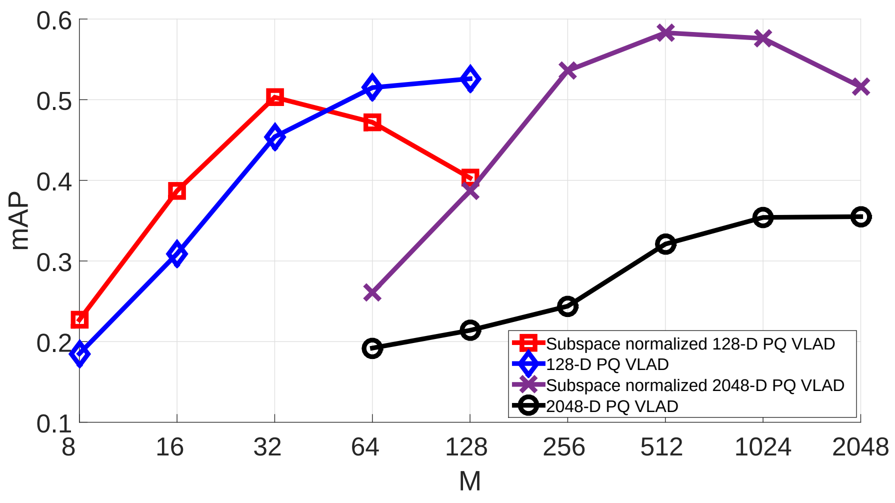


Figure 2.3: Plot of mean average precision (mAP) with varying number of PQ blocks, M, for PQ VLAD descriptors on the Holidays dataset.

size, this value becomes negligible in comparison to the storage requirement for the product-quantized descriptors.

Finally, with regards to the search complexity, the inner products have been replaced by read accesses to the look-up tables. As such, the product quantized Fast-VE now has an upper bound on complexity of $M \times V_\text{F}$ reads, which is independent of the descriptor dimension per cell.

2.2.2 Optimal Bit Allocation via Whitening and Subspace Normalization

Given the presence of multiple cells in the Voronoi-based encoding, it is important to derive an appropriate bit allocation strategy that minimizes the quantization distortion.

Assumption 1. *We consider successive samples of each subspace-normalized VE component (dimension) $\widetilde{\phi}_i$ $(1 \leq i \leq D/M)$ to be modelled by independent, normally-distributed, random variables, with corresponding variance σ_i.*

Under Assumption 1, the normalized random vectors $\widetilde{\boldsymbol{\phi}}_m$ of all subspaces m, $1 \leq m \leq M$, can then be represented by independent and identically-distributed multivariate Gaussians[3], with corresponding diagonal covariance matrices $\mathbf{\Sigma}_m$. The rate-distortion function for independent, normally-distributed random variables [114] can be extended to the multivariate case in order to derive the optimal bit allocation strategy for VE. This leads to the following proposition.

Proposition 1. *Under Assumption 1, optimal bit allocation after subspace normalization in VE can be achieved by balancing the variances of the subspaces.*

See Appendix B.

Recent work [113] employs an optimized product quantization (OPQ) that effectively leads to balanced subspace variances by assigning principal components to a subspace with the objective of balancing the product of eigenvalues per subspace. This corresponds to performing a permutation of the principal components to achieve balanced variances. Jegou *et al.* [21] propose balancing the component variance with a random orthogonal rotation, but this removes the decorrelation achieved by PCA. A different approach is proposed by Brandt *et al.* [112]: one can achieve a constant quantization distortion per subspace by varying the number of bits assigned to each principal component, at the cost of increased training and runtime complexity. Finally, Spyromitros-Xioufis *et al.* [115] consider the effects of applying a random orthogonal rotation on PCA-projected and whitened VLAD vectors prior to product quantization. However, whitening inherently balances the subspace variances by setting $\mathbf{\Sigma}_m$ to the identity matrix for all m, which also preserves decorrelation and mitigates descriptor bias from visual word/component co-occurrences [15, 16]. As such, we propose a simple and effective solution for the bit allocation that adheres with the theoretical result of Proposition 1: we use a whitening approach after PCA (and prior to the product quantization), together with the subspace normalization described in the previous section (and shown to be beneficial by the tests of Figure 2.3). Specifically, per cell, we can express the relationship between a projected descriptor $\widetilde{\boldsymbol{\phi}}_m$ and its whitened and normalized counterpart $\widehat{\boldsymbol{\phi}}_m$ in the m-th subspace as:

[3]The Gaussian assumption is necessary for some of the theoretical derivations, but is also proven to hold in practice [112, 113].

$$\widehat{\boldsymbol{\phi}}_m = \frac{\boldsymbol{\Lambda}_{\mathrm{S},m}\widetilde{\boldsymbol{\phi}}_m}{\left\|\boldsymbol{\Lambda}_{\mathrm{S},m}\widetilde{\boldsymbol{\phi}}_m\right\|}, \tag{2.10}$$

where $\boldsymbol{\Lambda}_{\mathrm{S},m} = \mathrm{diag}(\lambda_{a+1}^{-\frac{1}{2}}, \lambda_{a+2}^{-\frac{1}{2}}, \ldots, \lambda_{a+D'}^{-\frac{1}{2}})$ is the diagonal subspace matrix of eigenvalues of the training-set covariance matrix, with λ_i associated with the i-th largest eigenvector and $a = D'(m-1)$.

The advantage of using whitening and normalization against previous approaches is that there is no need for any additional pre-processing, such as learning a rotation matrix or variability in the bit allocation across the principal components. We term our approach *whitening & normalization based product quantization* (WNPQ).

2.3 Experimental Evaluation

2.3.1 Datasets

We measure performance on the Holidays and Caltech Cars (Rear) test image datasets. For both datasets, a set of predefined queries and hand-annotated ground truth is used.

Caltech + Stanford Cars [106, 116]: This test dataset consists of 1155 (360×240) photographs of cars taken from the rear. Subsequently, we test on a subset of 416 images from the Caltech Cars (Rear) dataset, from which we select 10 images and perform three tests: *(i)* we mimic a surveillance test by selecting only the license plates as ROI-queries; *(ii)* we select as mid-scale ROI-queries a section of the car trunk, and *(iii)* use the whole images as queries. An example of the query subset is given in the left part of Figure 2.4. For the license plate test, we manually create “good” and “junk” ground-truth files over matching images [99]; “junk” ground truth comprises any image in which the query (i.e., the license plate) is barely visible or not distinguishable by the interest point detector. To provide a more rigorous and diversified test, we combine the Caltech Cars subset with another independent set of 1000 distractor images from the Stanford Cars dataset [116], comprising various car models and orientations, giving the Caltech + Stanford Cars dataset.

Holidays [105]: The Holidays test dataset consists of 1491 images, mainly consisting of holiday photos. There are 500 "whole image" provided queries of a distinct scene or object. In order to test on a smaller scale, we also select salient regions from a subset of 40 query images as ROI queries into our system. An example ROI query with its corresponding matching image set is shown in the right part of Figure 2.4.

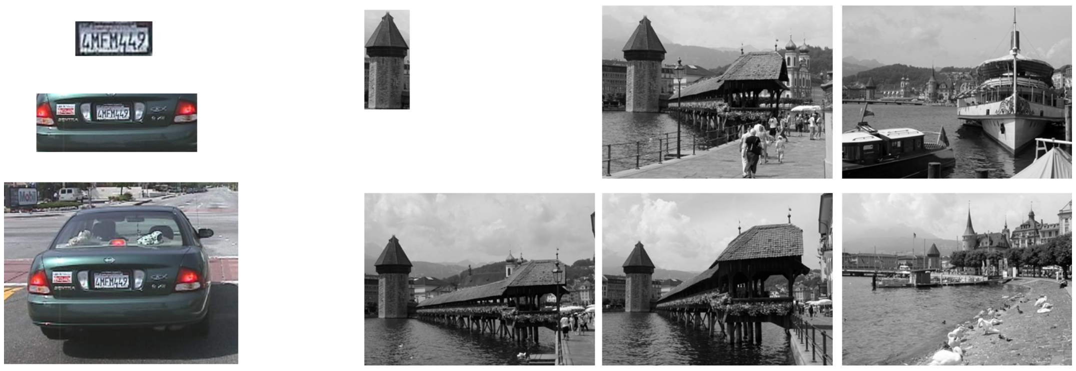


Figure 2.4: (Left) Example queries for the Caltech Cars dataset. (Right) Example ROI query (top left) and matching image set for the Holidays dataset (remaining images).

2.3.2 Accuracy-Complexity Tradeoff

Inline with our thesis objective, we report the accuracy-matching complexity tradeoff for our proposal with and without quantization versus baseline descriptors and Multi-VLAD or Multi-CNN variant that both use a conventional grid based partitioning. A Multi-CNN descriptor can be devised analogously to Multi-VLAD, i.e., by dividing the image at level l into an $(l+1) \times (l+1)$ grid and computing the similarity score between two images as the global maximum inner product over all partitions. For the general case of L levels, the total number of partitions (incl. the whole image as level 0) is:

$$P_{\text{tot}} = \sum_{l=0}^{L-1} (l+1)^2 = \frac{(L+1)(L+2)(2L+3)}{6} \tag{2.11}$$

Matching complexity is defined as the number of multiply-accumulate (MAC) operations for unquantized descriptors, or the number of look-up table reads for quantized descriptors, which scales with the size of the image corpus we are searching over.

For Caltech + Stanford Cars, we compare our proposed Fast-VVLAD descriptor with baseline VLAD descriptors of varying dimensionality (128-D, 768-D and 1664-D) and the Multi-VLAD descriptor. For Holidays, we compare our proposed Fast-VDCNN descriptor with a baseline 128-D and full unprojected 512-D CNN pooled feature descriptors, and the Multi-CNN descriptor. Unless stated otherwise, all vectors are whitened and re-normalized post-PCA. The retrieval performance is measured by creating a ranked list and computing the mAP over all queries. Per descriptor, we compute the matching complexity averaged over all tests and normalized to the baseline 128-D descriptor complexity.

2.3.2.1 Setup

Caltech + Stanford Cars: Due to the specificity of the Caltech + Stanford Cars dataset, together with the lower ROI resolutions, using a deep CNN pre-trained on ImageNet is not a viable option. For example, ImageNet (ILSVRC2012 dataset [111]) does not contain any substantial number of images (and associated labels) corresponding to car license plates; therefore, the pre-trained deep CNN descriptor will not be suitable for such images. For these reasons, we have established that, in this dataset, the utilized deep CNN descriptor is outperformed by VLAD descriptor variants, particularly on license-plate queries. Thus, we use this dataset to test how the proposed Voronoi-based encoding performs with the “shallow” learned VLAD descriptor of Subsection 2.1.1.1.

For the VLAD computation, we follow the design of Subsection 1.1.1.1. The PCA projection matrix, visual word centers and PQ codebook are learned on an independent dataset of 2000 car images from the Stanford Cars dataset [116]. For Fast-VVLAD, we set: $K = 64$, $L = 3$, $V = V_1 = V_2 = 3$, with 128-D VLAD per cell and compile a ranked list from the relevant similarity score. For VLAD, we use: 128-D, 768-D and 1664-D sizes, in order to align the VLAD matching complexity with that of the Fast-VVLAD descriptor. We set $L = 3$ for the Multi-VLAD descriptor, such that it is inline with our Voronoi partitioning. This results in a (128×14)-D descriptor size per image [23], as derived from (2.11).

WNPQ Parameter Selection: We set $M = 32$ for the quantized 128-D VLADs. For the 768-D and 1664-D quantized VLADs, we respectively set the block size to $M = 96$ and $M = 208$. For the quantized VVLAD, we set $M = 32$ for all cell VLADs. Finally, quantized Multi-VLAD also uses $M = 32$. These settings for the block size were chosen to align the matching complexity of the quantized 1664-D VLAD with that of the quantized Fast-VVLAD, whilst providing the 768-D VLAD as a solution with mid-range complexity. Notably, we fix $Z' = 256$ for all experiments. Higher values for Z' increase the computational load for each block quantizer, whilst increasing the storage requirement of the look-up tables ($\mathscr{O}(Z'^2)$), which is an important detriment as these tables need to be sufficiently small to fit in cache memory [50].

Holidays: The Holidays dataset provides a less controlled test for our system. The scenes in the Holidays dataset are better represented by a deep CNN architecture trained on ImageNet, particularly due to their high resolution. Similar to prior work [38], we have confirmed that deep CNNs substantially outperform VLAD descriptors for this dataset. Therefore, we use this dataset to test how the proposed Voronoi-based encoding performs with the deep CNN descriptor of Subsection 2.1.1.2.

For the utilized CNN-S architecture [110], all images and image partitions are resized to 224×224 and fed into the network after subtracting an average image. The final feature descriptor is 512-D, which can then be normalized, PCA-projected to 128-D and whitened. However, following a similar approach to the instance retrieval pipeline on the VLAD descriptor, we normalize, sign-square root and renormalize the feature descriptor prior to PCA and whitening, with the intention of minimising the burstiness of dimensions and thus adding to descriptor invariance [21]. It is worth noting that contrary to recent works [38, 40], we do not manually rotate the images in the Holidays dataset as we do not deem this to be a fair representation of data ‘in the wild’.

Parameter selection for the Voronoi partitioning: For the VE of all cases, the FAST corner detector [109] is used, and we learn the PCA projection matrix and PQ codebook on a subset of 4000 images from the ILSVRC-2010 validation set. For Fast-VDCNN, we set: $L = 3$, $V = V_1 = V_2 = 3$, with a 128-D CNN feature descriptor per cell and compile a ranked list from the relevant similarity score. For Multi-CNN, we use the same grid partitioning as Multi-VLAD, with $L = 3$, thus producing a (128×14)-D size per image.

WNPQ Parameter Selection: We set $M = 32$ for the quantized 128-D CNN feature descriptors. For the 512-D quantized CNN feature descriptor, we set the block size to $M = 128$. For the quantized Fast-VDCNN, we set $M = 32$ for all cell descriptors. Finally, quantized Multi-CNN also uses $M = 32$. As with the VLAD descriptors, we fix $Z' = 256$ for all experiments to keep the storage requirement for the lookup tables constant.

2.3.2.2 Results with Unquantized Descriptors

Caltech + Stanford Cars: Figure 2.5(a) summarizes the retrieval performance of all unquantized VLAD methods on the Caltech + Stanford Cars dataset. The first observation is that the Fast-VVLAD descriptor offers competitive performance to the larger 1664-D VLAD, whilst decreasing the matching complexity by more than 50%. In addition, Fast-VVLAD performs significantly better on license plate queries (blue stem) than both the 128-D VLAD and its 768-D VLAD complexity counterpart, yielding respective mAP gains of over 200% and 41%. Fast-VVLAD maintains consistently-good mAP even with the larger ROIs of car trunks (green stem) and whole-image queries (red stem), and is only outperformed on whole-image queries by VLAD by (up to) a 7% margin. Finally, Fast-VVLAD maintains competitive performance to Multi-VLAD on all query types, whilst offering lower dimensionality and matching complexity.

Holidays: Figure 2.5(b) summarises the retrieval performance for the 500 whole-

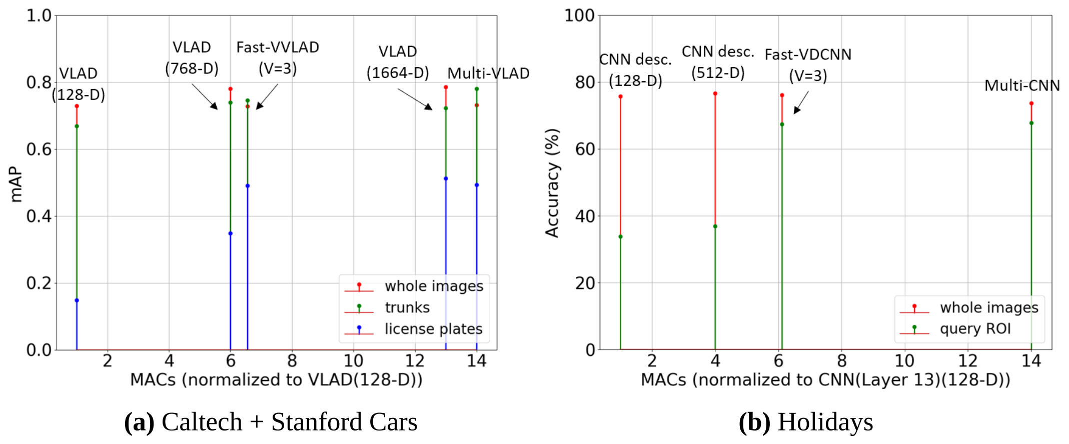


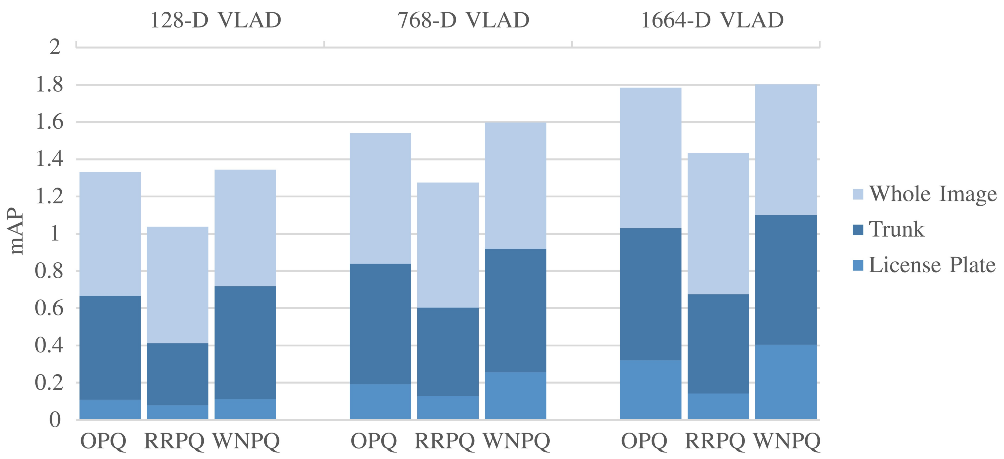
Figure 2.5: Plots of mean average precision (mAP) versus number of multiply-accumulates (MACs) for unquantized descriptors for (a) Caltech + Stanford Cars and (b) Holidays datasets. For Caltech + Stanford Cars, each descriptor is evaluated on license plates, trunks and whole images. For Holidays, each descriptor is evaluated on query regions-of-interest (ROI) and whole images. The stems for each query set are overlaid into a single plot. 'VLAD' and 'CNN desc.' refer to the baseline VLAD and CNN(Layer 13) descriptors respectively.

image queries and 40 smaller ROI queries on the Holidays dataset. Interestingly, the Fast-VDCNN remains competitive on whole image queries. This is attributed to the Fast-VDCNN similarity score of (2.4) that considers all partition levels, which provides robustness against false positives. The Fast-VDCNN is found to outperform Multi-CNN for whole image queries (red stem) and maintain very competitive performance on ROI queries (green stem), while offering more than 50% reduction in the matching complexity.[4] Fast-VDCNN was also found to substantially outperform the lower dimensional CNN feature descriptors for ROI queries (gains exceeding 50% in mAP). Given that the utilized CNN-S descriptor derived from Layer 13 is limited to 512 dimensions [110], we also benchmarked using the first fully connected layer (FC1), which allows for a large 4096-D feature descriptor. Nevertheless, the FC1 descriptor performed significantly worse than our 512-D Layer 13 descriptors for both query ROI and whole images, scoring mAP of 28.3%

[4]We have also validated that this saving translates to practical runtime saving: by adding a large distractor set (thereby scaling the dataset size to 150K images), we found that Fast-VDCNN based retrieval is 40% faster than Multi-CNN retrieval, with execution time comparable to the baseline 512-D CNN feature descriptor.

and 71.4%, respectively. This serves as an additional validation for our choice for the utilized CNN layer.

2.3.2.3 Results with Quantized Descriptors



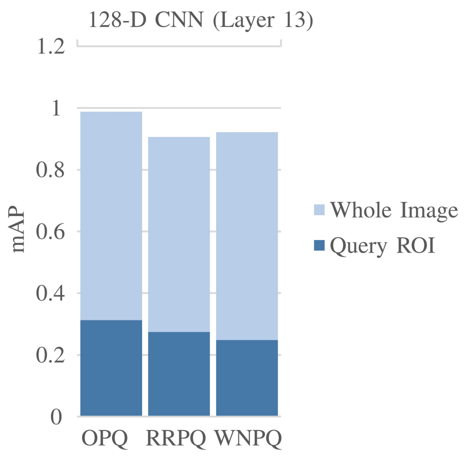
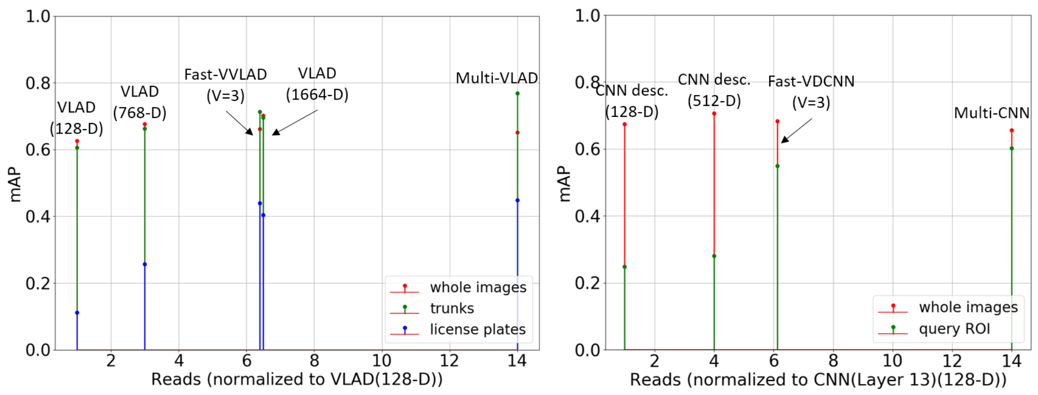


Figure 2.6: Comparison of stacked mean average precision (mAP) for optimized (OPQ), random rotation pre-processing (RRPQ) and our proposed whitening and normalization (WNPQ) product quantization methods. Top: Caltech + Stanford Cars dataset: mid blue = license plate, light blue = whole image, dark blue = trunk. Bottom: Holidays: dark blue = query ROI, light blue = whole image. We consider 128-D descriptors on Holidays dataset and 128-D, 768-D and1664-D descriptors on Caltech + Stanford Cars dataset.

WNPQ against other quantization methods: We first consider the performance of the proposed WNPQ method against other state-of-the-art methods, namely the parametric optimized product quantization (OPQ) [113] and product quantization



(a) Caltech + Stanford Cars **(b)** Holidays

Figure 2.7: Plots of mean average precision (mAP) versus number of lookup table reads for WNPQ quantized descriptors for (a) Caltech + Stanford Cars and (b) Holidays datasets. For Caltech + Stanford Cars, each descriptor is evaluated on license plates, trunks and whole images. For Holidays, each descriptor is evaluated on query regions-of-interest (ROI) and whole images. The stems for each query set are overlaid into a single plot. 'VLAD' and 'CNN desc.' refer to the baseline VLAD and CNN(Layer 13) descriptors respectively.

with a random rotation pre-processing (RRPQ)[21]. As the OPQ and RRPQ descriptors are not normalized, we use the squared Euclidean distance metric for these methods and compare retrieval performance on both datasets. The results of Figure 2.6 show that the proposed WNPQ method outperforms RRPQ and, for the majority of the tests, also outperforms OPQ. Essentially, the WNPQ maintains its high retrieval performance when the dimensionality is increased from 128-D to the 768-D and 1664-D VLAD descriptors. To ensure a fair comparison, and because the proposed WNPQ was shown to provide for the best overall performance, we use it to quantize all the descriptors under comparison.

Caltech + Stanford Cars: Figure 2.7(a) summarises the performance of the various descriptors with WNPQ on the Caltech + Stanford Cars dataset. On whole images, coupled with the aggregated similarity score, Fast-VVLAD offers superior performance to the 128-D VLAD, with an mAP gain of 6%. The 1664-D VLAD, which is now of comparable complexity to the Fast-VVLAD, is outperformed on the small license plate queries, with mAP gain of 9%, but remains superior for whole-image queries. However, it is worth mentioning that the gain from Fast-VVLAD on small queries outweighs any loss on larger queries, thus making it favorable. Finally, the quantized Multi-VLAD offers marginally superior mAP to Fast-VVLAD, albeit at the cost of twice the matching complexity and higher descriptor storage size[5].

Holidays: For the Holidays dataset, as illustrated in Figure 2.7(b), the quantized Fast-VDCNN maintains its mAP gain on query ROI over the quantized 128-D and 512-D CNN feature descriptors, while the descriptor storage has been reduced by a factor of 16 compared to its unquantized counterpart. In addition, the Fast-VDCNN still performs better than quantized Multi-CNN on whole image queries, with an mAP gain of 4%.

2.3.3 Further Improvements on Whole-image Search

The experimental results of the previous section show that Fast-VVLAD and Fast-VDCNN clearly outperform their counterparts for ROI image search, while being competitive for whole-image search. The performance on whole image queries is primarily controlled by the dimension of the level-0 (whole image) component. For experiments in the previous section, we set the dimension uniformly across all components of the Voronoi-based descriptor, i.e., 128-D descriptor per cell. As a result, mAP for the Voronoi-based descriptors on whole images is comparable to that of the 128-D reference descriptors. One option to tailor performance towards whole image queries or smaller ROI queries is by tapering the dimension across levels; we leave this as a topic for future study.

Another approach to boost performance for whole image queries is by accounting for multiple scales in *both* the query and dataset images. In other words, rather than applying Voronoi partitioning *only* on the dataset images, we can also apply Voronoi partitioning *on the query image* over multiple levels and submit each of the

[5]It is worth noting that, given we use a single PQ codebook for quantizing all cell components of Fast-VVLAD and Fast-VDCNN, all quantized based systems have a $Z' \times Z' \times M$ bit cost for storing the look-up tables. This means that, for example, although the 1664-D VLAD offers a lower storage size to Fast-VVLAD, there is an additional 1.7MB cost to store the look-up table, versus 262kB for Fast-VVLAD. However, as mentioned previously, the significance of this additional storage cost diminishes when increasing the test dataset size.

V_{tot} query partitions as a subquery. Notably, using the Fast-VDCNN for the dataset image encodings, each subquery is matched only against representative cells in the dataset images (i.e., between 4 to 7 cells), which are determined by the adaptive search proposed in Section 2.1.2. The inner product between the original query image and a dataset image is taken as the average inner product over all V_{tot} subqueries. While this incurs linear increase in the search complexity (by V_{tot}), this scales better than the quadratic search complexity achieved by Carlsson *et al.* [104, 103], where exhaustive search amongst all subqueries is carried out.

Table 2.2 compares the retrieval performance of the proposed Fast-VDCNN descriptor against the current state-of-the-art on the Holidays dataset that use networks pre-trained on ImageNet. The Fast-VDCNN descriptor is generated under the configuration of Section 2.3.2.1, albeit now also partitioning the queries with $V_1 = V_2 = 2$ and resizing image partitions to 448×448. Beyond benchmarking against the grid-based spatial search method of Carlsson *et al.* [104], we also compare our results with the recently-proposed CNN+VLAD [32], CKN-mix [34], the hybrid FV-NN approach of Peronnin *et al.* [117], as well as lower-dimensional but more computationally-intensive proposals [6] that perform competitively [38, 40, 118]. Evidently, the additional scale and location invariance provided by the Voronoi partitioning leads to the proposed Fast-VDCNN achieving competitive performance to other CNN derived frameworks and hybrid variants, without manually rotating the images, and despite the fact that our feature descriptor is built directly from a pre-trained network and incurs modest computational and storage requirements.

[6]In particular: the SPoC descriptor [38] offers the best performance to dimensionality, but utilizes a deeper and a more computationally heavy CNN (144M parameters vs 76M parameters for our architecture) and a larger image input size, the R-MAC based descriptor uses Siamese learning with supervised whitening, and NetVLAD requires additional processing (soft assignment and normalizations within the NetVLAD layer) to encode VLAD from the network activations. On the contrary, under the chosen configuration, the proposed Fast-VDCNN approach allocates only 128 dimensions per cell and accesses between 4 to 7 cells for each image subquery.

Table 2.2: Comparison of whole-image retrieval performance (mAP) with state-of-the-art for the Holidays dataset [105]. The proposed approach allocates 128 dimensions per partition cell.

	D_{tot}	Whole Image
Proposed Fast-VDCNN	1.66K (128)	0.821
FV-NN (Peronnin *et al.*) [117]	4K	**0.835**
CNN + VLAD [32]	2K	0.802
CNN (Carlsson *et al.*) [104]	4K-15K	0.769
CKN-mix [34]	4K	0.829
SPoC (w/o center prior) [38]	256	0.802
R-MAC [40]	512	0.825
NetVLAD [118]	256	0.799

2.4 Conclusion

We improve on unimodal content-based image retrieval by additionally processing the RGB images to generate more invariant features. Namely, we propose a novel descriptor design, termed Voronoi-based encoding. We show how VE could fit into a practical ROI-based retrieval system via the proposed fast search, memory-efficient design, product-quantization based lossy compression techniques, and robust similarity scoring mechanisms. We test retrieval performance on two datasets, using VLAD and a deep CNN as our descriptor basis. Our results show that our approach is descriptor agnostic; the proposed Fast-VVLAD and Fast-VDCNN maintain competitive retrieval performance over diverse ROI queries on two datasets and significantly improve on the retrieval performance (or implementation efficiency) of their respective descriptor variants with a grid spatial search, when dealing with smaller ROI queries. Moreover, improved scale invariance results in competitive retrieval performance to the state-of-the-art on whole image queries.

Chapter 3

Video Classification With CNNs: Using The Codec As A Spatio-Temporal Activity Sensor

In this chapter, we move from the image based retrieval task to video classification. Inline with our thesis objective, we want to improve the accuracy-complexity trade-off by either additionally processing pixel domain representations or supplementing with additional modalities. Whilst image partitioning for retrieval had minimal computational overhead in the previous chapter, most current methods for high-level semantic description in video require additional memory, compute-intensive decoding and additional processing of the pixel domain, in order to generate the optical flow modality [60, 64, 71]. However, it has consistently been shown in recent work [60, 64, 71] that supplementing pixel domain (RGB) frames with such a flow modality improves results for action recognition. Nonetheless, the high resolution & high frame-rate nature of decoded video and the format inflation (from standard to super-high definition, 3D, multiview, etc.) require highly-complex convolutional neural networks (CNNs) that impose massive computation and storage requirements [119].

Notably, because of storage and data-transfer limitations, all camera chipsets and video processing pipelines provide compressed-domain video formats like MPEG/ITU-T AVC/H.264 [120] and HEVC [121], or open-source video formats like VP9 [122] and AOMedia Video 1 instead of uncompressed video. Importantly, video codecs can be tuned at the macroblock (MB) level. For example, the MPEG/ITU-T AVC/H.264 and HEVC codecs divide the input video frames into 16×16 pixel MBs that form the basis for the adaptive inter (and intra) prediction. Inter-predicted MBs are (optionally) further partitioned into blocks that are predicted via motion vectors (MVs) that represent the displacement from matching blocks in previous or subsequent frames.

We therefore propose to consider the video encoder as an imperfect-yet-highly-efficient sensor that derives spatio-temporal activity representations with minimal processing. With regards to the temporal activity, we demonstrate in Section 3.1 that we can obtain MV representations from the compressed bitstream that are highly correlated with optical flow estimates and as such, can replace optical flow as a flow approximation in a multimodal framework. In Section 3.2 we propose a three-dimensional CNN that directly leverages on such MB MV information and compensates for the sparsity of these MB MVs with larger temporal extents. With regards to the spatial activity, we show that selective MB texture decoding can take place based on thresholding of the MB MV information. By superimposing such selectively-decoded MB texture information on sparsely-decoded frames, we obtain spatial representations that are shown to be visually similar to the corresponding fully-decoded video frames. This allows for the parsimonious use of a spatial CNN to augment the classification results derived from the temporal stream. We present results in Section 3.3 with the fusion of this two-stream CNN design on two widely-used datasets, where we show that competitive accuracy is obtained against the state-of-the-art, with end-to-end complexity that is found to be one to three orders-of-magnitude lower than that of all previous approaches based on pixel domain video. Importantly, the complexity gains from using compressed MB bitstreams will increase commensurably as video, multi-view and 3D video format resolutions and frame rates increase to accommodate advances in display technologies. This paves the way for exabyte-scale video datasets to be newly-discovered and analysed over commodity hardware.

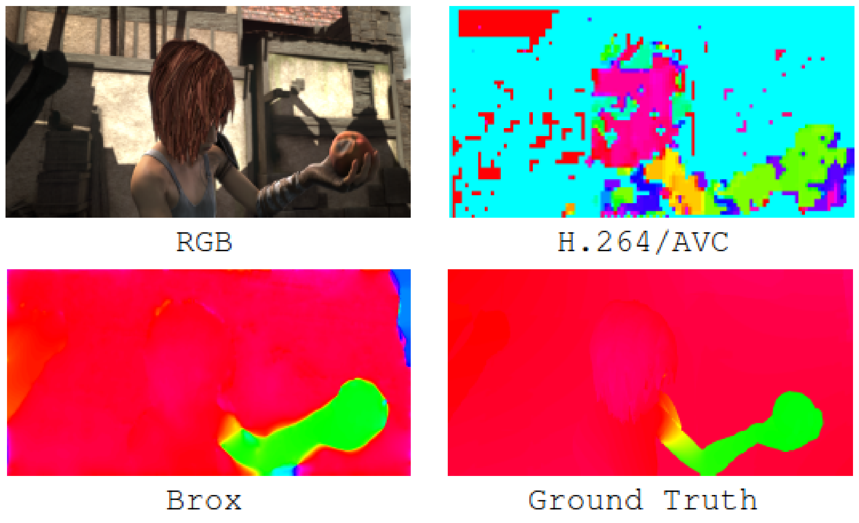


Figure 3.1: RGB frame from the MPI-Sintel dataset and pseudocolored images of the motion information amplitude. The H.264/AVC MB motion vectors are correlated with Brox optical flow extracted from decoded video frames [123][124] and the ground-truth motion available for this synthetic video.

3.1 Selective MB Motion Vector Decoding

Video compression standards like MPEG/ITU-T AVC/H.264, and HEVC rely on motion estimation and compensation as their main method to decorrelate successive input frames. Macroblock motion vectors are derived by temporal block matching and can be interpreted as approximations of the underlying optical flow [125][73], as shown in the example of Figure 3.1.

To derive a temporal activity map from encoded motion compensation parameters, we apply the following steps:

1. Motion vectors are extracted from certain compressed MB information of the utilized video codec[1].

2. If necessary, motion vectors are interpolated spatially to generate a finer rep-

[1]Based on FFMPEG's widely-used `libavcodec` library (which supports most MPEG/ITU-T standards) [126], we make use of the `AVMotionVector` structure (declared within the `avutil.h` header file) as explained in the following. When `libavcodec` attempts to read the compressed bitstream of a video frame using the `av_read_frame()` function, our MB MV extractor calls the `av_frame_get_side_data()` function to extract the MV parameters and place them in the `AVMotionVector` structure. Once the file parsing is completed, the horizontal and vertical coordinates of MVs of each MB within this structure are written in 16-bit integer binary format to disk in order to be used by the proposed 3D CNN. By limiting the bitstream access to solely using this function for the MB MVs, one can achieve the speed gains reported in Section 3.3.

resentation of motion activity in the video, i.e., with resolution corresponding to 8×8 or 4×4 blocks, and also to "fill in" for macroblocks where the video encoder may have used an intra prediction mode.

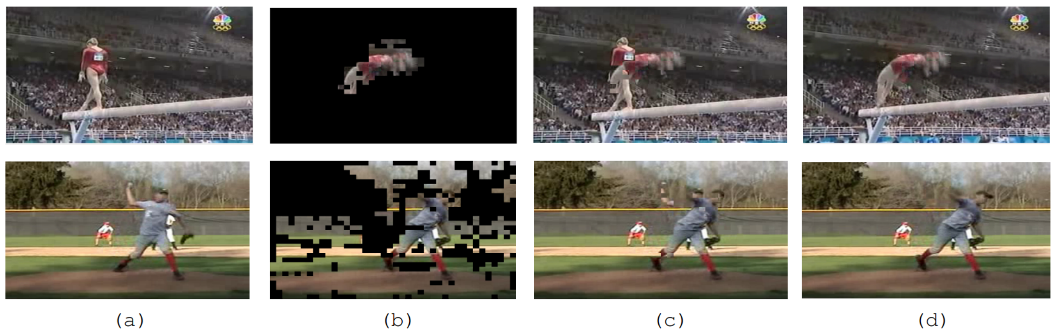


Figure 3.2: Two scenes from UCF-101 with & without camera motion (top & bottom row respectively); (a) Reference frame; (b) Selective decoding of MB texture ($A = 0$); (c) Rendered frame; (d) Fully decoded frame corresponding to the rendered frame.

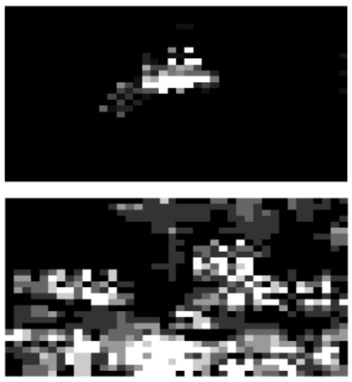


Figure 3.3: MV activity maps corresponding to Figure 3.2(b).

For the spatial stream, we employ selectively-decoded MB texture information using the extracted MVs as activity indicators. We do this by decoding one frame every X frames, with X set to inf, indicating that only the first frame of the video is decoded. In between fully-decoded frames, "rendered" frames can be produced at frame interval R, with $1 \leq R \leq X$. Each rendered frame is initialized as a copy of the immediately preceding fully-decoded frame. Then, texture information at active MB positions is decoded and replaces the initialized values in the corresponding locations in the rendered frame. Two examples of this process are shown in Figure 3.2.

We consider the area within a macroblock to be active when the corresponding MV information exceeds a specified threshold A, $A \geq 0$. As an illustration, Figure 3.3 shows a grayscale activity map derived from the MVs of Figure 3.2(b). To achieve such blockwise selective MB texture decoding via the `libavcodec` library [126], we use the motion vectors from `AVMotionVector` to access `AVFrame::data` and write decoded MB texture data wherever the conditions specified by $\{X,R,A\}$ are met. By increasing the values for $\{X,R,A\}$ we can decrease the frequency of full decoding and selective MB texture decoding in order to achieve any extraction speed desired within a practical application context. In addition, even though it is not explored in this chapter, we can investigate adaptive control of $\{X,R,A\}$ based on the average MV activity level within each video sequence.

3.2 Proposed Framework For Compressed-domain Classification

In this section we describe the proposed framework for training a temporal stream of MB motion vectors extracted directly from the video bitstream and a spatial stream comprising selective (motion-dependent) MB RGB texture decoding, and consider how the two streams can be fused during testing. Figure 3.4 outlines our proposed selective decoding and classification framework.

3.2.1 Network Input

3.2.1.1 Temporal Stream

For our temporal stream input, we extract and retain only P-type MB MVs, i.e., unidirectionally predicted MBs [121, 120]. The standard UCF-101 [58] and HMDB-51 [75] datasets are composed of 320×240 RGB pixels per frame. Therefore, for a frame comprising P-type MBs, a block size of 8×8 pixels results in a motion vector field $\mathbf{\Phi}_\mathrm{T} \in \mathbb{R}^{W_\mathrm{T} \times H_\mathrm{T} \times K_\mathrm{T}}$ of dimension $40 \times 30 \times 2$, where $W_\mathrm{T} \times H_\mathrm{T}$ is the motion vector spatial resolution and the number of channels $K_\mathrm{T} = 2$ refers to the δx and δy motion vector components.

In order to compensate for the low spatial resolution $W_\mathrm{T} \times H_\mathrm{T}$, we take a long

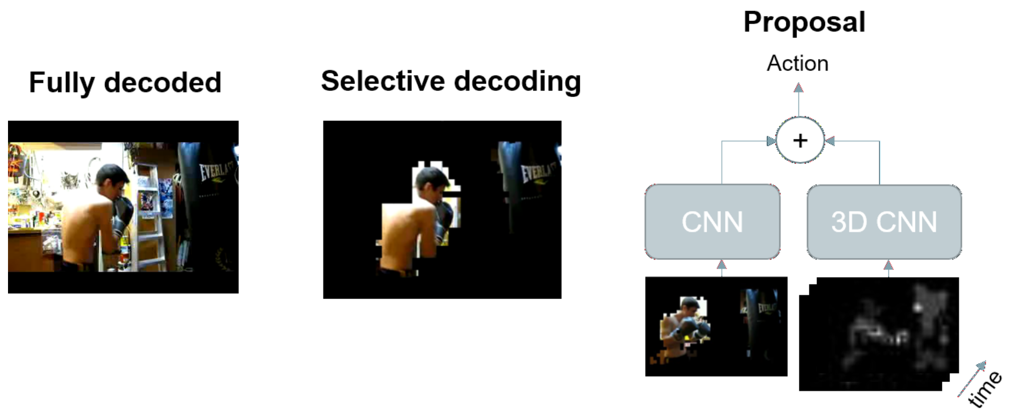


Figure 3.4: Proposed framework for compressed-domain classification: Only macroblocks containing motion are decoded and processed by an RGB CNN. Low resolution and long temporal extent motion vector volumes are encoded by a second 3D CNN and the results are fused across modalities with simply averaging. We generate a rendered texture frame for input to the spatial CNN by replacing the zero (non-decoded) blocks from the selectively decoded frame with texture blocks from a previous fully decoded frame.

temporal extent T over consecutive P frames, with $T > 100$. This is contrary to recent proposals based on high-resolution optical flow [60, 55], which typically ingest only a few frames per input (typically around 10). This is because, even with the latest GPU hardware, a long temporal extent cannot be processed without sacrificing the spatial resolution of the optical flow [60, 55]. On the other hand, given that our MB motion vector input is inherently low-resolution, it is amenable to a longer temporal extent, which is more likely to include the entirety of relevant action that is essential for the correct classification of the video. For example, we have found that the accuracy increases greatly for UCF-101 evaluated on our 3D CNN when moving from 10 to 100 frames, but eventually plateaus when T becomes sufficiently large such that the input extends to almost all P-type frames of the video files of the dataset. Therefore, we fix the temporal extent T to 160, which is roughly the average number of P-frames per video in UCF-101.

In order to make our network input independent of the video resolution, we use a fixed spatial size $N_{\mathrm{T}} \times N_{\mathrm{T}}$ which is cropped/resized from $\mathbf{\Phi}_{\mathrm{T}}$. In this chapter, we set $N_{\mathrm{T}} = 24$, which is large enough to encompass the action region without compromising accuracy, whilst allowing for data augmentation via random crop-

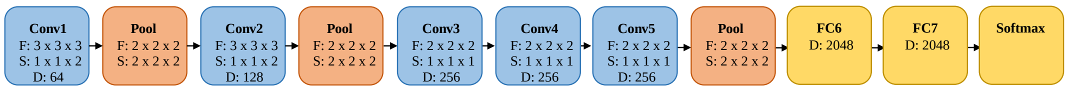


Figure 3.5: 3D CNN architecture: the blue, orange and yellow blocks represent convolutional, pooling and fully-connected layers; F is the filter size for the convolutional layers (or window size for pooling), formatted as width$\times$ height $\times$ time; S is the filter/window stride; D is the number of filters (or number of hidden units) for the convolutional and fully-connected layers.

ping. Furthermore, given it is divisible by eight, the P-frames can be continuously downsampled by a factor of 2 with pooling layers, without requiring any padding. Our final network input $\hat{\mathbf{\Phi}}_{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{T}} \times N_{\mathrm{T}} \times K_{\mathrm{T}} \times T}$ is thus 4D and can be ingested by a 3D CNN. As exemplified in numerous works [55, 56], the advantage of using a 3D CNN architecture with a 4D input, versus stacking the frames as channels and using a 3D input of size $N_{\mathrm{T}} \times N_{\mathrm{T}} \times TK_{\mathrm{T}}$ with a 2D CNN, is that, rather than collapsing to a 2D slice when convolving within the CNN, we preserve the temporal structure during filtering.

3.2.1.2 Spatial Stream

Previous work has shown that stacking RGB frames channel-wise and ingesting such volumes into a 2D CNN does not necessarily improve performance [60, 55]. Indeed, one option is to train a deep 3D CNN on a 4D RGB frame input, which is the proposed configuration for our temporal stream (with MV inputs). Whilst this has been shown to improve performance with RGB frames [56], it is far more computationally expensive to implement when the inputs are at pixel resolution, i.e., typically 224×224 for CNNs trained on ImageNet [27]. Therefore, the complexity of the network in terms of activations and weights quickly becomes unmanageable.

Our approach alleviates these problems by simply ingesting single RGB frames from the video as inputs to a 2D CNN, in order to exclusively model the scene semantics in the image; these comprise geometry, color and background information that can not be extracted from the P-frames directly. For example, in the case of action recognition, the green grass and net and racket texture patterns in the frame could distinguish a sequence as being related to tennis, rather than swimming. Given that such spatial structures and color information tends to be persistent across frames belonging to the same type of scene, we can gain substantial storage and complexity savings by our proposed sparse full-frame decoding and selective superpositioning of MB RGB texture decoding according to the motion activity, as described in Section 3.1 and illustrated in Figure 3.2 and Figure 3.3.

In order to make our input independent of the video resolution, we follow the approach of Simonyan *et al.* [60]. That is, we first resize the RGB frame, such that the smaller side is equal to 256 and we keep the aspect ratio. From the resized frame $\mathbf{\Phi}_{\mathrm{S}} \in \mathbb{R}^{W_{\mathrm{S}} \times H_{\mathrm{S}} \times K_{\mathrm{S}}}$, we crop/resize a fixed spatial size $N_{\mathrm{S}} \times N_{\mathrm{S}}$; $N_{\mathrm{S}} = 224$. Our spatial stream input is thus of size $224 \times 224 \times 3$.

3.2.2 Network Architecture

Our 3D CNN architecture is illustrated in Figure 3.5. All convolutions and pooling are spatiotemporal in their extent. 3D pooling is performed over a $2 \times 2 \times 2$ window with spatiotemporal stride of 2. The first two convolutional layers use 3D filters of size $3 \times 3 \times 3$ to learn spatiotemporal features. With a $24 \times 24 \times 2 \times 160$ motion vector input, the third convolutional layer receives input of size $6 \times 6 \times 2 \times 10$. Therefore, we set the filter size of the third, fourth and fifth convolutional layers to $2 \times 2 \times 2$, as this is sufficiently large to encompass the spatial extent of the input over the three layers whilst minimizing the number of parameters. In order to maintain efficiency when training/evaluating, we also use a temporal stride of 2 in the first and second convolutional layers to quickly downsize the motion vector input; in all other cases we set the stride to 1 for convolutional layers. The temporal downsizing substantially minimizes the number of activations (and thus, the number of floating point operations) in the lower layers. All convolutional layers and the FC6 & FC7 layers use the parametric ReLU activation function [127].

It is important to note that our network has substantially less parameters and activations than other architectures using optical flow. In particular, our 3D CNN stores 29.4 million weights. For comparison, ClarifaiNet [128] and similar configurations that are commonly used for optical-flow based classification [60, 71] require roughly 100 million parameters.

For the spatial stream, we opt for the commonly used VGG-16 [57] architecture, as it is sufficiently deep to learn complex representations from the input frames. The CNN is typically trained on ImageNet [27] for image classification. While we have also obtained similar results with shallower networks, VGG-16 allows for better generalization to larger datasets.

3.2.3 Network Training

We train on the temporal and spatial streams independently, as this permits sequential training on a single GPU and simplifies management of resources such as GPU RAM. It additionally permits evaluation on a single stream for faster runtime. The training details for each stream are as follows.

3.2.3.1 Temporal Stream

We train the temporal stream using stochastic gradient descent with momentum set to 0.9. The initialization of He *et al.* [127] is extended to 3D and the network weights are initialized from a normal distribution with variance inversely proportional to the fan-in of the filter inputs. Mini-batches of size 64 are generated by randomly selecting 64 training videos. From each of these training videos, we choose a random index from which to start extracting the P-frame MB motion vectors. From this position, we simply loop over the P-type MBs in temporal order until we extract motion vectors over T consecutive P frames. This addresses the issue of videos having less than T total P frames, e.g., cases where the video is only a few seconds long. For UCF-101, we train from scratch; the learning rate is initially set to 10^{-2} and is decreased by a factor of 0.1 every 30k iterations. The training is completed after 70k iterations. Conversely, for HMDB-51, we compensate for the small training split by initializing the network with pre-trained weights from UCF-101 (split 1). The learning rate is initialized at 10^{-3} and decayed by a factor of 0.1 every 15k iterations, for 30k iterations.

To minimize the chance of overfitting due to the low spatial resolution of these motion vector frames and the small size of the training split for both UCF-101 and HMDB-51, we supplement the training with heavy data augmentation. To this end, we concatenate the motion vectors into a single $W_{\mathrm{T}} \times H_{\mathrm{T}} \times 2T$ volume and apply the following steps; *(i)* a multi-scale random cropping to fixed size $N_\mathrm{c} \times N_\mathrm{c} \times 2T$ from this volume, by randomly selecting a value for N_c from $N_\mathrm{T} \times c$ with $c \in \{0.5, 0.667, 0.833, 1.0\}$; as such, the cropped volume is randomly flipped and spatially resized to $N_\mathrm{T} \times N_\mathrm{T} \times 2T$; *(ii)* zero-centering the volume by subtracting the mean motion vector value from each motion vector field $\mathbf{\Phi}_\mathrm{T}$, in order to remove possible bias; the δx and δy motion vector components can now be split into separate channels, thus generating our 4D network input $\hat{\mathbf{\Phi}}_\mathrm{T}$. During training, we additionally regularize the network by using dropout ratio of 0.8 on the FC6 and FC7 layers together with weight decay of 0.005.

3.2.3.2 Spatial Stream

We also train the spatial stream independently using stochastic gradient descent with momentum set to 0.9. As with the temporal stream, mini-batches of 64 are amalgamated over 64 randomly selected videos. We take advantage of the transferability of features from image to video classification, and pretrain all layers of our VGG-16 architecture on ILSVRC'12 [28]; all layers are subsequently fine-tuned on the video training sets. The learning rate is initialized at 10^{-3} and decayed by a factor of 0.1. We complete training at 15k iterations.

Again, due to the small training sizes, we risk overfitting during training; therefore we set dropout and weight decay on the first two fully connected layers to 0.8 and 0.005 respectively. We also use a multi-scale random cropping of the resized RGB frame by randomly selecting a value from $N_\mathrm{S} \times d$ with $d \in \{0.857, 1.0, 1.143\}$; the cropped volume is subsequently randomly flipped, spatially resized to $N_\mathrm{S} \times N_\mathrm{S} \times 3$ and zero-centered as per the temporal stream.

3.2.4 Testing

During testing, per video, we generate 2 volumes of temporal size T from which to evaluate on the temporal stream. The starting indices for the volumes are at the first P-frame and at half the total number of P-frames. Per volume, we crop the four corners, the center of the image (and its mirror image) to size $N_\mathrm{T} \times N_\mathrm{T} \times 2 \times T$. In order to generate our prediction for the video, we take the maximum score over all crops. Due to the low resolution and short duration of the HMDB-51 and UCF-101 videos, taking these extra crops and volumes is often redundant as the spatial resolution of the P-frames is low and the temporal extent T of the input is large enough to encompass the entire video duration. However, our approach is better suited to videos "in the wild" and we can afford the use of extra crops due to the low complexity of our 3D CNN.

We evaluate on the spatial stream by extracting only 5 frames from the set per video, albeit with only a single center crop (and its horizontal flip) of size $N_\mathrm{S} \times N_\mathrm{S} \times 3$. In our experiments, we have found this to be sufficient for the case of trimmed action recognition, where most frames are relevant to the associated video label. The frames are extracted at evenly spaced intervals from the video. To generate our prediction, we again compute the maximum score over all extracted frames. In order to produce a final score for the fusion of the two modalities, we simply average their maximum scores, which is equivalent to combining knowledge from the most relevant input in each stream.

3.3 Experimental Results

3.3.1 End-Point Error and Speed of MB MV Extraction and Decoding vs. Optical Flow Methods used in Video Classification

In order to examine the accuracy and extraction time of our approach versus decoding and optical flow estimation, we perform a comparison against the Brox [53] and FlowNet2 [124] optical flow estimation methods that were respectively used (amongst others) by Simonyan *et al.* [60] and Brox *et al.* [124]. Table 3.1 presents the motion field estimation accuracy, measured in terms of end-point error (EPE) on MPI-Sintel, for which ground truth motion flow is also available (see Figure 3.1). Since our CNN architecture downsamples the optical flow before ingestion [60], we measure the EPE for our MV flow estimation at the resolution of our CNN input. Under these settings, Table 3.1 shows that the EPE of our approach is 1.75 to 4.86 times higher than that of optical flow methods. Despite the detrimental accuracy,

Table 3.1: Motion field end-point error (EPE) for the proposed approach, Brox [53] and FlowNet2 [124].

Input	EPE
Proposed, MV	15.26
Brox	8.70
FlowNet2	3.14

our EPE results remain low enough to indicate high correlation with the ground-truth motion flow and the optical-flow based methods. Indeed, the results presented in the following subsections show that the codec MB MV accuracy suffices for classification results that are competitive to the state of the art.

In order to measure flow estimation and decoding speed (with I/O) in terms of frames per second (FPS), we now use video content that corresponds to our video classification tests, i.e., 100 video sequences from UCF-101 (see next subsection for the details of this dataset). All CPU-based experiments were carried out on an Amazon Web Services (AWS) EC2 r3.xlarge instance (Intel Xeon E5-2670 v2 CPU), while all GPU-based experiments were carried out on a AWS EC2 p2.xlarge instance (Tesla K80 GPU). For our selective decoding approach described in Section 3.1, we select values for the decoding interval X that correspond to the settings used in our video classification tests. The results of this experiment are summarised in Table 3.2. In terms of flow estimation speed, our CPU-based MV flow extraction is more than 1500 times faster than FlowNet2 and more than 977 times faster than Brox flow (both running on a GPU), as it does not require video decoding or any optical flow computation. At current AWS pricing[2], GPU instances require more than 2.7 times the cost of CPU instances; as such, our AWS-based implementation has more than 2600 times lower cost. This means that, for the public cloud cost that an optical flow method will process 1 hour of video, our approach will be able to process more than three and a half months of video footage.

In terms of decoding speed, Table 3.2 shows that selective decoding is an order of magnitude faster than the full-frame decoding required for Brox and FlowNet2. We illustrate the influence of selective decoding on the achieved FPS in more detail

[2]AWS EC2 spot pricing, (r3.xlarge vs. p2.xlarge N. Virginia, Sept. 2017)

Table 3.2: Flow estimation and decoding speed results for the proposed approach, Brox [53] and FlowNet2 [124].

Input	Frames Per Second (FPS)	
	Flow Estimation	Decoding
Proposed, $X = 10$	18226 (CPU)	1180 (CPU)
Proposed, $X = 50$	18226 (CPU)	2016 (CPU)
Brox	18.64 (GPU)	168 (CPU)
FlowNet2	12.08 (GPU)	168 (CPU)

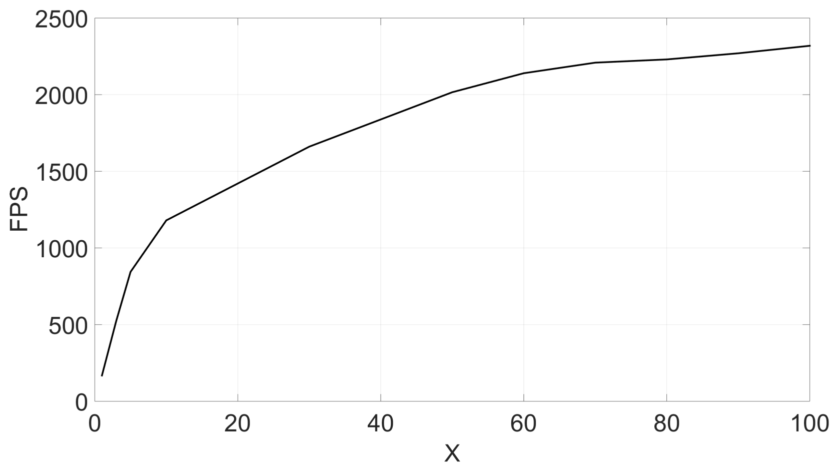


Figure 3.6: Achieved FPS of selective decoding for varying decoding interval X.

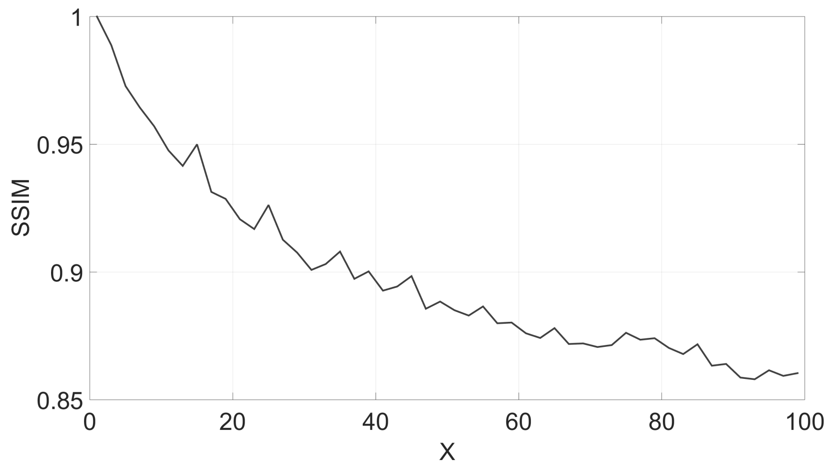


Figure 3.7: Structural similarity index metric (SSIM) for varying decoding interval X.

in Figure 3.6. The results show that the decoding FPS increases rapidly until $X > 50$ and begins to saturate after this point. In order to associate this speed up with a measure for the expected visual quality of the selective decoding and rendering approach, we plot the average structural similarity index (SSIM) [129] for multiple values of X in Figure 3.7, using the fully-decoded video sequences as reference. By combining the two figures, it is evident that, as the decoding speed increases and reaches a saturation at around 2500 FPS, the quality of all rendered frames decreases and plateaus at SSIM values around 0.85. We next assess whether the motion flow accuracy and visual quality allow for high-performant video classification with the proposed CNN-based architectures.

3.3.2 Datasets used for Video Classification

Evaluation is performed on two standard action recognition datasets, UCF-101 [58] and HMDB-51 [75]. UCF-101 is a popular action recognition dataset, comprising 13K videos from 101 action categories with 320×240 pixels per frame, at replay rate of 25 frames per second (FPS). HMDB-51 is a considerably smaller dataset, comprising only 7K videos from 51 action categories, with the same spatial resolution as UCF-101, and at 30 FPS replay rate. Finally, unless stated otherwise, we always cross-validate on the standard three splits for both datasets.

3.3.3 Evaluation Protocol and Results

For each dataset we follow the testing protocol of Section 3.2.4. Each UCF-101 training split consists of approximately 9.5K videos, whereas each HMDB training split has 3.7K videos. We report all single stream feedforward network runtimes without I/O, in order to isolate the efficiency of our proposed architecture. Speed is reported in terms of FPS, which is computed as the number of videos each network can process per second multiplied by the average number of frames per video (we use the average length of UCF-101 videos, i.e., 180 frames [60]). By using FPS as our metric, we account for both the network complexity and the number of inputs processed per video at inference, i.e., the number of crops and volumes taken, as reported in the respective papers. For frameworks where the number of inputs is a function of the video size, we again assume an average video length of 180 frames. All speed results correspond to a batch size of 32 on an AWS EC2 p2.xlarge instance, which comprises a single K80 GPU.

Table 3.3: Classification accuracy and speed (FPS) against state-of-the-art flow based networks. "Proposed 3D CNN" refers to our temporal stream that ingests MB motion vectors.

Framework	Input Size	Accuracy (%)		FPS
		UCF	HMDB	
Proposed 3D CNN	$24^2\times2\times160$	77.2	48.0	3105
TSCNN-Brox [60]	$224^2\times20$	81.2	55.4	185
LTC-Brox [64]	$58^2\times2\times100$	82.6	56.7	<100
LTC-Mpegflow [64]	$58^2\times2\times60$	63.8	–	<100
TSCNN-FlowNet2 [124]	$224^2\times20$	79.5	–	185
EMV-CNN (ST+TI) [71]	$224^2\times 20$	79.3	–	1537

Table 3.4: Complexity of proposed 3D CNN vs EMV-CNN with respect to millions of activations and weights (#A, #W), summed over conv, pool and FC layers in the utilized deep CNN of each approach.

Framework	Complexity	
	#A($\times10^6$)	#W($\times10^6$)
Proposed 3D CNN	4.0	29.4
EMV-CNN [71]	2.0	90.6

3.3.3.1 Temporal Stream

Table 3.3 presents the results of temporal stream CNNs on split 1 of the datasets, for which our method achieves 77.2% and 48.0% on UCF-101 and HMDB-51, respectively. When cross-validating on all three splits for both datasets, our accuracy is higher and we achieve 77.5% on UCF-101 and 49.5% on HMDB-51. It is evident that our approach performs competitively to recent proposals utilizing highly-complex optical flow, whilst minimizing the network complexity via the low number of activations in the lower convolutional layers and small spatial size of the input. As a consequence of the lower resolution inputs and longer temporal extents, our proposal is able to achieve 2 to 30-fold higher FPS in comparison to all other frameworks[3].

The closest competitor is the MV based EMV-CNN method [71], which achieves approximately half the FPS of our approach and therefore warrants further discussion. During test-time, EMV-CNN stacks 10 P or B frames as input to

[3]We remark that Laptev *et al.* [73] made a proposal that uses codec MVs; their method is based on the encoding of such MVs into Fisher vectors (instead of CNNs) to classify video activity. However, that approach is only capable of achieving an accuracy of 46.7% on HMDB [73] at a much lower frame rate (130 FPS) compared to recent CNN methods.

their temporal stream, whereas we stack 160 P-frames per input to our 3D CNN. According to the GOP structure, for every P-frame, EMV-CNN must additionally extract and process 2 B-frames, whereas our 160 P-frame temporal extent typically constitutes processing the entire video in one forward-pass. As such, we only require 12 inputs (2 volumes, 6 crops per volume) to classify a video from UCF-101, whereas EMV-CNN evaluates on 25 inputs per video. Importantly, unlike our approach, EMV-CNN requires upsampled P and B-frame MV fields due to supervision transfer, which leads to reduced MV extraction and CNN processing speed in comparison to our proposal.

In order to go into more detail on the complexity of our CNN against the one proposed within EMV-CNN, we present their network complexity in Table 3.4. The EMV-CNN architecture requires (approximately) 3 times the number of weights (and thus 3 times the memory) of our 3D CNN.

Finally, as discussed in Section 3.2.4, due to the low resolution of our input, taking a large number of crops is redundant in our proposal. Therefore, it should be possible to achieve similar performance even in the case of with one-shot recognition. Indeed, when evaluating our temporal stream on a single center crop, we achieve 76.2% on UCF-101 (split 1) and 46.7% on HMDB-51 (split 1). Such a simplification increases the frame rate to 7452 FPS.

3.3.3.2 Spatial Stream

With regards to the spatial RGB stream produced by the proposed selective decoding, Table 3.5 presents results with two values of X. The network is evaluated and accuracy is subsequently averaged over cross-validation with the three splits for both datasets. The first result ($X = 10$) fully decodes every 10 frames, whilst the second result corresponds to selective decoding and rendering every 10 frames and full decoding every 50 frames. In the latter case, we set the rendering frame interval to $R = 10$ and threshold $A = 0$, i.e., selectively decoding and writing the RGB texture of macroblocks corresponding to non-zero motion vectors once every 10 frames. As the number of crops/volumes that the network evaluates on is fixed

Table 3.5: Classification accuracy and runtime (FPS) against state-of-the-art RGB based networks. For our proposed spatial streams, we fully decode one frame every X frames.

Framework	Input Size	Accuracy (%) UCF	HMDB	FPS
Proposed, $X = 10$	$224^2 \times 3$	79.3	42.4	1228
Proposed, $X = 50$	$224^2 \times 3$	77.7	39.6	1228
TSCNN [60]	$224^2 \times 3$	73.0	40.5	252
SFCNN [55]	$170^2 \times 3 \times 10$	65.4	–	216
LTC [64]	$71^2 \times 3 \times 100$	82.4	–	<100
C3D[56]	$112^2 \times 3 \times 16$	82.3	–	<300

and independent of the fully decoding interval X and rendering frame interval R, the network FPS reported in Table 3.5 is the same for both cases. However, the decoding runtime for $X = 50$ is approximately 1.7 times faster than for $X = 10$ (see Table 3.2). The results demonstrate that the performance drop from selective decoding is marginal and that our spatial stream proposal significantly outperforms TSCNN [60] and SFCNN [55] on UCF-101, whilst performing inference with approximately 5 times higher speed. We achieve this speed by restricting the inputs to single frames and only evaluating on 10 inputs per video, which counterbalances the higher complexity of our pretrained VGG16 network. On the contrary, approaches like LTC [64], TSCNN [60] and C3D[56] evaluate on many more frames and multiple crops per frame, thereby incurring higher computational overhead and significantly lower frame rate for their evaluation process. However, it is worth noting that when comparing our proposed temporal and spatial stream FPS, the temporal stream runs approximately 2.5 times faster, which further motivates training the streams independently, as we can easily allocate more resources to processing the slower stream.

3.3.3.3 Spatio-temporal stream fusion and complementarity in predictions

Table 3.6 presents the summary of the classification performance of our proposed two-stream approach (averaged over the standard three splits) when fusing the spatial and temporal streams. Our two-stream network utilizing selective decoding

Table 3.6: Comparison against state-of-the-art fusion based frameworks. For our proposed two stream networks, the spatial stream ingests one fully-decoded RGB frame every X frames.

Framework	Accuracy (%)	
	UCF	HMDB
Proposed, $X = 10$	89.8	56.0
Proposed, $X = 50$	88.9	54.6
TSCNN (avg. fusion) [60]	86.9	58.0
TSCNN (SVM fusion) [60]	88.0	59.4
CNN-pool [67]	88.2	–
C3D (3 nets)+IDT[56]	90.4	–
LTC[64]	91.7	64.8
EMV + RGB-CNN [71]	86.4	–
IP+SVM [130]	–	59.5
Line Pooling [131]	88.9	62.2
TDD [132]	90.3	63.2

achieves 89.8% accuracy on UCF-101. Overall, our approach is within a few percentile points from the best results reported for both datasets, whilst skipping the complex preprocessing inherent with decoding and optical flow based methods. TSCNN, LTC and Line Pooling all use Brox optical flow in their temporal streams, whilst IP+SVM uses a combination of optical flow, pixel values and gradients for descriptor computation. On the other hand, methods such as C3D+IDT and Line Pooling use trajectory-based descriptor computation in their fusion based frameworks, in addition to deep CNN computation, which adds even further complexity to the classification pipeline. Line Pooling also adopts VGG-16 in their spatial stream as in our case, but forgoes our simpler end-to-end approach for frame pooling and VLAD encoding on an intermediate convolutional layer, which requires additional codebook learning.

Given that our spatio-temporal stream fusion strategy increases the accuracy by 8%-12% in comparison to independent stream evaluation, further investigation of the inference properties of the spatial and temporal CNNs is warranted. The temporal stream ingests inputs of low spatial resolution with long temporal extent, whereas the spatial stream ingests inputs with high spatial resolution but low temporal resolution (single frames). As such, the temporal and spatial stream raters are expected to be more disjoint in their learned representations, which translates

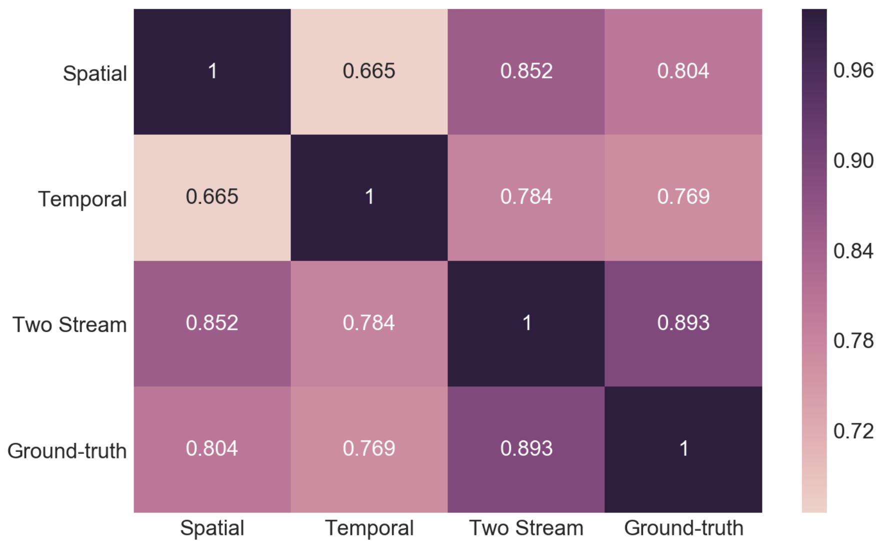


Figure 3.8: Cohen's kappa matrix over all rater combinations for UCF-101 (split 1). For the spatial stream, $X = 10, R = 10$.

to higher information gain and a significant increase in accuracy when their inferences are fused. To quantify their pairwise agreement, we compute Cohen's kappa κ [133] between the raters producing labels for the: temporal stream, spatial stream, two-stream and the ground truth (i.e., "null" model)

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3.1}$$

where p_o is the relative observed agreement amongst raters (equivalent to accuracy) and p_e is a hypothetical probability of random agreement, which is summation of marginal probabilities multiplied between raters. Cohen's kappa ranges from $\kappa = 1.0$ (raters of in complete agreement) to $\kappa \leq 0$ (no agreement amongst raters other than random expectation). We plot the symmetric matrix of κ values over all rater combinations for predictions on UCF-101 (split 1) in Figure 3.8. As expected, there is high inter-rater agreement between the spatio-temporal two-stream architecture and the ground-truth, with $\kappa(\text{two-stream, ground-truth}) = 0.893$. However, there is low inter-rater agreement between the independent spatial and temporal streams, i.e., $\kappa(\text{spatial, temporal}) = 0.665$. This low inter-rater agreement

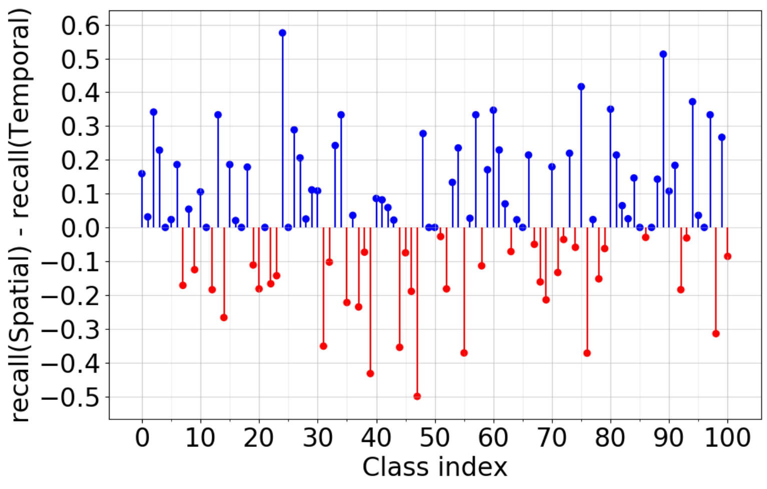


Figure 3.9: Recall difference graph for UCF-101 (split 1). Red line equates to a temporal bias, blue line equates to spatial bias. Classes are in alphabetical order.

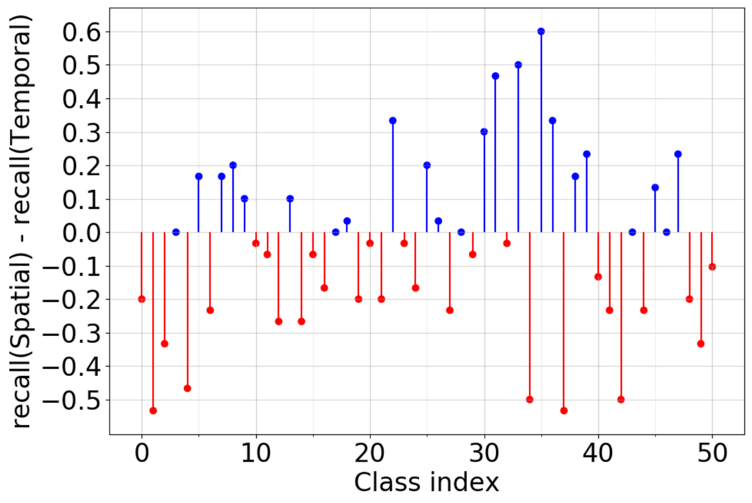


Figure 3.10: Recall difference graph for HMDB-51 (split 1). Red line equates to a temporal bias, blue line equates to spatial bias. Classes are in alphabetical order.

suggests that the temporal and spatial streams learn heterogenous representations that reliably classify different video subsets from the dataset. As such,the spatial and temporal streams complement each other and this translates to a substantial increase in accuracy when fusing the streams during inference, in comparison to inference on either stream independently. Indeed, if these video subsets were very similar to each other, then $\kappa(\text{spatial, temporal})$ would be approximately equal to $\min(\kappa(\text{spatial, ground-truth}), \kappa(\text{temporal, ground-truth})) = 0.769$; however, this is approximately 10% higher than the actual value of $\kappa(\text{spatial,temporal})$ and also turns out to be the average gain obtained by the spatio-temporal fusion.

In order to reinforce this point, we measure the difference in recall values for each class between the spatial and temporal stream, and plot this in Figure 3.9 and Figure 3.10 for UCF-101 and HMDB-51 respectively. If the two streams are in complete agreement, one would expect the recall difference to be close to 0; in other words, the two streams agree on the same video subsets. However, what we observe is that the temporal and spatial streams exhibit distinct biases in terms of class, depending on the nature of the activity. For UCF-101, the temporal stream is favorable for "high activity" classes, such as "high jump" (class 39), "jump rope" (class 47) and "salsa spin" (class 76). Conversely, the spatial stream performs better for "low activity" classes where a scene representation is more informative, such as "cutting in kitchen" (class 24), "rowing" (class 75) and "table tennis shot" (class 89). Intuitively, this means the stream fusion provides better generalization over all classes, or a lower network variance.

3.3.4 Accuracy-Cost Tradeoff

Finally, in order to associate our implementation results with the cost incurred by the fastest competing methods from Table 3.6 (namely, TSCNN (avg. fusion), EMV + RGB-CNN and LTC), we present the AWS deployment cost of each method in Table 3.7. The table shows the cost incurred per component of each method, as well as the total end-to-end cost, C_{tot}. The four components benchmarked in the table are: flow estimation, decoding, temporal stream inference and spatial stream inference, with costs:

$$C_{\text{component}} = \frac{A}{3600} \times \frac{P_{\text{component}}}{F_{\text{component}}}, \tag{3.2}$$

where: A is the average number of frames required for inference in UCF-101 split 1 (180 frames/video $\times$ 3783 videos in split 1), $F_{\text{component}}$ comprises the FPS results reported in Tables 3.1-3.5 for component $\in$ {flow, decode, t-stream, s-stream} of each method, and $P_{\text{component}}$ is the \$/hr cost of the AWS instance used to achieve

Table 3.7: Cost per component and end-to-end cost C_{tot} for our proposed two-stream framework versus competitive frameworks to perform inference on UCF-101 (split 1).

Framework	C_{flow} (\$)	C_{decode} (\$)	$C_{\text{t-stream}}$ (\$)	$C_{\text{s-stream}}$ (\$)	C_{tot} (\$)
Proposed, $X = 10$	0.003	0.053	0.055	0.139	0.250
Proposed, $X = 50$	0.003	0.031	0.055	0.139	0.228
TSCNN (fusion) [60]	9.133	0.375	0.920	0.676	11.103
EMV + RGB-CNN [71]	0.006	0.375	0.111	0.676	1.167
LTC [64]	9.133	0.375	1.702	1.702	12.912

the reported FPS. Specifically, based on the on-demand cost for a p2.xlarge (K80 GPU instance) and r3.xlarge (quadcore CPU) instance:

- $P_{\text{t-stream}} = P_{\text{s-stream}} = 0.9$ \$/hr and $P_{\text{decode}} = 0.333$ \$/hr,
- $P_{\text{flow}} = 0.333$ \$/hr for our proposal and EMV-CNN,
- $P_{\text{flow}} = 0.9$ \$/hr for TSCNN because it requires GPU-based Brox flow estimation.

The total cost C_{tot} to process the UCF-101 (split 1) is:

$$C_{\text{tot}} = C_{\text{flow}} + C_{\text{decode}} + C_{\text{t-stream}} + C_{\text{s-stream}}. \tag{3.3}$$

From Table 3.7, it is evident that the cost of dense optical flow in TSCNN and LTC completely overshadows all other costs. On the other hand, our MV flow incurs less than 0.005% its cost. In addition, the combination of:

- selective decoding (that incurs only 8.3% the cost of full frame decoding with $X = 50$),
- the more efficient CNN processing, and
- the cost C_{flow} of our MV flow estimation being approximately half that of EMV + RGB-CNN due to our temporal CNN only requiring 24×24 crops of the P-frame MV field (while EMV-CNN ingests upsampled P and B-frame MV fields to carry out the supervision transfer [71]),

lead to the proposed method incurring only 20% of the cost of EMV + RGB-CNN for inference.

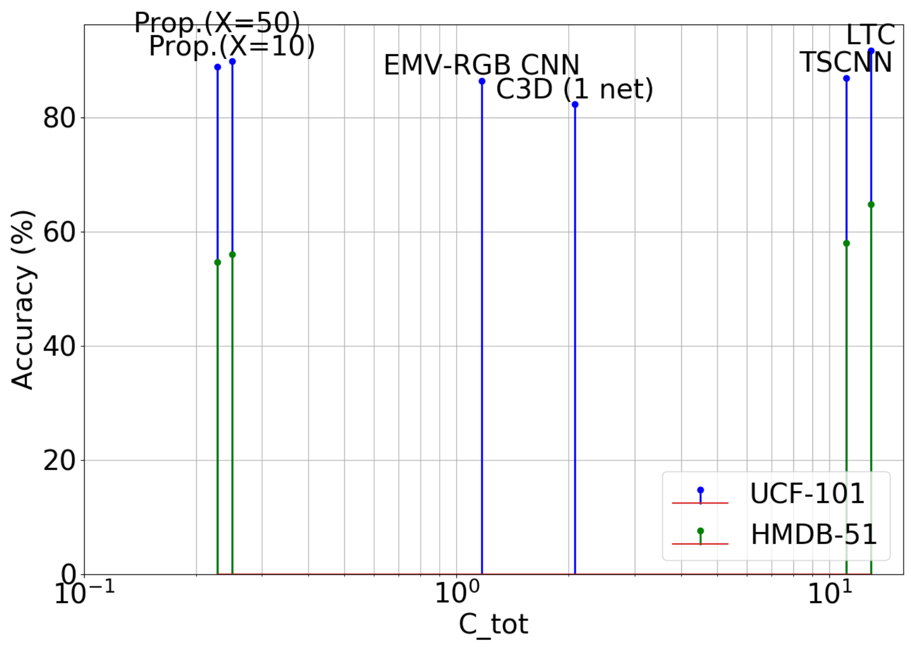


Figure 3.11: Accuracy-complexity plot for our proposed two stream framework with fully decoding rate $X = 10$ and $X = 50$ versus the fastest competing methods. Complexity is denoted in terms of end-to-end cost C_{tot} and plotted on a logarithm scale. We do not plot accuracy for EMV-RGB CNN and C3D (1 net) on HMDB-51 as these are not provided in the paper.

Inline with our thesis objective of improving the accuracy-complexity tradeoff by supplementing with compressed visual modalities, we now plot the accuracy versus end-to-end complexity, which we define in terms of C_{tot}, in Figure 3.11. From the figure, we note that our proposed two stream framework is able to substantially outperform the fastest competing methods in terms of complexity, whilst remaining competitive in terms of accuracy. In particular, whilst LTC is able to slightly outperform our proposal in terms of accuracy, the performance gain is negated by the substantially heavier processing and cost at inference. This cost is largely attributed to the use of optical flow, which contributes to 71% of the total cost; however, we also note that our temporal stream is 30 times cheaper in performing inference than the LTC temporal stream. We also plot C3D (1 net), which only relies on the RGB modality, and show that despite no flow computation, the combination of full decoding and computationally heavy architecture leads to higher cost and worse performance that our proposed framework. Overall, our approach is found to be 5 to 57 times cheaper to deploy on AWS than the most efficient methods from the state-of-the-art in video classification.

3.4 Conclusion

We extend from the unimodal retrieval task to multimodal action recognition, where the RGB frames are conventionally supplemented with a flow modality such as optical flow. We aim to improve on the complexity of dense optical flow generation whilst minimizing the detriment to classification accuracy. Namely, we propose a 3D CNN architecture for video classification that instead utilizes compressed-domain motion vector information for substantial gains in speed and implementation cost on public cloud platforms. We fuse the 3D CNN with a spatial stream that ingests selectively decoded frames, determined by the motion vector activity. Our MV extraction is found to be three orders of magnitude faster than optical flow methods. In addition, the selective macroblock RGB decoding is one order of magnitude faster than full-frame decoding. By coupling the high MV extraction and selective RGB decoding speed with lightweight CNN processing, we are able to classify videos with one to two orders of magnitude lower cloud computing cost in comparison to the most efficient proposals from the literature, whilst maintaining competitive classification accuracy (Tables 3.6 and 3.7). Further refinements of our approach may allow for the first time CNN-based classification of exascale-level video collections to take place via commodity hardware, something that currently remains unattainable by all CNN-based video classification methods that base their training on full-frame video decoding and optical flow estimation. Source code related to the proposed approach is available online, at http://www.github.com/mvcnn.

Chapter 4

Neuromorphic Vision Sensing For CNN-based Action Recognition

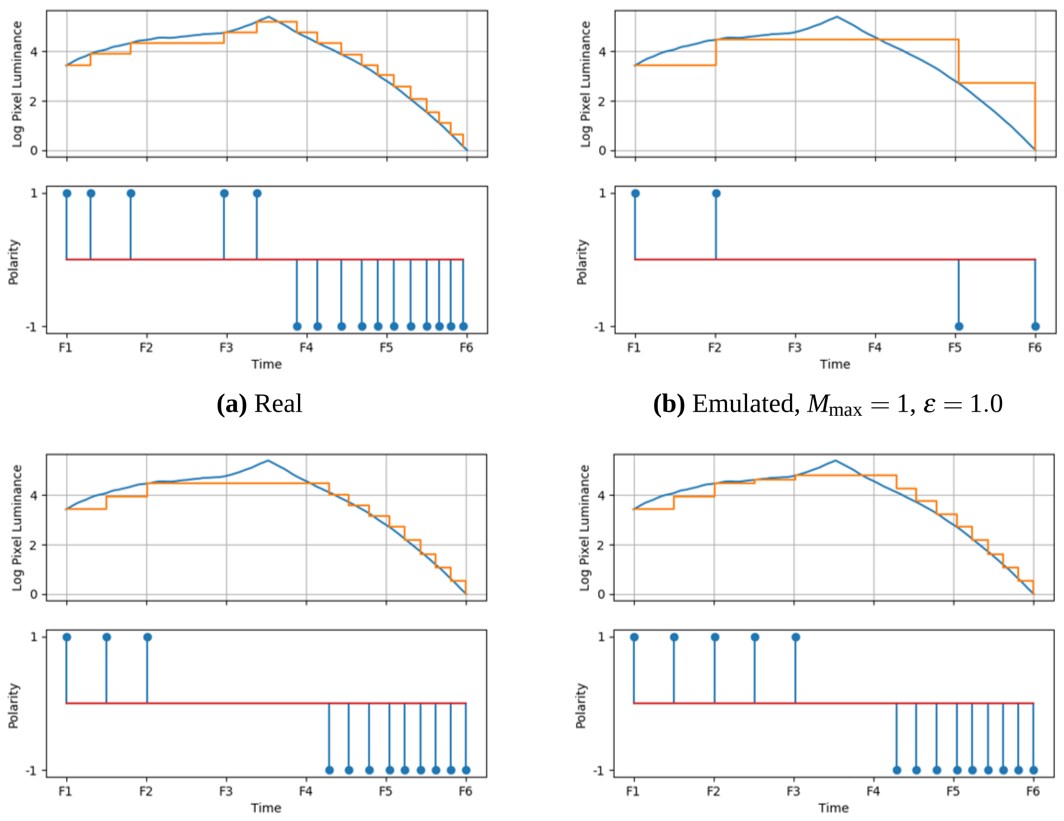
Conventional recognition systems typically classify active pixel sensor (APS) video according to the illustrated human activity (“tennis match”, “cooking”, “people marching”,...). Indeed, both optical flow computation and motion vector extraction considered in the previous chapter have an APS video requirement. Motion vectors require access to the video codec, whereas optical flow requires full extraction and decoding of the APS video frames. However, APS-based video representations are known to be cumbersome for machine learning systems, due to [134]: limited frame rate, too much redundancy between successive frames, calibration problems under irregular camera motion, blurriness due to shutter adjustment under varying illumination, and very high power requirements. Inspired by these observations, hardware designs of neuromorphic sensors, a.k.a., silicon retinas [135, 8], have been proposed recently that improve on the sensing efficiency. Silicon retinas mimic the photoreceptor-bipolar-ganglion cell information flow of biological retinas by producing coordinates and timestamps of on/off spikes in an asynchronous manner, i.e., when the logarithm of the intensity value of a CMOS sensor grid position changes beyond a threshold due to scene luminance changes. Unlike conventional frame-based cameras that tend to blur the image due to slow shutter speed, silicon retinas capture the illumination changes caused by fast object motion and are inherently differential in nature. In practice, this means that neuromorphic vision sensing (NVS) data from hardware like the iniLabs DAVIS and the Pixium Vision ATIS cameras [8, 9, 10, 11] can be rendered to representations comprising up to 2000 frames-per-second (fps), whilst operating with robustness to changes in lighting and at low power, on the order of 10mW. Conversely, a typical APS video camera only captures (up to) 60 fps at more than 100 times the active power consumption and with shutter-induced blurriness artifacts when rapid illumination changes take place. The combination of these advantages makes NVS-based sensing particularly appealing within Internet-of-Things (IoT) and robotics contexts [136, 137, 138], where NVS data would be gathered at very low power and streamed to cloud computing servers for back-end analysis with deep convolutional neural networks (CNNs).

Recent work [135, 136], has advocated possibilities for ingesting NVS data into frame-based deep CNN architectures deployed with state-of-the-art libraries like TensorFlow, in order to gain from the lower power and high frame-rate inherently available with an NVS camera. However, most activities relate to low-cost on-board processing for robotics and guidance systems and do not consider higher-level tasks like human action recognition or semantic scene analysis. In addition, all existing work is hampered by the lack of annotated NVS training data [135, 139, 140]. To mitigate the latter issue, several proposals recorded annotated APS video datasets under controlled conditions [141, 142, 134, 143], i.e., video frames displayed in a monitor under controlled frame-rate and brightness/contrast conditions and recorded with a DVS camera. Such experimental approaches are very valuable as they provided the first available annotated video datasets in NVS format. However, their scale-up to larger datasets is hampered by: *(i)* the recording being potentially affected by environmental and monitor conditions (e.g., lighting, monitor flicker, vibrations, etc.) and being specific to the utilized NVS camera (i.e., needing to be repeated as new generations of NVS hardware emerge); *(ii)* highly-accurate synchronization being required between the played-out video frames and the corresponding NVS because of drift between the playout device timing and the DVS camera timestamping (especially as the dataset grows in size); *(iii)* the slow passage of real time and the expense of the physical hardware involved (e.g., when considering that APS video datasets like YouTube-8M [144] and Kinetics [145] comprise hundreds of thousands of clips). To this end, Katz *et al.* [146] and Mueggler *et al.* [139] proposed models to generate neuromorphic spiking activity using piecewise linear interpolation of the pixel intensity given by successive rendered images. The recently proposed PIX2NVS framework [147] and work by Furber *et al.* [148] provide for parametric software-based solutions for the conversion of pixel-based videos to neuromorphic spike events. While initial results have been presented with a small number of videos and a limited set of conversion options, there is currently no validation of performance for deeper semantic tasks like action recognition.

In this chapter we address these issues. First, in Section 4.1 we briefly explain the NVS camera operation that we are trying to emulate. In Section 4.2 we address the lack of training data for NVS based recognition by improving on recently proposed emulator frameworks [147, 148] that can generate emulated neuromorphic vision streams from any APS video format. Notably, we introduce new options for spike event generation, reference frame updating and frame grouping, resulting in emulated frames that can be ingested directly by a convolutional neural network. We also tune the parameters of an NVS emulator framework configured with our proposed additions, in order to minimize the domain shift between real and emulated spike event distributions. As such, by converting pixel-domain video datasets such as UCF-101 and HMDB-51 with the improved emulator, we are able to provide an abundance of data on which to train a CNN in the NVS domain. For the action recognition task, we show in Section 4.3 how such an emulator can be embedded into a larger multi-modal teacher-student framework, where we capitalize on the availability of optical flow labelled data by employing a pre-trained optical flow stream as a teacher network to transfer knowledge to the emulated NVS student network. Contrary to recent work [71] that considers homogeneous transfer between flow domains (optical flow to motion vectors), we propose an heterogeneous or multimodal transfer from the flow to NVS domains. The motivation is that: *(i)* despite the difference in modality, there is still more information embedded in the (softened) logits of a pre-trained optical flow teacher network about inter-class dependencies than in ground-truth labels, which we can leverage on in a teacher-student framework; *(ii)* the emulated NVS events and optical flow are both derived from intensity variation between frames. Finally, in Section 4.4 we evaluate the performance of the framework on emulated NVS data and compare with recent APS-based methods on standard action recognition datasets. Our results show that, for the first time, an NVS-based CNN approaches the accuracy of complex methods based on optical flow extraction from APS video at lower CNN complexity, thus making NVS inputs a very competitive alternative to APS-based cameras and optical flow extraction for low-power IoT and robotics applications requiring action recognition.

4.1 Neuromorphic Vision Sensing Camera Operation

We illustrate in Figure 4.1(a) the operation of an NVS camera in generating change detection events. Contrary to an APS video camera, where visual information is acquired and recorded at a constant sampling rate (i.e., the framerate) over the pixel array, each individual pixel in an NVS camera optimizes its own sampling based on the luminance variation that it receives. Specifically, the NVS camera uses level-crossing sampling [149], where the luminance of each pixel is tracked continuously and independently and a change detection event or spike is only recorded when the shift in luminance is greater than a defined threshold. As such, the events are generated and timestamped asynchronously over the pixel array, and the e-th event is recorded with its spatio-temporal position $(x_e^{\text{exp}}, y_e^{\text{exp}}, t_e^{\text{exp}})$ and polarity, $P_e^{\text{exp}} = \pm 1$. The polarity indicates whether the log luminance has increased or decreased by the preset threshold (ON or OFF). The camera outputs a stream of events over time, which resembles the signals transmitted between photoreceptors and ganglion cells in the human visual system. The low power of the NVS camera is attributed to the per-pixel sampling rates and the fact that a pixel remains idle to avoid unnecessarily



(a) Real

(b) Emulated, $M_{\text{max}} = 1$, $\varepsilon = 1.0$

(c) Emulated, $M_{\text{max}} = 5$, $\varepsilon = 1.0$

(d) Emulated, $M_{\text{max}} = 5$, $\varepsilon = 0.5$

Figure 4.1: (Best viewed in color) Graphs of log pixel luminance and polarity versus time for an example pixel intensity variation. (a) is representative of the real NVS camera operation, whereas (b)-(d) represent our improved emulator framework with varying parameter settings. The blue curve on the luminance-time plots shows the continuous analog luminance varying with time. The orange step function shows the points where the change in luminance exceeds the preset threshold $T_{\text{map}} = 0.4$. These are translated to on and off change detection events which we plot as event streams on the polarity-time plots. We mark the video frame timestamps F1-6 on the time axis.

acquiring redundant information when there is no change in luminance.

4.2 Improving Neuromorphic Emulator Frameworks

Conventional neuromorphic emulator frameworks [147, 148] retrieve a pixel-domain video that may be encoded and wrapped in any standard format container (e.g., MP4, MKV, etc.) and extract a series of pixel-domain video frames. The emulator generates event tuples $E_e^{\text{emu}} = \langle x_e^{\text{emu}}, y_e^{\text{emu}}, t_e^{\text{emu}}, P_e^{\text{emu}} \rangle$ over the video sequence:

analogous to the NVS camera, the first two parameters correspond to the spatial coordinates of the eth event within a frame, and the event polarity (± 1, representing ON or OFF) is indicated by P_e^{emu}. All event tuples are stored in a text file and/or in an AEDAT stream [135]. The aim of the emulator is to produce emulated NVS events that are as similar as possible to the ones that would have been generated if the equivalent scene would have been captured with an NVS-generating device like iniLabs DAVIS or Pixium ATIS hardware. Given that the resolution of event timestamps is restricted to that of the input video framerate, its use is in recognition and classification tasks that involve activities lasting between 0.1-10 seconds, such as recognition of human actions or scene classification.

Figures 4.1(b)-(d) represent our improved emulator framework with varying emulator parameters. Contrary to the real NVS camera in Figure 4.1(a), which operates on the continuous analog luminance values (blue curve) for event detection, the emulator operation is discrete in terms on of both luminance and time. Specifically, when converting APS video to NVS events, we only have access to the video frames (marked at F1-6 on the figure). In the case of luminance, this means we are restricted to the digital values (0-255) as derived from the RGB channels and not the analog values that the NVS camera would receive. On the other hand, in the case of time, this means we can not track the pixel intensity variation between frames - as such, we propose methods for accurate event approximation between frames. Namely, in the following parts of this section, we introduce additional options in spike event generation and reference frame updating along with their associated parameters for improving the similarity of the emulated NVS events to that of a conventional NVS camera.

4.2.1 Spike Event Generation

We generate emulated spikes through colocated differencing [147] of the nth video frame log-scaled luminance values $l_{x,y}[n]$ with the previous reference frame luminance values, $\bar{l}_{x,y}[n-1]$.

$$d_{x,y}[n] = l_{x,y}[n] - \bar{l}_{x,y}[n-1] \tag{4.1}$$

The eth spike corresponding to the nth frame (out of $e_{\text{tot}}[n]$ spikes detected in that frame) is generated iff $|d_{x,y}[n]| \geq T_{\text{map}}$, with T_{map} a preset threshold; in such a case, the polarity of the spike is: $P_e^{\text{emu}} = \text{sgn}(d_{x,y}[n])$ and the coordinates of the spike are $(x_e^{\text{emu}}, y_e^{\text{emu}}) = (x,y)$.

4.2.1.1 Local Inhibition

As correlation in small spatial neighborhoods of natural images is known to be high [150], we assume that neighboring pixels transmit redundant information. We therefore consider emulating local inhibition in the framework by applying a local maximum on non-overlapping patches of the differences $d_{x,y}[n]$, with dimensions $B \times B$:

$$(x^*, y^*) = \arg\max(d_{(x+s_x),(y+s_y)}[n]) \quad \forall s_x, s_y \in \{0, 1, \ldots, B\} \tag{4.2}$$

Hence, when local inhibition is enabled, within each patch, we only keep $d_{x,y}[n]$ for the (x^*, y^*) positions corresponding to the locally maximum difference values.

4.2.1.2 Number of Spikes

We denote the number of spikes to generate per position per frame as $M_{x,y}[n]$. Standard emulator frameworks [147] only generate a single spike per position per frame; i.e., $M_{x,y}[n] = 1$. However, in this way, the number of generated spikes per position are limited by the frame rate; this does not encapsulate cases where the pixel intensity difference $d_{x,y}[n]$ is high [148] relative to the preset threshold T_{map}. We address this issue by allowing more than one spike to be generated between frames following an approach similar to Furber *et al.* [148]. First, we assign $M_{\max}$ as the maximum number of spikes per position per frame. We then compute $M_{x,y}[n]$, as:

$$M_{x,y}[n] = \min\left(M_{\max}, \left\lfloor \frac{d_{x,y}[n]}{T_{\text{map}}} \right\rfloor\right) \tag{4.3}$$

and these additional spikes are assigned a timestamp t_e by linearly interpolating between the timestamps of the $n-1$-th and n-th frames [147]. We can visualize the effect of increasing the number of spikes by comparing Figures 4.1(b) and (c). In

Figure 4.1(b), the maximum number of spikes M_{max} is set to 1; therefore, only a single spike will be generated per frame if the intensity variation exceeds the preset threshold T_{map}. On the other hand, increasing M_{max} to 5 allows us to generate more spikes for larger intensity variation, which improves the similarity of the emulated event stream to the real event stream in Figure 4.1(a).

4.2.2 Reference Frame Update

The default option of many NVS emulation tools like PIX2NVS [147] is to update the reference frame $\bar{l}_{x,y}[n]$ based on the log-scaled value computed during the conversion process, i.e., $\bar{l}_{x,y}[n] = l_{x,y}[n]$. However, the above reference update method only refers to the current frame and does not consider the transient response of neuromorphic sensors. Similar to Furber *et al.* [148], the reference frame update can alternatively be modeled by considering an exponential moving average over past frames. This provides a potentially more accurate representation by accounting for the capacitive memory of neuromorphic sensors. We define the new reference update for the nth frame as:

$$\bar{l}_{x,y}[n] = \begin{cases} l_{x,y}[0], & \text{if } n = 0 \\ \varepsilon l_{x,y}[n] + (1-\varepsilon)\bar{l}_{x,y}[n-1], & \text{otherwise} \end{cases} \tag{4.4}$$

where $0 < \varepsilon < 1$ is the update rate, which can be tuned to match the capacitive properties of a neuromorphic sensor. Current NVS cameras only update the log-scaled values of recently-detected positions. If we follow this approach then we only update the reference frame for positions (x,y) where a spike is detected, i.e., $\forall(x,y) \in \{(x_1,y_1),\ldots,(x_{e_{\text{tot}}},y_{e_{\text{tot}}})\}$. We can visualize the effect of the exponential moving average by comparing Figures 4.1(c) and (d). In Figure 4.1(c), we set the parameter to 1.0, which equates to no moving average ($\bar{l}_{x,y}[n] = l_{x,y}[n]$). We note that the emulator is unable to detect any events between video frame timestamps F2 and F3, even events are recorded by the NVS camera in Figure 4.1(a). We are able to subtly improve the event stream approximation by decreasing ε to 0.5. In this chapter, we fix the parameter $\varepsilon = 0.5$.

4.2.3 Frame Generation

We can aggregate the spikes into a single NVS frame correspondence to the n-th APS video frame by summing the polarities at each spatial position (x,y) for spike events falling between the $n-1$-th and n-th video frame timestamps. For the n-th video frame, we refer to the summed emulated polarities per position as $P_{\Sigma}^{\text{emu}}[n]$.[1] While this frame grouping is artificial, it allows for the introduction of quantization in time [11, 10] and the use of CNNs for the recognition [136].

4.2.4 Validation of the Emulation by Quantifying the Domain Shift

We quantify the domain shift between generated events and real NVS events with Earth Mover's distance (EMD), by incorporating our proposed options into the PIX2NVS framework [147]. EMD has been proposed as the means to quantify the dissimilarity between two signatures [151], which is defined as the minimum "cost" that must be paid to transform one signature into the other, where there is a "ground distance" between the basic features that are aggregated into the signature. Thus, EMD is an effective method of measuring the domain shift between the real and emulated spike datasets. Essentially, we want to find the flow $\mathbf{G} = [g(i,j)]$ between the aggregated spikes in emulated and real event sets $\hat{\boldsymbol{E}} = \{< x_1^{\text{emu}}, y_1^{\text{emu}}, P_{\Sigma,1}^{\text{emu}} >, \cdots < x_I^{\text{emu}}, y_I^{\text{emu}}, P_{\Sigma,I}^{\text{emu}} >\}$ and $\hat{\boldsymbol{R}} = \{< x_1^{\text{exp}}, y_1^{\text{exp}}, P_{\Sigma,1}^{\text{exp}} >, \ldots, < x_J^{\text{exp}}, y_J^{\text{exp}}, P_{\Sigma,J}^{\text{exp}} >\}$ that minimizes the "work optimization" problem stated below:

$$\begin{aligned}
\text{minimize} \quad & W(\hat{\boldsymbol{E}}, \hat{\boldsymbol{R}}, \mathbf{G}) = \sum_{i=1}^{I}\sum_{j=1}^{J} g(i,j)d(i,j) \\
\text{subject to} \quad & g(i,j) \geq 0, \quad 1 \leq i \leq I, 1 \leq j \leq J, \\
& 1 \leq j \leq J : \sum_{i=1}^{I} g(i,j) \leq \left|P_{\Sigma,j}^{\text{exp}}\right|, \quad 1 \leq i \leq I : \sum_{j=1}^{J} g(i,j) \leq \left|P_{\Sigma,i}^{\text{emu}}\right|, \\
\text{and} \quad & \sum_{i=1}^{I}\sum_{j=1}^{J} g(i,j) = \min\left(\sum_{\forall i}\left|P_{\Sigma,i}^{\text{emu}}\right|, \sum_{\forall j}\left|P_{\Sigma,j}^{\text{exp}}\right|\right)
\end{aligned} \tag{4.5}$$

[1] We similarly denote the sum of real event polarities between the $n-1$-th and n-th video frame timestamps as $P_{\Sigma}^{\text{exp}}[n]$.

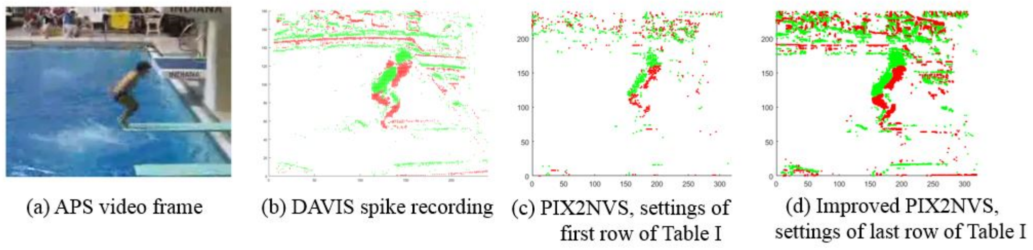


Figure 4.2: (Best viewed in color) Comparison of emulator conversion for (c) original PIX2NVS and (d) PIX2NVS with our proposed improvements against (b) the spike events recorded with a DAVIS camera . Green/red points correspond to +1/-1 (or ON/OFF) spike polarity.

where P_Σ^{emu} and P_Σ^{exp} are the summed polarities per position for the emulated and experimental (real) NVS events respectively. After initializing the flow uniformly, this optimization problem can be solved using linear programming [151]. We set the "ground distance" to
$d(i,j) = \left|(x_i^{\text{emu}}, y_i^{\text{emu}}) - (x_j^{\text{exp}}, y_j^{\text{exp}})\right|$, where $(x^{\text{emu}}, y^{\text{emu}})$ and $(x^{\text{exp}}, y^{\text{exp}})$ represent the spatial positions of the emulated and experimental summed events respectively. EMD can thus be interpreted as the minimum work required to "transport" the polarity between emulated and real spike event sets $\hat{\boldsymbol{E}}$ and $\hat{\boldsymbol{R}}$ such that both sets are evenly distributed, normalized by the total optimum flow $\mathbf{G}_{\text{opt}}$, i.e.,

$$\text{EMD}(\hat{\boldsymbol{E}}, \hat{\boldsymbol{R}}) = \frac{\sum_i \sum_j g_{\text{opt}}(i,j) d(i,j)}{\sum_i \sum_j g_{\text{opt}}(i,j)} \tag{4.6}$$

The size of the flow matrix $\mathbf{G}$ grows exponentially with the number of spike events and, as such, becomes non-trivial to compute. We are able to partially offset the complexity by dividing each frame into a grid of spike blocks and computing the EMD between spatially corresponding blocks of real and emulated spikes. Denoting the subset of emulated and real spikes in block k as $\boldsymbol{E}_k^{\text{S}} \in \hat{\boldsymbol{E}}$ and $\boldsymbol{R}_k^{\text{S}} \in \hat{\boldsymbol{R}}$ respectively, the distance D for frame n is now computed over $1 \leq k \leq K$: $D(n) = \sum_{k=1}^{K} \text{EMD}(\boldsymbol{E}_k^{\text{S}}, \boldsymbol{R}_k^{\text{S}})$. In this chapter, we set $K = 16$. The final distance for a video sequence is computed by averaging $D(n)$ over all frames.

We report the obtained distances for an indicative set of parameters in Table 4.1, by using the real UCF-50 NVS recordings from Mueggler *et al.* [139] as ground

Table 4.1: EMD (smaller is better) between UCF-50 real and emulated spikes, w.r.t. our different parameter settings for our improved PIX2NVS framework: `new=TRUE` enables the selective update when a spike is detected (Section 4.2.2) and `exp=TRUE` enables the exponential updating of (4.4). The original emulator provides results corresponding to the top two rows of the table.

		$\boldsymbol{\theta}$				
$T_{\log}$	B	$M_{\max}$	T_{map}	`new`	`exp`	EMD
20	2	1	0.4	`False`	`False`	2.448
0	2	1	0.2	`False`	`False`	2.597
0	2	1	0.2	`False`	`True`	2.467
0	2	1	0.2	`True`	`True`	2.434
0	1	3	0.2	`True`	`True`	2.307

Table 4.2: Accuracy on emulated events on UCF-101 (split 1) without teacher supervision for original PIX2NVS settings and improved PIX2NVS with our proposed options enabled. Accuracies are reported on a *single shot* of 8 frames.

			$\boldsymbol{\theta}$				
	$T_{\log}$	B	$M_{\max}$	T_{map}	`new`	`exp`	Accuracy(%)
Original	20	2	1	0.4	`False`	`False`	59.3
Improved	0	1	3	0.2	`True`	`True`	71.0

truth. The first two rows correspond to the original PIX2NVS framework, without our proposed improvements. The bottom row has the smallest distance and thus should represent the optimal parameter set $\boldsymbol{\theta}^*$ for the improved PIX2NVS framework, where both our proposed selective and exponential updates are enabled and we increase the threshold for the number of spikes generated at each position $M_{\max}$. In this case, we expect the emulated event distribution is closest to the real event distribution. We validate the improvement from the original settings by visually assessing the similarity between emulated frames generated with the parameter settings of the first and last rows of Table 4.1, as compared to real NVS recordings in Figure 4.2. Evidently, with our proposed modifications, the emulated frames are more visually similar to the real NVS frames. Finally, we validate the improvement in terms of accuracy on emulated events on UCF-101 (split 1) in Table 4.2, using the network configuration in Section 4.3 but *without* teacher supervision (i.e., $\alpha = 0, \beta = 1$). The accuracy for the improved PIX2NVS with our proposed modifications is 12% higher than for the original framework. Therefore, we fix the parameters to this optimal set for the remainder of the chapter.

4.3 Teacher-Student Framework

In order to validate NVS inputs as a low-cost activity-based alternative to optical flow traditionally used for action recognition, we propose to embed the improved PIX2NVS emulator with the best parameter set $\boldsymbol{\theta}^*$ into a teacher-student framework based on knowledge distillation [152], where we essentially transfer knowledge from a pre-trained optical flow teacher network to the NVS student network by drawing pairwise correspondences and minimizing the cross entropy between the output distributions. The framework is illustrated in Figure 4.3 for the training and inference stages. After training the student model, the emulator component of the framework can potentially be replaced by an NVS camera during inference, to perform action recognition on either real or emulated events directly on the student, without the use of optical flow.

4.3.1 Cost Function and Training

We leverage on the recently introduced large-scale Kinetics [74] action recognition dataset. Our choice of architecture is a variant of the Inception-3D (I3D) [74] CNN, which is essentially Inception-Vl with inflated spatio-temporal filters Notably, we replace the final pooling layer with a spatio-temporal global average pooling. Our implementation first involves initializing the optical flow I3D with the Kinetics trained weights and then fine-tuning on the target action recognition dataset, such as UCF-101. Secondly, we use the improved PIX2NVS framework to extract the emulated NVS events from RGB video frames, which provides us with training data correspondences for the NVS domain. Next, we initialize the student NVS CNN with the teacher weights. It is worth noting that our NVS inputs are only single-channel, whereas flow is dual-channel for the δx and δy components respectively; in order to apply an exact copy of our teacher network as the student, we simply replicate the NVS input channel-wise. The teacher network is now fixed and we train the student using a two-term cross-entropy loss for NVS frame volumes

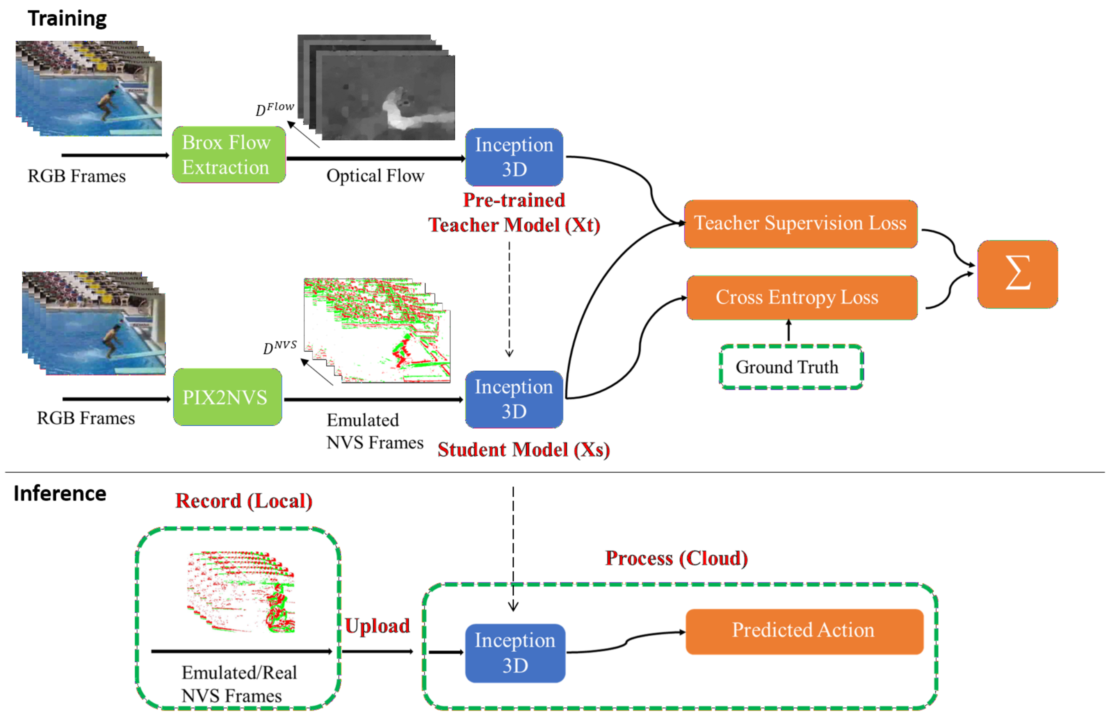


Figure 4.3: Teacher-student framework using the improved PIX2NVS emulator.

$\boldsymbol{v}_s \sim \mathbb{V}_s$, teacher flow volumes $\boldsymbol{v}_t \sim \mathbb{V}_t$ and labels $y \sim \mathbb{Y}$:

$$L = -\beta \mathbb{E}_{(\boldsymbol{v}_s, y)\sim(\mathbb{V}_s,\mathbb{Y})} \sum_{k=1}^{K} 1_{[k=y]} \log(p(\boldsymbol{v}_s)) - \alpha T^2 \mathbb{E}_{(\boldsymbol{v}_s,\boldsymbol{v}_t)\sim(\mathbb{V}_s,\mathbb{V}_t)} q(\boldsymbol{v}_t, T) \log(p(\boldsymbol{v}_s)) \tag{4.7}$$

The first term represents the standard cross-entropy loss between the softmax output of the student network $p(\boldsymbol{v}_s)$ and a one-hot encoded vector derived on the ground truth labels y. The second term is the teacher supervision cross-entropy loss between the teacher softmax output $q(\boldsymbol{v}_t, T)$ and $p(\boldsymbol{v}_s)$. The temperature T is the parameter that scales the logits z_i of the teacher network, such that $q_i(\boldsymbol{v}_t, T) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$. This softens the teacher distribution over classes, which can exemplify the class inter-dependencies for the student to learn a more informative representation. We treat the parameters α and β as simply binary coefficients on the loss terms, responsible for enabling/disabling the label cross entropy loss and teacher supervision loss respectively.

We train both the teacher and student CNNs with momentum and a decay rate of 0.9. Every convolutional layer is followed by batch normalization [111] with a decay rate of 0.9. The networks are trained with an initial learning rate of 0.01 and the learning rate is decayed when the validation accuracy saturates. Our preprocessing constitutes a per-frame normalization and extracting a distorted bounding box crop from the original resized frame with a random horizontal flip. Importantly, we only infer on a single shot, extracted at the point of maximum motion activity, in order to minimize the temporal footprint during inference. This point n^* is localized by using the emulated NVS frames as an activity sensor and finding the n^*-th frame with the maximum sum over absolute summed polarities P_Σ:

$$n^* = \underset{n}{\operatorname{argmax}} \left(\sum_{x,y} |P_\Sigma[n](x,y)| \right) \tag{4.8}$$

4.3.2 Ablation Study

Table 4.3 represents a basic exploration over the parameters α, β and T in (4.7). We note that we only ingest inputs $\boldsymbol{v} \in \mathbb{R}^{H\times W\times D\times C}$ of size $224 \times 224 \times 8 \times 2$ for both the flow and NVS streams, in order to speed up convergence. As configured, the optical flow CNN achieves 84.4% on a single input. The emulated NVS CNN, when trained without the teacher supervision loss and inferring on a single shot of emulated NVS frames, achieves 71.0%. This shows that incorporating the teacher supervision loss results in a substantial accuracy increase, with optimum accuracy attainable when the teacher logits are additionally softened with $T = 2$. Increasing T beyond 2 causes a decrease in accuracy, as the output begins to converge to a more uniform distribution. For the remainder of the chapter, we fix $\alpha = 1$, $\beta = 1$ and $T = 2$.

4.4 Comparison with APS-based Methods

Finally, we compare the proposed framework against current state-of-the-art APS-based methods. Given the combination of global average pooling and 3D convolutional layers in our proposal, this inherently means that the number of weights per CNN layer is not a function of the input temporal resolution D. Therefore, we initially train the optical flow teacher CNN on a larger input temporal resolution

Table 4.3: Recognition accuracy on UCF-101 (split 1) for the teacher and student, when varying parameters α, β and T of (4.7). Accuracies are reported on a *single shot* of 8 frames. For the student, both training and inference is performed on emulated NVS events generated by the improved PIX2NVS emulator.

	α	β	T	Accuracy(%)
Teacher	-	-	-	84.4
Student	0	1	-	71.0
	1	0	1	73.1
	1	1	1	75.9
	1	1	2	77.0

$D^{\text{Flow}} = 32$. We then initialize NVS student CNN with the flow pre-trained weights, but reduce the input temporal resolution to $D = 16$ when training with teacher supervision and during inference. In this way, we transfer long temporal dependencies encoded in the teacher output during training, whilst minimizing the student complexity, without directly modifying the student architecture as proposed by Hinton *et al.* [152]. For our final implementation of the trained NVS student, we therefore infer only on a single shot of emulated frames with size $224 \times 224 \times 16 \times 2$.

As discussed in the previous chapter, conventional APS-based methods typically use a combination of RGB frame and highly complex optical flow inputs during training and inference. Conventional NVS cameras, such as the iniLabs DAVIS240C are equipped with an onboard RGB camera. The DAVIS240C camera has an array size of 240×180 pixels with an APS bandwidth of 35 FPS. Therefore, we propose to combine the trained NVS student with an RGB stream during inference, which we restrict to ingesting inputs $\boldsymbol{v}^{\text{RGB}} \in \mathbb{R}^{H \times W \times D \times C}$ of size $224 \times 224 \times 8 \times 3$. Our choice of architecture is I3D, pre-trained on Kinetics and fine-tuned on the target action recognition dataset. We train the RGB stream independently with momentum set to 0.9 and pre-processing with a distorted bounding box crop, as in Section 4.3.1. During inference, we infer on a single shot of RGB frames only; we seek the video segment with the maximum motion activity, again by employing the NVS stream as an activity sensor [see (4.8)]. We cross-reference the timestamp of the maximum motion NVS frame to the RGB stream and extract a single shot of frames at this point for inference, which can be uploaded and processed on the cloud, as with the NVS recorded events. In order to minimize the latency in uploading the RGB frames compared to the NVS frames, we downsample captured RGB frames by a factor of 2, and only upsample to the original resolution prior to CNN processing. Subsequently, we fuse the NVS and RGB modalities by following Simonyan *et al*. [60] and simply average the scores per video instance, thus generating our prediction.

Table 4.4: Recognition accuracy on UCF-101 and HMDB-51 for stream modalities utilized during inference. The NVS stream is trained in the teacher-student framework with $\{\alpha, \beta, T\} = \{1, 1, 2\}$.

	Accuracy(%)	
	UCF-101	HMDB-51
NVS (emulated) CNN	78.6	51.6
RGB CNN	84.0	55.9
NVS (emulated)-RGB CNN	89.0	62.0

In order to isolate the performance of each stream utilized during inference, we first report the accuracy of each modality separately in Table 4.4 and compare this to the accuracy when the streams are fused for the UCF101 [58] and HMDB-51 [75] action recognition datasets. As is evident, the performance of the RGB CNN is boosted by the NVS CNN, despite complexity savings in opting for a short temporal extent for both streams and only inferring on a single shot at maximum motion activity.

4.4.1 Accuracy-Complexity Tradeoff

Our final results for the NVS-RGB CNN are reported along with recent external methods in Table 4.5. We report complexity for CNN processing in terms of total GFLOPs by multiplying the GFLOPs per input to the CNN with the number of inputs required during inference per video. Our proposed NVS-RGB CNN is able to achieve a competitive accuracy-complexity trade-off when compared to state-of-the-art methods utilizing highly complex optical flow or motion vectors. The only method substantially outperforming our approach in terms of accuracy is I3D [74]; however, I3D requires optical flow for both training and inference (and substantial APS activity for I3D (RGB-only)) and comprises a deep CNN with 3.7 to 7.7 times

Table 4.5: Accuracy and complexity versus the state-of-the-art (results reported where available) for UCF-101 and HMDB-51. Results are reported after averaging over the three splits per dataset. We also report the theoretical GFLOPs for CNN processing of all inputs per video during inference.

Method	Σ GFLOPs	UCF-101	HMDB-51
inc. optical flow			
Two-Stream [60]	150	88.0	59.4
3D Conv Fusion [61]	153	92.5	65.4
Action-VLAD [62]	-	92.7	66.9
ST-ResNet [153]	-	93.4	66.4
Two-Stream I3D [74]	648	97.8	80.9
no optical flow			
EMV-CNN [71]	150	86.4	-
CoViAR[72]	110	90.4	59.1
C3D [56]	385	82.3	51.6
Res3D [154]	193	85.8	54.9
I3D (RGB only)[74]	324	95.1	74.3
LTC (RGB only) [155]	308	82.4	-
Proposed, NVS (emulated)-RGB CNN	84	89.0	62.0

more GFLOPs than our NVS-RGB CNN. Additionally, the speed in generating conventional Brox optical flow [156] compared to emulated and real NVS frames is reported in Table 4.6 in terms of framerate; our improved PIX2NVS emulator can generate NVS frames at 357 FPS whilst the real NVS camera can output NVS frames at 2000 FPS (3 to 6 orders of magnitude faster than Brox flow). We therefore note that any RGB-derived model can easily be fused with our NVS stream for performance gain with lightweight computation in input generation and processing.

Table 4.6: Average framerate over 4600 video frames for Brox optical flow (CPU) and emulated (improved PIX2NVS) (CPU) and real (DAVIS240C) NVS frame generation.

	Framerate (FPS)
Improved PIX2NVS	357
DAVIS240C NVS camera [157]	2000
Brox optical flow [156]	0.314

Assuming we use a p2.xlarge AWS instance for CNN inference, which is equipped with a single K80 GPU that can process 8-9 TFLOPs/second, we can compute a theoretical approximate number of videos that can be processed per second (VPS) from the theoretical number of GFLOPs in Table 4.5. It is worth noting

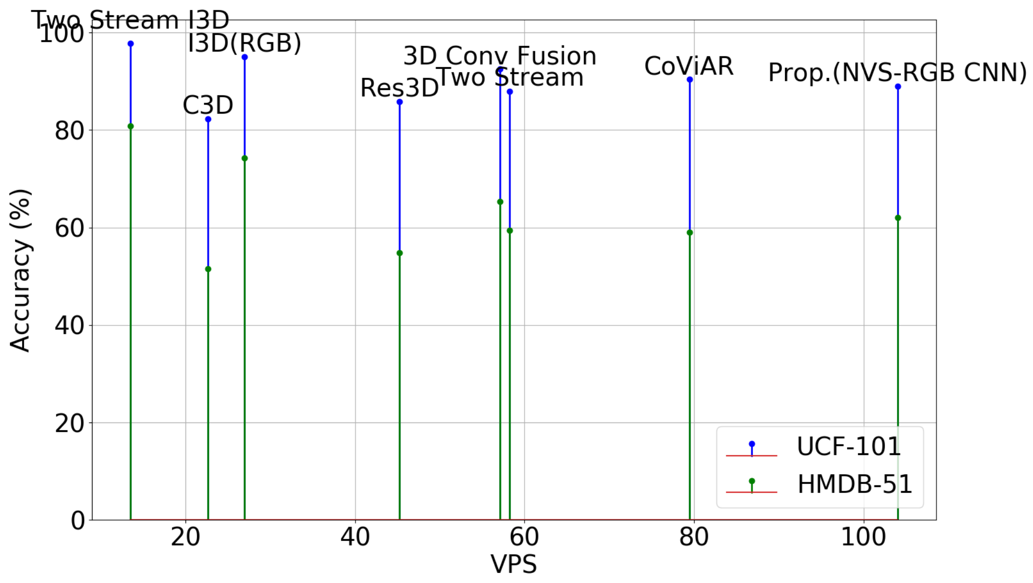


Figure 4.4: Accuracy-complexity plot for our proposed NVS(emulated)-RGB CNN versus the fastest competing methods with and without optical flow. Complexity is denoted in terms of the number of video processed per second (VPS).

that this is only an approximation, as we do not consider GPU memory, CPU and RAM bottlenecks. Nonetheless, it represents an upper bound to the true number of videos processed per second. Inline with our thesis objective, we plot an accuracy-complexity graph in Figure 4.4, where complexity is denoted by the upper bound on the VPS. From the graph, we note that Two-Stream I3D performs better than all other methods but with much lower VPS. Furthermore, Two-Stream I3D requires 250 optical flow frames, which each take $0.314^{-1} = 3.2$ seconds to generate; this presents a substantial bottleneck to the end-to-end VPS. Our NVS framework is able to circumvent this cost with emulated/real NVS frames and offers a better accuracy-complexity tradeoff compared to other frameworks.

4.5 Conclusion

We improve the sensing efficiency for multimodal action recognition by replacing conventional APS-based flow modalities with NVS frame representations for supplementing sparsely extracted RGB frame inputs. Namely, we propose the first method for NVS-based action recognition that performs competitively to state-of-the-art APS methods on standard datasets. We achieve this by first proposing improvements to current emulation frameworks that minimize the domain shift between real and emulated events. This provides us with an abundance of training data, as we are able to convert any APS dataset to the NVS domain. As such, we incorporate our proposed options into the PIX2NVS emulator and report a 12% improvement in accuracy for action recognition compared to the original settings. The improved NVS emulator is subsequently embedded into a large multi-modal teacher-student framework, in order leverage on the availability of labelled data for training with highly complex optical flow. Given its promising recognition performance and low complexity, our framework could become a viable solution within IoT and robotics contexts, where NVS data would be gathered at very low power and streamed to cloud computing servers for the back-end CNN processing.

Chapter 5

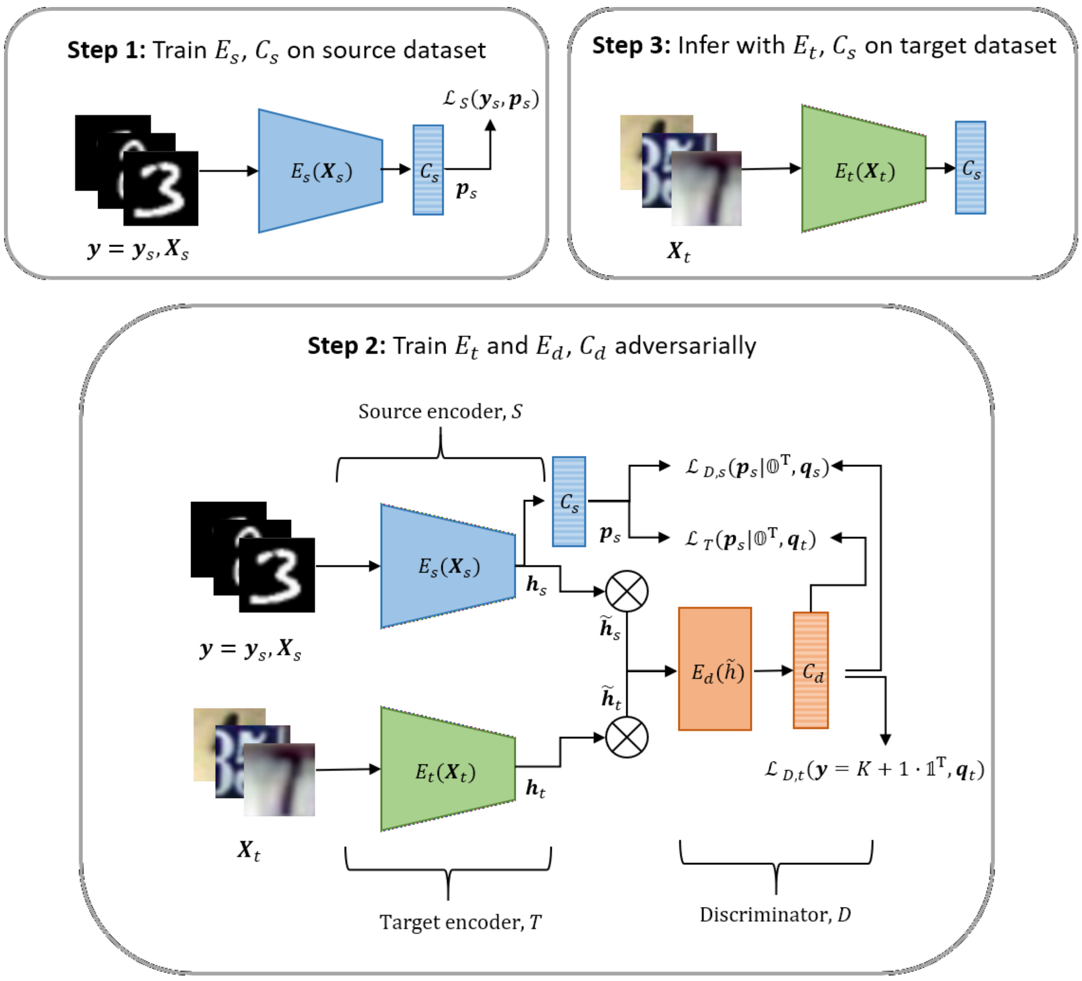
Improving Adversarial Discriminative Domain Adaptation

One of the limiting factors of the neuromorphic vision sensing (NVS) framework described in the previous chapter is the domain shift between the emulated and real NVS events, which we try to minimize by tuning the emulator parameters and quantify with the Earth Mover's distance in Section 4.2.4. Essentially, generating representative neuromorphic data can be considered a domain adaptation problem [89, 86], where the source domain is centered on the real NVS data, as produced by an NVS camera, and the target domain is the emulated NVS data. Unsupervised domain adaptation directly aims at improving the generalization capability between a labelled source domain and an unlabelled target domain. Deep domain adaptation methods can generally be categorized as either being discrepancy based or adversarial based, with the common end goal of minimizing the difference between the source and target distributions. Adversarial methods in particular have become increasingly popular due to their simplicity in training and success in minimizing the domain shift. One option would be to use a generative model, such as a generative adversarial network (GAN) [89, 90] for aligning the target domain to the (fixed) source domain. The advantage of a GAN over other generative methods is that there is no need for complex sampling or inference during training; the downside is that GANs are notoriously difficult to train, and this is made substantially worse due to the lack of available data in the source domain (i.e., real NVS data).

In this chapter, we focus on the recently proposed adversarial discriminative domain adaptation (ADDA) [91], which is related to generative adversarial learning and uses the GAN [89] objective to train on the target domain adversarially until it is aligned to the source domain. We assume the transductive setting, where the label sets are common between the source and target domains. Whilst ADDA only pretrains the source encoder with source dataset labels, in Section 5.1 we improve on the ADDA framework by first extending the discriminator output over the task classes, in order to additionally incorporate task knowledge into the adversarial loss functions. In adversarial training, we leverage on the fixed distribution over source encoder posteriors, in order to propose a maximum mean discrepancy (MMD) [78] and reconstruction-based loss function for training the target encoder and discriminator respectively. Subsequently, in Section 5.2, we provide an analysis of how our method extends over a simple multi-class extension of ADDA as well as other discriminative variants of semi-supervised GANs [158, 159]. In Section 5.3, we validate our proposal on conventional pixel domain datasets; namely, the digits and Office-31 datasets, on which we surpass the performance of ADDA by up to 13% and remain competitive to other recent proposals. Finally, we introduce and validate our proposal on a new emulated-to-real NVS dataset for sign language recognition, in which we substantially improve on accuracy compared to ADDA and training on the source only.

5.1 Improving Adversarial Adaptation

We illustrate the framework for improving unsupervised adversarial discriminative domain adaptation in Figure 5.1. Let $\boldsymbol{X}_S = \{(\boldsymbol{x}_s^i, y_s^i)\}_{i=0}^{N_s}$ represent the set of source image and label pairs, where $(\boldsymbol{x}_s, y_s) \sim \mathbb{D}_S$, $\boldsymbol{X}_T = \{(\boldsymbol{x}_t^i)\}_{i=0}^{N_t}$ represent the set of unlabeled target images, $\boldsymbol{x}_t \sim \mathbb{D}_T$. Let $E_s(\boldsymbol{x}_s;\theta_s)$ represent the source encoder function, parameterized by θ_s which maps an image $\boldsymbol{x}_s$ to the encoder output $\boldsymbol{h}_s$, where $(\boldsymbol{h}_s, y_s) \sim \mathbb{H}_S$. Likewise, let $E_t(\boldsymbol{x}_t;\theta_t)$ represent the target encoder function, parameterized by θ_t which maps an image $\boldsymbol{x}_t$ to the encoder output $\boldsymbol{h}_t$, where $\boldsymbol{h}_t \sim \mathbb{H}_T$. In addition, C_s represents a classifier function that maps the encoder output $\boldsymbol{h}$ to



Step 1: Train E_s, C_s on source dataset
$\mathcal{L}_S(\boldsymbol{y}_s, \boldsymbol{p}_s)$
$E_s(\boldsymbol{X}_s)$
C_s
$\boldsymbol{p}_s$
$\boldsymbol{y} = \boldsymbol{y}_s, \boldsymbol{X}_s$
Step 3: Infer with E_t, C_s on target dataset
$E_t(\boldsymbol{X}_t)$
C_s
$\boldsymbol{X}_t$
Step 2: Train E_t and E_d, C_d adversarially
Source encoder, S
C_s
$\boldsymbol{p}_s$
$\mathcal{L}_{D,s}(\boldsymbol{p}_s | \mathbb{0}^T, \boldsymbol{q}_s)$
$\mathcal{L}_T(\boldsymbol{p}_s | \mathbb{0}^T, \boldsymbol{q}_t)$
$E_s(\boldsymbol{X}_s)$
$\boldsymbol{h}_s$
$\tilde{\boldsymbol{h}}_s$
$\boldsymbol{y} = \boldsymbol{y}_s, \boldsymbol{X}_s$
$E_d(\tilde{h})$
C_d
$\tilde{\boldsymbol{h}}_t$
$E_t(\boldsymbol{X}_t)$
$\boldsymbol{h}_t$
$\mathcal{L}_{D,t}(\boldsymbol{y} = K + 1 \cdot \mathbb{1}^T, \boldsymbol{q}_t)$
$\boldsymbol{X}_t$
Target encoder, T
Discriminator, D

Figure 5.1: Proposed improvements for adversarial discriminative domain adaptation. The figure shows the best configuration for training and inference explored in the chapter.

class probabilities $\boldsymbol{p}$. In this chapter, we only consider $\boldsymbol{h}_s$ and $\boldsymbol{h}_t$ as representing the source and target logits respectively and therefore C_s simply denotes the softmax function on the logits. Finally, let $E_d(\boldsymbol{h}; \phi_d)$ represent an encoder mapping from $\boldsymbol{h}$ to an intermediate representation, and C_d represent a softmax function on said representation; E_d and C_d jointly constitute our discriminator mapping, which we refer to as $D = C_d(E_d)$.

Our objective is to substantially improve the adversarial training of the target encoder by maintaining the class separation during domain alignment. Rather than training the discriminator D and target encoder E_t with the standard GAN loss formulations (i.e., training a logistic function on the discriminator by assigning labels

1 and 0 to the source and target domains respectively and training the generator with inverted labels [89]), our approach is inspired by semi-supervised GANs [158, 159], where it has been found that incorporating task knowledge into the discriminator can jointly improve classification performance and quality of images produced by the generator. Under the discriminative adversarial framework, we can equivalently incorporate task knowledge by replacing the discriminator logistic function with a $K+1$ multi-class classifier. However, unlike the GAN setting, the discriminator inputs and outputs can now both be represented with $K+1$ dimensions, with each dimension representing a class; we can directly leverage on this fact in our proposed loss formulations in Sections 5.1.2.1 and 5.1.2.2 to improve the convergence properties of our framework.

We begin by outlining three main steps for our proposed adversarial framework, which involve learning the source mapping on the source dataset, adversarial training to align the source and target domains and finally inferring on the target dataset. The classifier C_s is fully interchangeable between the source encoder E_s and the target encoder E_t. This means we can embed C_s into the adversarial training of the target encoder E_t and discriminator D.

5.1.1 Step 1: Supervised Training of the Source Encoder and Classifier

Given that we have access to labels in the source domain, we first train the source encoder E_s and classifier C_s on the source image and label pairs $(\boldsymbol{x}_s, y_s \in \{1,...,K\})$ in a supervised fashion, by minimizing the standard cross entropy loss with K classes:

$$L_S = -\mathbb{E}_{(\boldsymbol{x}_s,y_s)\sim\mathbb{D}_S} \sum_{k=1}^{K} 1_{[k=y_s]} \log\left(C_s(E_s(\boldsymbol{x}_s))_k\right) \tag{5.1}$$

The source encoder parameters θ_s are now frozen, which fixes the distribution $\mathbb{H}_S$. This becomes our reference distribution for adversarial training, analogous to the real image distribution in the GAN setting, where our aim is now to align the target distribution $\mathbb{H}_T$ to $\mathbb{H}_S$ by learning a suitable target encoding E_t.

5.1.2 Step 2: Adversarial Training of the Encoder

5.1.2.1 Discriminator loss function L_D^{REC}

We train a target encoder adversarially by passing the source and target encoder logits, $\boldsymbol{h}_s$ and $\boldsymbol{h}_t$, to a discriminator D. In doing so, we implicitly align the target encoder distribution to that of the source; i.e., $E_t(\boldsymbol{x}_t) \sim \mathbb{H}_S$. As the source encoder has fixed parameters, we learn an asymmetric encoding with untied weights, which is the standard setting in both ADDA [91] and GAN implementations [89, 90]. In addition, we can improve the convergence properties by first initializing the target encoder weights with the source encoder weights; i.e., θ_t = θ_s.

We extend the discriminator output $\boldsymbol{q}$ to a $K+1$ dimensional vector representing the class probabilities, in which the first K dimensions represent the joint distribution over source domain and the task specific classes and final $K+1$-th dimension represents the target domain. We denote the $K+1$ class labels as $y \in \{1,\ldots,K+1\}$, where each source encoder logit $\boldsymbol{h}_s$ is assigned its task label $y = y_s \in \{1,\ldots,K\}$ and the 'target domain' label $y = K+1$ is only assigned to target encoder logits $\boldsymbol{h}_t$. However, contrary to semi-supervised GANs where the discriminator inputs are images, the discriminator inputs and outputs now share common supports over the K task classes. For the source domain, we can leverage on this fact by effectively modelling the discriminator as a denoising autoencoder [160], where we can jointly train the discriminator to reconstruct the source encoder logits and encourage the discriminator to potentially learn a more informative representation by corrupting the logits. A denoising autoencoder is an effective method of approximating the underlying source logit manifold and ensures that the discriminator deviates away from learning a simple identity function. We refer to the corruption process as $N(\tilde{\boldsymbol{h}}_s|\boldsymbol{h}_s)$, which represents the conditional distribution over the corrupted source encoder logits $\tilde{\boldsymbol{h}}_s$ given the source encoder logits $\boldsymbol{h}_s$. Therefore, the first term of our discriminator loss function is effectively a reconstruction loss, which we set as the cross entropy between the transformed source encoder posteriors $\hat{\boldsymbol{p}}_s = C_s(\boldsymbol{h}_s)||0$ and source discriminator posteriors $\boldsymbol{q}_s$ (i.e., post-softmax), where $||$ denotes a concatentation operation:

$$\begin{aligned} L_{D,s}^{\text{REC}} &= -\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\mathbb{E}_{\tilde{\boldsymbol{h}}_s\sim N(\tilde{\boldsymbol{h}}_s|\boldsymbol{h}_s)}(C_s(\boldsymbol{h}_s)||0\cdot\ \log(D(\tilde{\boldsymbol{h}}_s))) \\ &= -\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\mathbb{E}_{\tilde{\boldsymbol{h}}_s\sim N(\tilde{\boldsymbol{h}}_s|\boldsymbol{h}_s)}\sum_{k=1}^{K}\hat{p}_{s,k}\log(q_{s,k}) \end{aligned} \tag{5.2}$$

Notably, we append a zero to the source encoder posteriors to represent the $K+1$-th ‘target domain’ class, which maintains a valid probability distribution (sums to 1), whilst enforcing a zero probability that the posteriors were generated by the target encoder. In this chapter, the corruption process N is simply configured as dropout on the encoder logits.

We also apply dropout independently to the target encoder logits $\boldsymbol{h}_t$, in order to symmetrize the source and target encoder inputs presented to the discriminator. However, we want the discriminator to distinguish between the source and target encoder logits. We train the discriminator to assign the $K+1$-th ‘target domain’ class to the corrupted target encoder logits $\tilde{\boldsymbol{h}}_t$, such that they lie in an orthogonal space to the source domain. In other words, the second term of our discriminator loss function for the target encoder logits is:

$$L_{D,t}^{\text{REC}} = -\mathbb{E}_{\boldsymbol{h}_t\sim\mathbb{H}_T}\mathbb{E}_{\tilde{\boldsymbol{h}}_t\sim N(\tilde{\boldsymbol{h}}_t|\boldsymbol{h}_t)}\log(D(\tilde{\boldsymbol{h}}_t)_{K+1}) \tag{5.3}$$

where $D(\tilde{\boldsymbol{h}}_t)_{K+1}$ is the $K+1$-th dimension of $D(\tilde{\boldsymbol{h}}_t)$. The discriminator loss function L_D^{REC} is thus simply the sum of (5.2) and (5.3): $L_D^{\text{REC}} = L_{D,s}^{\text{REC}} + L_{D,t}^{\text{REC}}$. In order to further motivate this reconstruction based loss function, we derive a loss function akin to a discriminative variant to semi-supervised GANs in Section 5.2.1 and compare with our proposed formulation.

5.1.2.2 Target encoder loss function L_T^{MMD}

In order to train the target encoder adversarially, we want the target encoder to generate an output that is representative of one of the first K task-specific classes rather than the $K+1$-th ‘target domain’ class that it is assigned when training the discriminator. To achieve this, we leverage on the two source posteriors, $\boldsymbol{p}_s = C_s(\boldsymbol{h}_s)$

and $\boldsymbol{q}_s = D(\tilde{\boldsymbol{h}}_s)$, generated by the source encoder and discriminator respectively. Contrary to supervised domain adaptation methods, there are no known source and target pairwise correspondences and we cannot formulate a paired test over the posteriors. However, we can formulate the problem as a two-sample test by considering the distribution over target discriminator posteriors, $\boldsymbol{q}_t = D(\tilde{\boldsymbol{h}}_t)$, compared to the distribution over the source encoder posteriors $\boldsymbol{p}_s$, where our null hypothesis is that the distributions are equal. We consider a set of target discriminator posteriors $\mathbf{Q}_T = \{\boldsymbol{q}_t^1, \ldots, \boldsymbol{q}_t^n\} \sim \mathbb{Q}_T$ and a set of source encoder posteriors $\mathbf{P}_S = \{\boldsymbol{p}_s^1, \ldots, \boldsymbol{p}_s^n\} \sim \mathbb{P}_S$, where n is the set cardinality and $\mathbb{P}_S$ and $\mathbb{Q}_T$ are the respective posterior distributions. Effectively, we want to minimize the distance between $\mathbb{P}_S$ and $\mathbb{Q}_T$ without performing any density estimation. To this end, we adopt the Maximum Mean Discrepancy (MMD) [78] metric as a measure of distance between the mean embeddings of $\boldsymbol{p}_s$ and $\boldsymbol{q}_t$. For reproducing kernel Hilbert space (RKHS) $\mathscr{H}$, function class $\mathscr{F} = \{f : \|f\| \leq 1\}$ and infinite dimensional feature map $\phi : \mathscr{X} \rightarrow \mathscr{H}$ the MMD can be expressed as:

$$\begin{aligned} \mathscr{D}_{\text{MMD}} &= \sup_{f \in \mathscr{F}, \|f\|_{\mathscr{H} \leq 1}} \left| \mathbb{E}_{\boldsymbol{p}_s \sim \mathbb{P}_S} f(\boldsymbol{p}_s || 0) - \mathbb{E}_{\boldsymbol{q}_t \sim \mathbb{Q}_T} f(\boldsymbol{q}_t) \right| \\ &= \left\| \mathbb{E}_{\boldsymbol{p}_s \sim \mathbb{P}_S} \phi(\boldsymbol{p}_s || 0) - \mathbb{E}_{\boldsymbol{q}_t \sim \mathbb{Q}_T} \phi(\boldsymbol{q}_t) \right\|_{\mathscr{H}} \end{aligned} \tag{5.4}$$

The distribution $\mathbb{P}_S$ over source encoder posteriors is fixed during adversarial training and, as such, we are effectively aligning the distribution $\mathbb{Q}_T$ over target discriminator posteriors to $\mathbb{P}_S$. We again append a 0 to the source encoder posteriors to represent the ‘target domain’ class probability, such that both source and target posteriors are $K+1$ dimensional prior to mapping to $\mathscr{H}$. This zero constraint on the $K+1$-th class acts as a stronger prior upon which to learn the target encoder; as such, the source encoder posterior provides a more informative representation than the source discriminator posterior. The feature map ϕ in (5.4) corresponds to a positive semi-definite kernel k such that $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathscr{H}}$, which means we can rewrite (5.4) in terms of k. The loss function on our target encoder that we wish to minimize can thus be written for aligning $\mathbb{Q}_T$ to $\mathbb{P}_S$ as:

$$\begin{aligned} L_T^{\text{MMD}}(\mathbb{P}_S \to \mathbb{Q}_T) = {D_{\text{MMD}}}^2 = &\mathbb{E}_{\boldsymbol{p}_s,\boldsymbol{p}_s' \sim \mathbb{P}_S,\mathbb{P}_S} k(\boldsymbol{p}_s||0, \boldsymbol{p}_s'||0) \\ &- \mathbb{E}_{\boldsymbol{p}_s,\boldsymbol{q}_t \sim \mathbb{P}_S,\mathbb{Q}_T} k(\boldsymbol{p}_s||0, \boldsymbol{q}_t) \\ &+ \mathbb{E}_{\boldsymbol{q}_t,\boldsymbol{q}_t' \sim \mathbb{Q}_T,\mathbb{Q}_T} k(\boldsymbol{q}_t, \boldsymbol{q}_t') \end{aligned} \tag{5.5}$$

In this chapter we opt to use a linear combination of r multiple radial basis function (RBF) kernels over a range of standard deviations, such that $k(\boldsymbol{x},\boldsymbol{y}) = \sum_r \exp\{-\frac{1}{2\sigma_r}\|\boldsymbol{x}-\boldsymbol{y}\|^2\}$, where σ_r is the standard deviation of the r-th RBF kernel. We find that the standard RBF kernel as above performs better in practice than a generalized RBF kernel with a distribution based metric such as chi-squared distance or squared Hellinger's distance, although these are viable options. By introducing a linear combination over varying bandwidths, we improve the generalization performance over different sample distributions. This method of generalization with fixed kernels is commonly used both in generative models [161, 162] and other domain adaptation discrepancy based methods [80, 95]. In order to further motivate our proposed MMD loss formulation, we introduce alternative target encoder loss functions in Section 5.2 and a full ablation study on all introduced discriminator-encoder loss combinations in Section 5.3.1.2.

5.1.3 Step 3: Inference on the Target Dataset

After training the target encoder, we can now perform inference on the target dataset. However, we have effectively trained two sets of target predictions; namely, the mapped target encoder output $C_s(\boldsymbol{h}_t)$ and the discriminator output $\boldsymbol{q}_t$. In the optimal setting, where we have trained the discriminator to equilibrium, we would expect the discriminator mapped source and target distributions would be aligned. However, we empirically find that evaluation on $\boldsymbol{q}_t$ is marginally worse (˜1%) than evaluation on $C_s(\boldsymbol{h}_t)$. Therefore, for the remainder of the chapter, we infer on the target encoder output. The class prediction y_{pred} is given as:

$$y_{\text{pred}} = \arg \max_{j \in \{1,\ldots,K\}} (h_{t,j}) \tag{5.6}$$

5.2 Bridging the Gap from ADDA to Our Proposal

We provide further insight on the design of our adversarial loss formulations by first demonstrating in Section 5.2.1 how we can extend from ADDA [91], to a multi-class version of ADDA with separate task and domain classification heads and, finally, to a framework with a single classification head. For the latter, we perform a detailed comparison between our proposal and discriminative variants of semi-supervised GANs [158, 159] in Section 5.2.2.

5.2.1 Transitioning from Two Heads to One Head

Let us denote a discriminator classification head as the layer H and the preceding discriminator layers as D'. We begin with ADDA, which is typically trained with a domain classification head H_{domain}, in which the discriminator assigns a domain label $y_b \in \{0,1\}$ to instances (where 1 corresponds to the source domain instance and 0 to the target domain instance). The discriminator loss function can be written as [91]:

$$\begin{aligned} L^{\text{ADDA}}_{D',H_{\text{domain}}} = &-\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S} \log\left(p_{\text{domain}}(y_b = 1|\boldsymbol{h}_s)\right) \\ &-\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T} \log(1 - p_{\text{domain}}(y_b = 1|\boldsymbol{h}_t)) \end{aligned} \tag{5.7}$$

where $p_{\text{domain}}(y_b = 1|\boldsymbol{h})$ is the posterior probability output by $H_{\text{domain}}(D'(\boldsymbol{h}))$ that logit $\boldsymbol{h}$ is from the source domain. Similarly, the target encoder can be trained in an adversarial setting with a minimax loss function $L^{\text{MAX}}_T = \mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T} \log(1 - p_{\text{domain}}(y_b = 1|\boldsymbol{h}_t)$ or an inverted label loss function $L^{\text{INV}}_T = -\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T} \log(p_{\text{domain}}(y_b = 1|\boldsymbol{h}_t))$.

The simplest extension of ADDA to a multi-class variant that leverages on source task knowledge would be add another K-dimensional head H_{task} to the discriminator. This additional head performs task classification, and trains the discriminator to classify the source examples only based on their task labels $y_s \in \{1,\ldots,K\}$. This setup is analogous to the DANN [85] except we have separate domain encoders and we replace the gradient reversal layer with a discriminator and adversarial training. In order to simplify the expressions, we can write the discriminator loss function for two heads in terms of posteriors as:

$$L^{\text{MULTI}}_{D',H_{\{\text{domain,task}\}}} = -\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\log\left(p_{\text{task}}(y_s|\boldsymbol{h}_s)\right) + L^{\text{ADDA}}_{D',H_{\text{domain}}} \tag{5.8}$$

where $p_{\text{task}}(y_s = k|\boldsymbol{h}_s)$ is the posterior probability output by $H_{\text{task}}(D'(\boldsymbol{h}_s))_k$ that source logit $\boldsymbol{h}_s$ is from class with label $y_s = k$. The first term represents the cross entropy loss with source task labels and the remaining terms equate to $L^{\text{ADDA}}_{D',H_{\text{domain}}}$. As we only train the domain head adversarially, the adversarial loss function for training the target encoder is simply L^{INV}_T or L^{MAX}_T.[1]

We can rewrite (5.8) as:

$$\begin{aligned} L^{\text{MULTI}}_{D',H_{\{\text{domain,task}\}}} = &-\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\log\left(p_{\text{task}}(y_s|\boldsymbol{h}_s).p_{\text{domain}}(y_b = 1|\boldsymbol{h}_s)\right) \\ &-\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(1 - p_{\text{domain}}(y_b = 1|\boldsymbol{h}_t)) \end{aligned} \tag{5.9}$$

As is evident from the first term in (5.9), with two heads we are effectively optimizing the likelihood of the joint posterior distribution over the task classes and source domain, but treating source domain classification and task classification as independent events. Notably, as we only have access to labels in the source domain, the task classifier is only trained on source domain examples. As such, we can improve generalization by removing the independence assumption and model with a single multi-task classification head H_{joint}:

$$\begin{aligned} L^{\text{JOINT}}_{D',H_{\text{joint}}} = &-\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\log\left(p_{\text{joint}}(y_s,y_b = 1|\boldsymbol{h}_s)\right) \\ &-\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(1 - p_{\text{joint}}(y_b = 1|\boldsymbol{h}_t)) \\ = &-\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\log\left(p_{\text{joint}}(y_s|\boldsymbol{h}_s,y_b = 1).p_{\text{joint}}(y_b = 1|\boldsymbol{h}_s)\right) \\ &-\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(1 - p_{\text{joint}}(y_b = 1|\boldsymbol{h}_t)) \end{aligned} \tag{5.10}$$

By directly optimizing the joint posterior distribution $p_{\text{joint}}(y_s,y_b = 1|\boldsymbol{h}_s)$, we

[1]Another option would be to train H_{task} adversarially in addition to H_{domain} by alternately minimizing and maximizing a distribution metric between $H_{\text{task}}(D'(\boldsymbol{h}_s))$ and $H_{\text{task}}(D'(\boldsymbol{h}_t))$. However, as there is no 'target domain' class, this intuitively means that the only way for the discriminator to maximize the metric would be to introduce intra-class confusion within the target domain - thus leading to instability during training.

can now also implicitly model a conditional dependency $p_{\text{joint}}(y_s|\boldsymbol{h}_s, y_b = 1)$ for task classification given the source domain. Furthermore, if we marginalize over the task labels y_s, we end up with the standard ADDA loss formulation $L^{\text{ADDA}}_{D',H_{\text{domain}}}$, of (5.7).

As in our proposal, we can write (5.10) in terms of a single $K+1$ classification head with $K+1$ labels $y \in \{1,\ldots,K+1\}$, where the first K classes model the joint distribution over task classes and source domain and the $K+1$-th class models the distribution over the target domain:

$$\begin{aligned} L^{\text{JOINT}}_{D',H_{\text{joint}}} &= -\mathbb{E}_{(\boldsymbol{h}_s,y)\sim\mathbb{H}_S}\log\left(p_{\text{joint}}(y, y<K+1|\boldsymbol{h}_s)\right) \\ &\quad -\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(p_{\text{joint}}(y=K+1|\boldsymbol{h}_t)) \\ &= -\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\log\left(p_{\text{joint}}(y_s|\boldsymbol{h}_s)\right) \\ &\quad -\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(p_{\text{joint}}(y=K+1|\boldsymbol{h}_t)) \end{aligned} \tag{5.11}$$

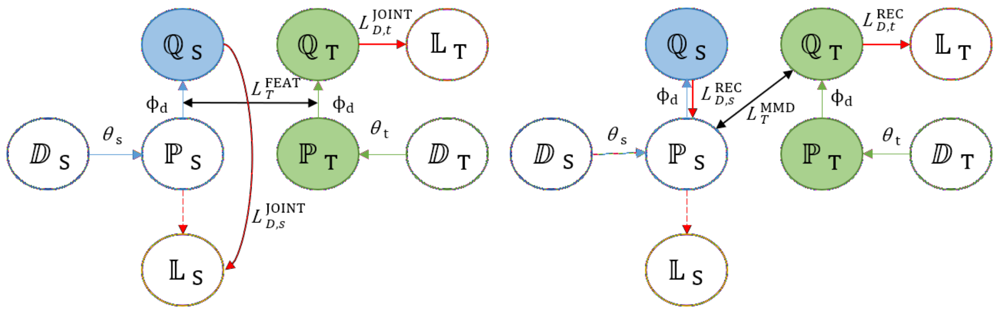
With the above notation, $p_{\text{joint}}(y=K+1|\boldsymbol{h}) = (1-p_{\text{joint}}(y_b=1|\boldsymbol{h}))$ and $p_{\text{joint}}(y, y<K+1|\boldsymbol{h}_s) = p_{\text{joint}}(y_s, y_b=1|\boldsymbol{h}_s))$. Finally, we can rewrite (5.11) in terms of the discriminator $D(\boldsymbol{h})_k = p_{\text{joint}}(y=k|\boldsymbol{h})$ and $D = H_{\text{joint}}(D')$, which gives us the loss function for a discriminative variant of semi-supervised GANs [158, 159]:

$$\begin{aligned} L^{\text{JOINT}}_{D} &= -\mathbb{E}_{(\boldsymbol{h}_s,y_s)\sim\mathbb{H}_S}\sum_{k=1}^{K}1_{[k=y_s]}\log\left(D(\boldsymbol{h}_s)_k\right) \\ &\quad -\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}\log(D(\boldsymbol{h}_t)_{K+1}) \end{aligned} \tag{5.12}$$

We denote the first expectation term in (5.12) as $L^{\text{JOINT}}_{D,s}$ and the second expectation term as $L^{\text{JOINT}}_{D,t}$.

5.2.2 From Discriminative Variants of Semi-Supervised GANs to Our Proposal

Given the above derivation, we now compare our proposed loss formulations (as detailed in Sections 5.1.2.1 and 5.1.2.2) with the conventional loss functions adopted in semi-supervised generative adversarial networks, repurposed for the discriminative setting. The discriminator is trained using the loss function L^{JOINT}_{D} in (5.12) for discriminative variants of semi-supervised GANs. Our proposed discriminator loss



(a) JOINT – FEAT (feature matching) **(b)** REC – MMD (proposed)

Figure 5.2: (Best viewed in color) Computational graphs showing the discriminator-encoder loss formulations for a) feature matching as in semi-supervised GANs and b) our improved framework for discriminative adversarial training. Blue nodes/arrows represent the source domain components and green nodes/arrows represent the target domain components. Uncolored and colored nodes represent fixed and trainable elements during adversarial training. Red and black arrows represent the discriminator and encoder loss components respectively and their arrow direction represents the direction of alignment for asymmetric losses.

function L_D^{REC} follows the same format, except we substitute logits $\boldsymbol{h}$ for noisy logits $\tilde{\boldsymbol{h}} \sim N(\tilde{\boldsymbol{h}}|\boldsymbol{h})$ and substitute the indicator function $1_{[k=y_s]}$ with the source encoder posteriors $\boldsymbol{p}_s = C_s(\boldsymbol{h}_s)$, thus emulating a denoising autoencoder in the first term.

Semi-supervised GANs are typically trained adversarially with either a minimax or feature matching objective function. For a discriminative variant with the minimax objective, the target encoder is trained by simply maximizing (5.12). For feature matching, the target encoder is trained to minimize a L2 distance-based loss on the averaged intermediate source and target activations $f(\boldsymbol{h})$ of the discriminator:

$$L_T^{\mathrm{FEAT}} = \left\| \mathbb{E}_{(\boldsymbol{h}_s, y_s)\sim\mathbb{H}_S}(f(\boldsymbol{h}_s)) - \mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{H}_T}(f(\boldsymbol{h}_t)) \right\|_2^2 \tag{5.13}$$

In particular, Figure 5.2 presents the computational graphs showing the discriminator-encoder loss formulations for (a) feature matching as in semi-supervised GANs with L_D^{JOINT} and L_T^{FEAT} , and (b) our proposal with L_D^{REC} and L_T^{MMD} for training the discriminator and target encoder respectively. Each node is represented with its modelled distribution; for example, for the source encoder S, $\mathbb{L}_S$ represents the source label distribution, $\mathbb{D}_S$ represents the source data distribution and $\mathbb{P}_S$ and $\mathbb{Q}_S$ represent the distribution over source encoder and discriminator posteriors respectively. We define $\mathbb{L}_T$ as a unit impulse at $y = K + 1$. We note from Figure 5.2(a) that for L_T^{FEAT}, both $f(\boldsymbol{h}_s)$ and $f(\boldsymbol{h}_t)$ are not fixed but changing with time, as the discriminator parameters ϕ_d are trained during adversarial alignment. Thus, the target encoder is more prone to experiencing internal covariate shift, due to the changing source reference.[2] In the case of domain adaptation with feature matching, the constantly changing $f(\boldsymbol{h}_s)$ adds noise to the target encoder alignment and destabilizes training. On the other hand, in our proposal we recognize that the distribution of the source encoder posteriors $\boldsymbol{p}_s \sim \mathbb{P}_S$ is fixed and only changes stochastically with mini-batch; as such, we show in Figure 5.2(b) that we centralize our discriminator and encoder loss functions around this distribution. That is, for the proposed encoder loss L_T^{MMD}, $\mathbb{Q}_T$ is aligned to the fixed $\mathbb{P}_S$. This, along with the imposed zero constraint on the source encoder posteriors for the $K+1$-th class probability, is key for stabilizing training of the target encoder compared to both minimax and feature matching objectives. It is also worth noting that MMD employed in our proposed target encoder loss function can also be interpreted as matching all moments between the source and target posterior distributions, whereas conventional feature matching of (5.13) is only empirically matching the first order moments (means) of the intermediate discriminator layer activations.

For the sake of completeness, we propose a final discriminative variant inspired by unsupervised GAN training, where the generator is typically trained with an inverted label objective (i.e., inverting the generator label and training with cross entropy), instead of maximizing (5.12), in order to stabilize training. As the inverted label objective is not viable for a multi-class discriminator output in our proposal, we instead propose a pseudo-label objective for training the target encoder in the discriminative setting. This objective draws parallels to unsupervised domain adaptation work that use pseudo-labels (typically in conjunction with co-training). The pseudo-label is taken as the index of the maximum of the first K discriminator logits $\boldsymbol{h}_d$. In other words, denoting $\hat{y}_t = \text{argmax}_{j\in 1,\ldots,K}\, \boldsymbol{h}_d$, we train the target encoder by

[2]Internal covariate shift is the phenomenon wherein the distribution of inputs to a layer in the network changes due to an update of parameters of the previous layers, and is typically synonymous with batch normalization [111], which tries to minimize internal covariate shift by normalizing each layer to be zero mean and unit variance.

minimizing:

$$L_T^{\text{PSEUDO}} = -\mathbb{E}_{(\boldsymbol{h}_t)\sim\mathbb{P}_T} \sum_{k=1}^{K} 1_{[k=\hat{y}_t]} \log(D(\boldsymbol{h}_t)_k) \tag{5.14}$$

We note that, unlike our proposed target encoder loss function that is distribution-based, both inverted label assignment and our pseudo-label assignment are instance-based. This potentially means they are more prone to instability from noisy examples in the training batch.

In order to motivate our proposed adversarial loss functions compared to these discriminative variants of semi-supervised GANs, we perform an extensive ablation analysis in Section 5.3.1.2 on the SVHN $\rightarrow$ MNIST domain adaptation task.

5.3 Experimental Results and Analysis

We present experimental results and analysis on the unsupervised domain adaptation task. In order to compare with ADDA and other recently proposed methods, we experiment on four digits datasets of varying sizes and difficulty: MNIST-M [85], MNIST [163], USPS and SVHN [164]. We demonstrate substantial gain over ADDA and other recent methods, which is evident on the more difficult domain adaptation tasks such as SVHN $\rightarrow$ MNIST. We additionally report accuracy on the Office-31 dataset [165] compared to the current state-of-the-art methods. Finally, since neuromorphic vision sensing presents a pertinent application for domain adaptation, we additionally introduce and validate on a new NVS sign language dataset, demonstrating substantial gain in target accuracy compared to training with the source domain only. For each domain adaptation task, we extract 5% of each target adversarial training split for validation, in order to tune the hyperparameters. To ensure consistency and demonstrate lack of sensitivity to hyperparameters, we fix these globally over all tasks. Specifically, for the MMD kernel combination k, we found that on average, the optimal performance for our framework is achieved with a summation over five kernels, with $\sigma_r = 10^{-r}, r \in \{0, \ldots, 4\}$. Finally, as the discriminator is typically overcomplete (more nodes in the hidden layers than input classes), we add an L1 regularization term to (5.2) on the discriminator weights $\boldsymbol{w}_d$

to improve the feature selection with regularization coefficient $\lambda = 0.001$ for all cases.

5.3.1 Digits datasets

We consider four standard domain adaptation scenarios between dataset pairs drawn from MNIST-M [85], MNIST [163], USPS and SVHN [164] digits datasets, which are each comprised of $K = 10$ digit classes (0-9). Specifically, we evaluate on MNIST $\rightarrow$ USPS, USPS $\rightarrow$ MNIST, SVHN $\rightarrow$ MNIST and MNIST $\rightarrow$ MNIST-M. The difficulty in domain adaptation task increases as the variability between datasets increases. We follow a similar training procedure of [91]. For the MNIST $\rightarrow$ USPS and USPS $\rightarrow$ MNIST experiments, we sample 2000 images from MNIST and 1800 from USPS, otherwise we train and infer on the full datasets. For MNIST $\rightarrow$ MNIST-M, we generate the unlabelled MNIST-M target dataset by following the process described by [85]. For all experiments we use a modified LeNet architecture [163] for the source and target encoder. The discriminator is comprised of 2 fully connected layers with 500 hidden units and a final fully-connected layer with $K+1 = 11$ hidden units that outputs the logits. With this setup, our network is roughly the same complexity as ADDA in terms of number of parameters. In step 1, the source encoder is trained with the Adam [166] optimizer for 10k iterations with a batch size of 128 and learning rate of 0.001. In step 2, the target encoder is trained with a batch size of 128 per domain for 5k iterations on the smaller datasets MNIST $\rightarrow$ USPS, USPS $\rightarrow$ MNIST and 10k otherwise, with a lower learning rate of 0.0002, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We resize all images to a fixed size of 28×28 prior to CNN processing. Additonally, we use data augmentation for MNIST $\rightarrow$ MNIST-M by randomly inverting the MNIST grayscale values and replicating the MNIST inputs channel-wise to match MNIST-M dimensions. Our results are provided in Table 5.2 compared to the current state-of-the-art and when training on source only. We focus our comparison on ADDA [91] and DIFA [92], which are recently proposed adversarial methods.

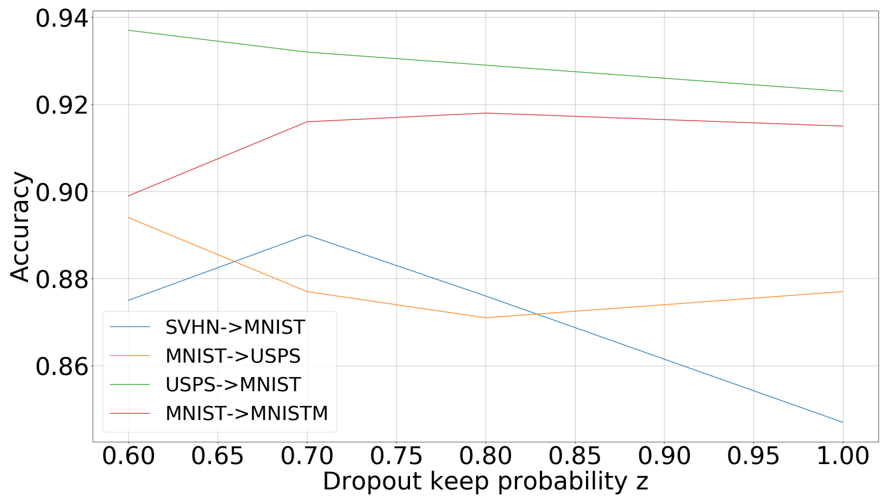


Figure 5.3: Graph of dropout keep probability z versus accuracy on subset of target dataset used for validation for various digits domain adaptation tasks

5.3.1.1 Parametric exploration for discriminator loss function L_D^{REC}

In Figure 5.3 we perform a parametric exploration over different values of dropout keep probability z by computing the accuracy on the validation set for various digits domain adaptation tasks. It is worth noting that a keep probability $z > 0.5$ generally maintains overlapping class supports between the encoder and discriminator posteriors and the accuracy can drop substantially for $z < 0.5$; therefore we only plot z for $[0.6, 1.0]$. No denoising corresponds to $z = 1.0$. We note from the figure that including denoising either improves or maintains accuracy; in particular for the most difficult task SVHN→MNIST, the accuracy improves by 5% when decreasing z from 1.0 to 0.7. As $z = 0.7$ provides the most consistent gain for the four tasks, we fix z to this value for the remainder of the chapter.

5.3.1.2 Ablation Study

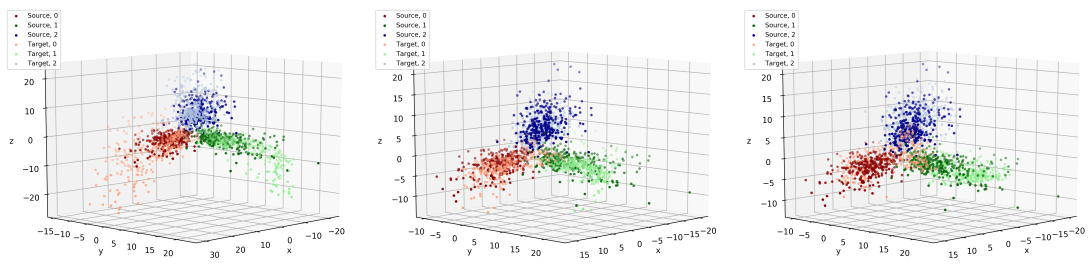
In order to illustrate the performance of our method and better understand where the performance gains are coming from, we perform an ablation analysis over various discriminator-encoder loss combinations for the most difficult digit domain adaptation task, SVHN→MNIST. This includes the loss formulations introduced in

Table 5.1: Accuracy on SVHN → MNIST task with 3 (0,1 and 2) and 10 classes for all considered discriminator ($L_{D',H}$)-encoder (L_T) loss combinations (as detailed in Section 5.2). Each loss function is denoted by its corresponding column and superscript (e.g., ADDA $\rightarrow L^{\text{ADDA.}}_{D',H_{\text{domain}}}$). $\mathbb{P}_S \rightarrow \mathbb{Q}_T$ refers to distribution alignment of $\mathbb{Q}_T$ to $\mathbb{P}_S$.

$L_{D',H}$	L_T	3 classes	10 classes
ADDA	INV (Inverted label)	0.831	0.787
	MAX (Minimax)	0.866	0.799
MULTI	INV (Inverted label)	0.847	0.783
	MAX (Minimax)	0.868	0.796
JOINT	MAX (Minimax) [159, 158]	0.856	0.753
	FEAT (Feature matching) [159]	0.805	0.772
	PSEUDO (Pseudo-label)	0.863	0.800
	MMD ($\mathbb{P}_S \rightarrow \mathbb{Q}_T$)	0.895	0.856
REC (proposed)	MMD ($\mathbb{Q}_S \rightarrow \mathbb{Q}_T$)	0.878	0.804
	MMD ($\mathbb{P}_S \rightarrow \mathbb{Q}_T$) (proposed)	0.948	0.890

Section 5.2. Our results are presented in Table 5.1, for all considered loss combinations, when training on 3 classes only (0,1 and 2) and all 10 classes. All hyperparameters and the training procedure are as described above for digits datasets; the only distinction is that for 3 classes we train on the source for only 5000 iterations to minimize overfitting.

One head versus two heads: In the first two parts of Table 5.1, we evaluate performance of training with ADDA versus a multi-class variant of ADDA (MULTI) with two classification heads (as introduced in Section 5.2.1). For these two benchmarks, we train the target encoder with the inverted label setting or minimax, i.e., L_T^{INV} and L_T^{MAX} respectively. The results show that adding the additional task classification head H_{task} has little effect on accuracy compared to baseline ADDA. We attribute this to the fact that the inputs $\boldsymbol{h}_s$ and outputs of $H_{\text{task}}(D'(\boldsymbol{h}_s))$ are both learned with the source labels, and the output is not conditioned on the domain. The task classification head H_{task} simply learns to invert the mapping learned by the preceding discriminator layers D'; $H_{\text{task}} \sim D'^{-1}$. This further motivates our proposed learning with a single classification head that models the joint distribution between the task and source domain classification, as defined in (5.12). However, the third part of Table 5.1 shows that there appears to be a stronger discriminator bias and a slight



(a) Proposed **(b)** Minimax **(c)** Feature matching

Figure 5.4: (Best viewed in color) 3D scatter plot of source and target logits for SVHN $\rightarrow$ MNIST domain adaptation task on 3 classes only (0, 1 and 2) for our proposed, minimax and feature matching loss formulations. Source and target examples are randomly selected from the SVHN and MNIST test datasets respectively for visualization.

detriment in accuracy for JOINT $-$ MAX compared to the baseline ADDA $-$ MAX discriminator-encoder loss combinations. This motivates the need for our proposed encoder loss function L_T^{MMD}.

Proposed $L_D^{\text{REC}}, L_T^{\text{MMD}}$ versus discriminative variants of semi-supervised GANs: We next consider how our proposed MMD based target encoder loss function $L_T^{\text{MMD}}(\mathbb{P}_S \rightarrow \mathbb{Q}_T)$, improves over conventional minimax L_T^{MAX}, feature matching L_T^{FEAT} and pseudo-label L_T^{PSEUDO} encoder loss formulations, as defined in Section 5.2. In order to isolate the performance of our proposed MMD loss function and perform a fair comparison, we fix the discriminator loss function to L_D^{JOINT}. The third part of Table 5.1 shows that, on both 3 and 10 classes our proposed MMD loss formulation outperforms all other target encoder loss variants. In particular, when compared to feature matching, our proposal provides accuracy gains of up to 9% on both 3 and 10 classes. In order to establish the source of this gain, we present 3D scatter plots in Figure 5.4 of the source and target logits when trained on 3 classes only from the SVHN $\rightarrow$ MNIST domain adaptation task.[3] As shown in the figure, both minimax and feature matching are prone to overfitting on the source dataset. While both formulations result in a tight bound on the source distribution, they forego a good class separation close to the source origin, where the distribu-

[3]We opted for this approach instead of using a reduction method such as t-SNE [167] that introduces additional hyperparameters such as perplexity to visualize the domain shift.

tion over classes is more uniform and the target encoder loss would be smaller in magnitude, potentially unfavourably biasing towards the discriminator. This misclassification around the origin is most noticeable for the '0' digit class and is worst for feature matching which validates its poor accuracy results on 3 classes in Table 5.1.

Having validated the performance gain from our proposed target encoder loss function, we now switch the discriminator loss function from L_D^{JOINT} to our proposed reconstruction loss L_D^{REC} , with $z = 0.7$. Combining our two proposed loss formulations for adversarial training, L_D^{REC} and $L_T^{\text{MMD}}(\mathbb{P}_S \to \mathbb{Q}_T)$, we are able to achieve the best performance on 3 and 10 classes. This is due to a combination of increased separability between the source and target domains in RKHS, and the fixed source distribution and hard zero constraint on the 'target' class that minimizes the internal covariate shift when training to align the distribution over target discriminator posteriors $\mathbb{Q}_T$. In order to isolate the detriment from internal covariate shift, we also consider aligning the distributions over source and target discriminator posteriors $(\mathbb{Q}_S \to \mathbb{Q}_T)$ in L_T^{MMD}, with $\boldsymbol{q}_s \sim \mathbb{Q}_S$ and $\boldsymbol{q}_t \sim \mathbb{Q}_T$. The second last row of the table shows the performance if we align $\mathbb{Q}_T$ to $\mathbb{Q}_S$ in our encoder loss function; our accuracy drops substantially to 87.8% and 80.4% for 3 and 10 classes respectively, which illustrates the effect of the internal covariate shift, as in feature matching.[4]

5.3.1.3 Adaptation on digits datasets

Finally, we report accuracy for our proposed method. In order to isolate the performance gain from domain adaptation, we compute the percentage increase (relative) over the source only accuracy reported in the chapter (shown in parentheses in Table 5.2). On average over all datasets, our proposals outperform DIFA and ADDA. Our proposed loss formulations lead to an accuracy of 93.2% and 92.0% on USPS $\to$ MNIST and MNIST $\to$ MNISTM, surpassing all other methods and surpassing

[4]Whilst feature matching also suffers from internal covariate shift, it is also only aligning the empirical means between distributions, as discussed in Section 5.2.

Table 5.2: Accuracy for our proposed method compared to the current state-of-the-art. In order to isolate the performance gain from domain adaptation for our proposals, we report in parentheses the percentage increase (relative) over the source-only accuracy, as reported in the respective papers for DIFA [92] and ADDA [91].*UNIT [97] and DTN [88] use additional SVHN data (131 images and 531 images respectively). **This is our implementation of ADDA [91] on MNIST → MNIST-M, as this task is not used in the original paper.

Method	SVHN → MNIST	USPS → MNIST	MNIST → USPS	MNIST → MNIST-M
Source only	0.644	0.597	0.754	0.705
DANN [85]	0.739	0.730	0.771	0.529
DDC [79]	0.681	0.665	0.791	-
DSN [95]	0.827	-	-	0.832
DTN [88]	0.844*	-	-	-
UNIT [97]	0.905*	-	-	-
CoGAN [86]	no convergence	0.891	0.912	-
ADDA [91]	0.760 (26%)	0.901 (58%)	0.894 (19%)	0.800 (14%)**
DIFA [92]	**0.897 (32%)**	0.897 (43%)	**0.923 (28%)**	-
Proposed	0.890 (38%)	**0.932 (56%)**	0.881 (17%)	**0.920 (30%)**

Table 5.3: Accuracy for proposed configurations, compared to state-of-the-art on the Office-31 dataset.

Method	A → W	A → D	D → A
Source only	0.707	0.720	0.581
DANN [85]	0.730	0.723	0.534
DDC [79]	0.618	0.644	0.521
DRCN [94]	0.687	0.668	0.560
JAN [81]	0.752	0.728	0.575
ADDA [91]	0.751	-	-
Proposed	**0.821**	**0.799**	**0.610**

the recently proposed DIFA by 3.5% on USPS → MNIST. DIFA only achieves a marginally higher (0.07%) accuracy on the SVHN → MNIST task, at the expense of additional complexity in feature generation prior to adversarial training. Furthermore, our method achieves achieves a higher relative percentage gain over source only accuracy than DIFA.

5.3.2 Office-31 dataset

We report results on the standard Office-31 [165] dataset in Table 5.3. The Office-31 dataset consists of 4,110 images spread across 31 classes in 3 domains: Amazon, Webcam, and DSLR. Our results focus on the three of the more difficult domain adaptation tasks; Amazon → Webcam (A → W), Amazon → DSLR (A → D) and DSLR $\rightarrow$ Amazon (D $\rightarrow$ A). In order to demonstrate the strength of our proposal, we use VGG-16 pre-trained on ImageNet and fine-tune only the final fully-connected layer. We train with stochastic gradient descent and a learning rate of 0.001. We set the dropout keep probability $z = 0.7$ as in the digits task. Our discriminator is restricted to only 500 hidden units per layer and we only train adversarially for 2k iterations. We note that the number of training parameters is 377 thousand in total, compared to over 6 million utilized for ADDA [91]. Despite only training on a small subset of total parameters, our proposal remains competitive or surpass the performance of other recent methods. We additionally note that under our training setup, ADDA consistently obtains a degenerate solution due to instability during training.

5.3.3 NVS ASL dataset

We introduce a new sign language recognition dataset for NVS-based unsupervised domain adaptation. The primary motivation behind creating the dataset and validating our framework with it is that progress in neuromorphic spike-based event or action recognition is severely hampered from the lack of NVS training data with reliable annotations [168]. This is partially addressed via emulators, which convert annotated APS video datasets into emulated NVS data in order to train advanced discriminative models in a supervised manner. However, beyond the unavoidable gap between the experimental and the emulated NVS data distributions, the NVS camera technology is in constant evolution and new versions of hardware devices like DAVIS and ATIS [169] and their multiple settings cause further domain shift against their previous versions and previously-released software emulation frameworks.

Our experimental dataset is comprised of 1200 unlabelled real recordings and 1200 labelled emulated recordings, each representing a different static sign of 24 letters (A-Y, excluding J) from the America Sign Language (ASL). We note that similar to other APS-based sign language recognition tasks [170], letters J and Z are excluded as their ASL designation requires motion. Figure 5.5 shows the required hand pose for each letter of the dataset. As is evident from the figure, sign

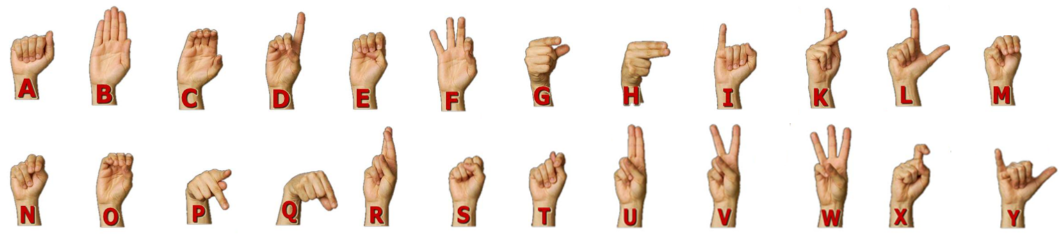


Figure 5.5: Signs for letters A-Z from the American Sign Language dataset. Note that some letters such as M and N only have subtle differences. Letters J and Z are excluded given that they are not static signs and require a particular gesture.

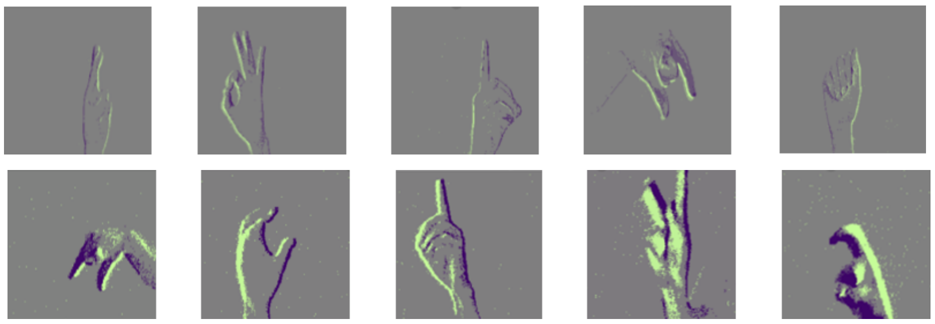


Figure 5.6: (Best viewed in color) Select frames from the NVS American Sign Language recognition dataset. The source domain constitutes emulated NVS frames (top row) which are generated from APS recorded video with PIX2NVS [171]. The target domain constitutes real NVS frames (bottom row) as recorded by an iniLabs DAVIS240c NVS camera. The green/purple points correspond to the +1/-1 (or ON/OFF) spike polarity.

language recognition presents a substantially more difficult task than digit recognition, considering that for some letters (e.g., M and N) there is very little variation in fingers' positioning.

In order to generate the emulated spike events we first record APS video of someone performing the sign for each letter with translational and rotational motion over the video duration, thus increase the difficulty of the recognition task. Next, the APS video is recorded with a standard laptop camera, and consecutive APS frames are passed into the PIX2NVS emulator. PIX2NVS converts the APS frames to the corresponding emulated NVS frames, and this constitutes our source domain. The real NVS recordings are recorded directly with an iniLabs DAVIS240c NVS camera, again with rotational and translational motion over the video duration.

Figure 5.6 shows a selection of emulated (top row) and real NVS frames (bottom row) from the dataset. There is a discernible domain shift between the emulated

Table 5.4: Overall and per-letter recognition accuracy for select letters of the NVS ASL dataset. We evaluate on the source-only, ADDA and our proposed method.

Letter	A	B	E	F	M	N	S	T	U	V	overall
Source only	0.928	0.738	0.800	0.124	0.127	0.414	0.846	0.157	0.302	0.996	0.611
ADDA	0.988	0.962	0.261	0.865	0.152	**0.808**	**0.810**	0.562	**0.954**	1.000	0.837
Proposed	**1.000**	**0.966**	**0.831**	**0.950**	**0.418**	0.670	0.771	**0.747**	0.942	**1.000**	**0.875**

and real spike events, with the real NVS events exhibiting a substantially higher spike density that increases the visibility of the signed letter, despite also carrying some background 'salt & pepper' noise. Nonetheless, we are able to demonstrate that our proposed method can reduce this domain shift. As the recordings represent static signs we train on individual frames and remove a subset of frames from the start and end of the recording where there may be no sign distinguishable. As such, we have $\sim 80,000$ emulated NVS frames for source training and $\sim 50,000$ real NVS frames. We use ~ 40000 of the real NVS frames for domain adaptation and ~ 10000 for inference. The frame resolution in both domains is 240×180. Our source encoder is VGG16 [57], which we train in step 1 on the emulated NVS frames using stochastic gradient descent with momentum set to 0.9. The learning rate is set to 0.001, the batch size to 24 and we complete training at 15k iterations. In terms of data augmentation, we first resize the input such that the smaller side is 256 and keep the aspect ratio. We then use a multi-scale random cropping of the resized RGB frame; the cropped volume is subsequently randomly flipped, and normalized according to its mean. In step 2, we initialize the target encoder from the source pre-trained weights and follow the same procedure with data augmentation on both input domains, but only train the target encoder fully connected layers adversarially for 10k iterations and fix all convolutional layers. Contrary to the APS datasets, the discriminator is again restricted to 500 hidden units per layer. We infer on the target dataset by extracting a single center crop.

We present both the overall and per-letter recognition accuracy in Table 5.4 when evaluating on the NVS ASL dataset. For clarity, we only include letters in the table with subtle differences in sign configuration such as M and N. We include results on our proposed framework, with both variants of target encoder loss function, along with results when evaluating with the source only. Our proposal provides substantial increase in accuracy compared to training on the source only, and also outperforms ADDA on most letters and overall, by 4%. By looking at the per-letter accuracies we can distinguish where ADDA substantially underperforms compared to our proposal; namely, on the set $\boldsymbol{C} = \{\mathrm{E,M,N,S,T}\}$. If we cross-reference with Figure 5.5, we note that the letters in this set are not easily distinguishable from each other, which would be made worse when transforming to the NVS domain. Whereas ADDA generally performs poorly on all letters in the set, and effectively misclassifies the majority of instances of the letters E and M in order to align the target to the source domain, our proposal is able to better transfer some of the class separation learned from training on the source domain with labels and maintains consistently good accuracy across the set.

5.4 Conclusion

We extend adversarial discriminative domain adaptation by explicitly accounting for task knowledge in the discriminator during adversarial training and leveraging on the fixed distribution over source encoder posteriors, with which we derive reconstruction and MMD loss formulations for adversarial training. In particular, we consider the discriminator as a denoising autoencoder with a reconstruction loss function and minimize the maximum mean discrepancy between the discriminator posterior and source encoder posterior distribution to train the encoder. We compare and analyse how our method improves over conventional semi-supervised GAN loss formulations. Our framework is shown to compete or outperform the state-of-the-art in unsupervised transfer learning on standard datasets, while remaining simple and intuitive to use. Finally, we show that our proposal allows for unsupervised domain adaptation between emulated and real neuromorphic spike events for a sign language recognition application, in order to improve performance compared to source training only. As such, the recognition model from Chapter 4 can be embedded into this domain adaptation framework in order to improve generalization from the emulated to real NVS domain.

Chapter 6

Conclusions

In this thesis we have successfully transitioned from unimodal learning in the pixel domain to efficient multimodal recognition in the spike domain. Notably, we have worked to improve the trade-off between recognition performance and computational overhead that inevitably increases when adding additional processing or modalities into a framework. In the case of multimodal recognition in particular, we have worked to design frameworks that are able to leverage on compressed motion representations such as motion vectors or neuromorphic vision sensing (NVS) events that offer improved sensing efficiency and acquisition complexity. Specifically for NVS based recognition, we have worked to overcome the scarcity of real labelled NVS data by generating emulated data and worked to minimize the domain shift between real and emulated datasets via unsupervised domain adaptation.

We began in Chapter 2 with the content based image retrieval (CBIR) task, which involves finding matching images to a given query from a corpus of images. In order to improve retrieval performance on arbitrary sized regions-of-interest (ROI), we propose to additionally process the RGB images with an multi-scale spatial Voronoi partitioning, which extracts more semantically relevant blocks than a conventional grid based spatial partitioning. We showed that with a combination of a tree-pruning mechanism over partitions and product quantization of the cell representations, our proposed Voronoi encoded variants can improve performance with ROI queries and also generalizes well to whole image queries. Finally, we demonstrated the capability of our method by designing and implementing a large scale image video frame retrieval system, as a deliverable to our industrial partner BAFTA Research.

In Chapter 3 we moved from CBIR to action recognition, where multimodal learning is commonplace, with many recent methods modelling spatial and temporal dependencies with flow and RGB modalities respectively. The flow is typically derived from the pixel domain by computing partial derivatives between consecutive frames under a brightness consistency constraint or generating with CNN; however, in both cases this presents a major bottleneck in the recognition pipeline. We proposed to circumvent this by extracting the motion vectors (MVs) directly from the video codec and adopting them as both a flow approximation to replace conventional dense optical flow and a spatio-temporal activity sensor for only selectively decoding regions of RGB frames which correspond to non-zero macroblocks; i.e., regions with activity. We ultimately designed a low-cost multimodal framework for processing the the selectively decoded frames and motion vectors.

In Chapter 4, we considered replacing the APS-derived motion stream with a neuromorphic vision sensing (NVS) stream for the action recognition task. APS hardware is restricted by power and framerate constraints, which limits the processing capabilities for high-speed recognition in commercial application such as drone surveillance. These restrictions are effectively removed with asynchronous event based sensing. We addressed the scarcity of labelled NVS data for training - we showed how conventional NVS emulators can be repurposed for the recognition task, by first proposing new emulator options for minimizing the domain shift between the emulated and real data. We then addressed the inherent sparsity of NVS representations by leveraging on the large amount of labelled optical flow data to train a teacher-student framework between flow and emulated NVS frames - for the first time showing competitive accuracy for action recognition when inferring on NVS data singly and jointly with only a single shot of RGB frames.

In Chapter 5, we revisit the domain shift problem between emulated and real NVS events. We presented this as an unsupervised domain adaptation problem, where there is a scarcity of labelled data in the real NVS target domain compared to the emulated NVS source domain. Building upon recent work in adversarial methods for domain adaptation, we proposed a novel solution to the domain adaptation problem, in which we embedded task knowledge into the discriminator and leveraged on the fixed source encoder posterior in order to stabilize and improve convergence during the adversarial training. We showed that our proposed method is able to surpass or compete with existing domain adaptation methods in the pixel domain on standard domain adaptation datasets. In order to verify that this performance extends to the spike domain, we introduced a new neuromorphic sign language dataset, on which we are able to improve on the source only accuracy by over 20%.

6.1 Major Challenges

There were two main challenges that had to be overcome during the course of the PhD. Most of the current state-of-the-art methods in content-based image retrieval and action recognition are deep learning-based and rely on GPU processing for efficient computation. Given that at the start of the PhD we did not have access to a GPU cluster for training, we decided to focus on shallow learning methods such as VLAD for content-based image retrieval that could efficiently be trained on a CPU. This led to the Voronoi VLAD descriptors discussed in Chapter 2, which we eventually extended to Voronoi deep-learning descriptors. However, these deep learning descriptors were derived from small ImageNet pretrained CNN models that did not require additional training for ‘weakly supervised’ retrieval and could efficiently perform inference on a CPU. After completing the work in Chapter 2 and moving towards multimodal action recognition, we began to use AWS instances for training on a GPU, along with a GPU cluster that was funded through the Industrial Fellowship by the Royal Commission for the Exhibition of 1851. Nonetheless, whilst being a challenge, starting with fewer resources helped in appreciating the importance of complexity saving, which constitutes one of the core motivations in this thesis. The other challenge arose from dealing with the lack of structure and independent learning that is in some ways an implicit part of the process when undertaking a PhD. This was initially a problem at the start of the PhD, as the goal of obtaining a PhD at that point seemed so distant. However, as we began to submit and publish papers, focus was shifted to paper submissions as intermediate goals to work towards.

6.2 Future Work

There are a number of ways the work presented in this thesis can be extended in future work. We note that in the context of deep learning, our proposed Voronoi encoded CNN descriptors in Chapter 2 are currently restricted to weakly supervised retrieval, where the networks are pre-trained on an auxiliary task such as ImageNet. In addition, the current method of Voronoi partitioning does not account for background noise, especially with a simple feature detector that could distort the partitions. If the feature detection and Voronoi partitioning was trainable in an end-to-end framework, in which we could learn what better constitutes a 'region-of-interest' in the image and tune the partitioning around these regions.

Due to limited resources, for our work on multimodal action recognition we were only able to train on datasets UCF-101 and HMDB-51, which have since been surpassed in size by the substantially larger Kinetics dataset. In future work, it would be interesting to see whether performance of our motion vector (MV)-based action recognition framework in Chapter 3 scales up to these larger datasets, particularly due to the low resolution of the MV frames, which are quantized by a factor of the macroblock size. It is worth noting that following the work completed in Chapter 3, we have since published work which considers how we can adaptively tune the quantization parameter (QP) in the macroblock bitstream, in order to further reduce the required bitrate, whilst maintaining competitive recognition performance.

The work in Chapters 4 and 5 paves the way for more commercially viable NVS based recognition and tracking systems that are able to capitalize on the advantages of the NVS camera over APS, in order to achieve superhuman levels of performance. However, we note that our current implementation for NVS frame generation foregoes one of the advantages of NVS cameras; the high frame-rate.

Namely, in order to generate denser NVS frame representations for CNN training, we aggregate per-pixel events over time, at the video frame-rate. This is necessary for learning from optical flow in our teacher student framework. It would be interesting to see if we could extend the framework to work directly on the NVS event stream, or design a more adaptive frame grouping based on the motion activity in the video. Finally, in Chapter 5, whilst we demonstrate that our proposed method for unsupervised domain adaptation generalizes and substantially improves accuracy on both NVS and APS datasets, we note that our method currently forgoes some of the class separability on more easily confused classes for a better domain separation, as is evident from the NVS results; we intend to work on maintaining more of this class separability in future work.

Appendix A

Level Projection for VE Storage Compaction

In order to decrease the storage requirements for unquantized Voronoi-based encoded (VE) representations in Chapter 2, the descriptor over two constituent cells x and y (i.e., spatially-neighboring cells belonging to the same cell of the upper level), can be approximated as:

$$\widetilde{\boldsymbol{\phi}}_{x\cup y} = \widetilde{\boldsymbol{\phi}}_x + \widetilde{\boldsymbol{\phi}}_y. \tag{A.1}$$

This holds because both PCA and whitening are linear mappings, therefore, if we do not consider the vector truncation and subsequent normalization of the individual cell vectors, the additivity property holds in the projected domain as well. Given that directionality is preserved under normalization, (A.1) provides an approximation to the normalized encoding computed directly over the two cells. Therefore, we can trade-off computation for memory by solely storing the last-level PCA-projected descriptors (level $L-1$) and computing all other cell encodings for all lower levels at runtime via repetitive application of (A.1) amongst constituent cells and renormalizing before carrying out the similarity measurement of (1.3). This is an appealing proposition for practical systems because vectorized addition and scaling for normalization is extremely inexpensive in modern SIMD-based architectures. As such, this approach requires storing only $\prod_{l=1}^{L-1} V_l$ cell descriptors, instead of V_{tot} cell vectors. Naturally, there is a dependency on the projection error, which will evidently be greater with less dimensions retained post-PCA.

We can integrate product quantization with a modified level projection for quantized VE storage compaction. As before, we only store the last-level (quantized) descriptors offline. However, as the inner product satisfies the distributive law, we should now directly approximate the inner product between a query encoding and a level $l-1$ cell descriptor as an L_1 normalized summation:

$$S_{\mathrm{ROI},l-1} = \frac{1}{V_l}\sum_{i=1}^{V_l} S_{\mathrm{ROI},l_i \in l-1} \tag{A.2}$$

where each inner product is read from a look up table.

Appendix B

Proof of Proposition 1

Proof. In order to optimize the bit allocation to the various descriptions (subspaces), we optimize the rate distortion expression:

$$R(E) = \min_{\sum E_m = E} \sum_{m=1}^{M} \log \frac{|\mathbf{\Sigma}_m|}{E_m} \tag{B.1}$$

where E_m is the product of the sub-component distortions, $|\mathbf{\Sigma}_m|$ is the determinant of the covariance matrix $\mathbf{\Sigma}_m$ and E is the overall distortion value. The minimum rate for given E is derived when all distortions E_m are equal, i.e., $E_m = E/M$.

Using results derived from rate distortion theory [172], E_m for the m-th subspace can be approximated by:

$$E_m \approx |\mathbf{\Sigma}_m| \prod_{i=1}^{D'} h_{im} 2^{-2b_{im}} = |\mathbf{\Sigma}_m| h'_m 2^{-2B'_m} \tag{B.2}$$

where $h_{im} = \frac{1}{12}\left\{\int_{-\infty}^{\infty} f_{im}(x)\ dx\right\}^3$ is a variable determined by the univariate Gaussian of the normalized components, $f_{im}(x)$, and b_{im} is the average number of bits encoded per dimension. Due to the independence property, the product of h_{im} in the m-th subspace yields the variable h'_m, which is now determined by the multivariate Gaussian distribution for the normalized subspace random vectors. This distribution is independent of subspace, and, as such, h'_m is constant for all m. Similarly, if the size of the bit encoding and block dimension D' is fixed per subspace, then B'_m is a constant for all m. For E_m to be equal for all m, $|\mathbf{\Sigma}_m|$ must be constant, independent of subspace. $\square$

Appendix C

Retrieval System Implementation

Based on the work detailed in Chapter 2, we have developed a two-component retrieval system within the remit of the VideoClarity project, a BAFTA research project sponsored by Innovate UK (101932). The system:

1. Extracts Fast-VDCNN signatures and thumbnails from a large corpus

2. Returns matches to queries from the corpus based on similarity

Fundamentally, the system has been extended to work on both images and videos, by breaking down videos into representative frames using shot/scene detection and extracting the Fast-VDCNN signature on representative frames. Signature extraction for the database videos is performed offline. Representative frames are extracted using shot detection with over-compensation. The process flow is highlighted in Figure C.1, with the retrieval system represented as a 'black box'. In this example, the user feeds in a screenshot from the a clip of the film 'Alien'. The screenshot is queried against a corpus of video folders where each folder can contain mutliple videos/screenshots. Per folder, the Fast-VDCNN signatures for listed video content is contained in a binary file and indexed in a corresponding text file. The retrieval system outputs the thumbnails for matching frames, with their name and timestamp, thus providing both matching and localisation utilities. Two versions of the system have currently been delivered to BAFTA, which respectively use unquantized and product quantized signatures (with lookup tables).

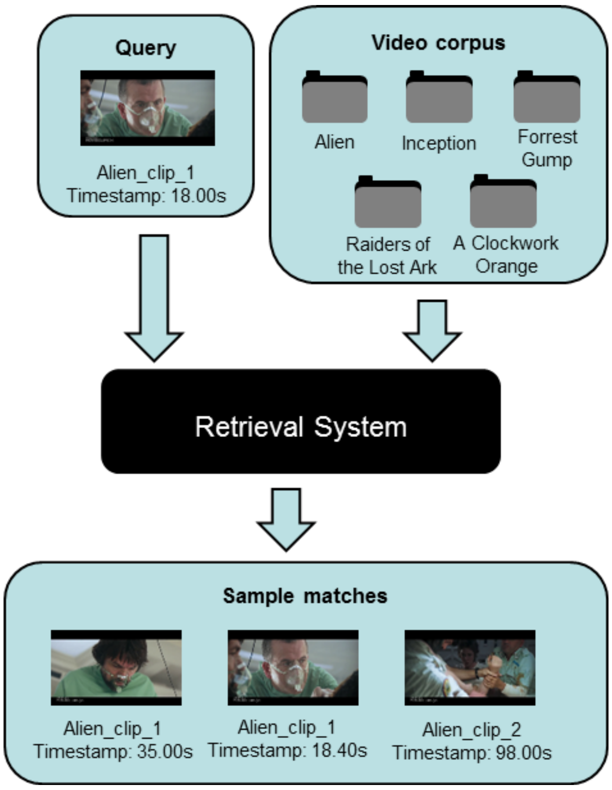


Figure C.1: High Speed Analysis of Big Video Data: Retrieval System

Such a system is intended to integrate with an overarching web interface, where a user could upload a query video to match to a database of videos crawled offline from various sites. The matching process is expected to be performed online and in real time. Notably, the system has been tested by the industrial partner, BAFTA Research and has exhibited state-of-the-art performance. In particular:

1. Video signature extraction is ˜700x faster than real-time playback using a single 32 core server

2. The system can search through 232 hours of content within 20 seconds

Bibliography

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Rec. (CVPR)*, pages 2911–2918, 2012.

[2] Jiajun Liu, Zi Huang, Hongyun Cai, Heng Tao Shen, Chong Wah Ngo, and Wei Wang. Near-duplicate video retrieval: Current research and future trends. *ACM Computing Surveys (CSUR)*, 45(4):44, 2013.

[3] Xiangyang Xu, Wenjing Geng, Ran Ju, Yang Yang, Tongwei Ren, and Gang-shan Wu. Obsir: Object-based stereo image retrieval. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.

[4] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1116–1124, 2015.

[5] Cheng-Bin Jin, Shengzhe Li, and Hakil Kim. Real-time action detection in video surveillance using sub-action descriptor with multi-cnn. *arXiv preprint arXiv:1710.03383*, 2017.

[6] Guido Borghi, Roberto Vezzani, and Rita Cucchiara. Fast gesture recognition with multiple stream discrete hmms on 3d skeletons. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 997–1002. IEEE, 2016.

[7] Fahimeh Rezazadegan, Sareh Shirazi, Ben Upcroft, and Michael Milford. Action recognition: From static datasets to moving robots. *arXiv preprint arXiv:1701.04925*, 2017.

[8] Tobi Delbrück, Bernabe Linares-Barranco, Eugenio Culurciello, and Christoph Posch. Activity-driven, event-based vision sensors. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 2426–2429. IEEE, 2010.

[9] E Neftci, C Posch, and E Chicca. Neuromorphic engineering. *Comput. Intel.-Vol. II*, page 278, 2015.

[10] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Trans. Neural Netw. and Learn. Syst.*, 25(2):407–417, 2014.

[11] Quentin Sabatier, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based fourier analysis. *IEEE Trans. on Image Processing*, 26(5):2192–2202, 2017.

[12] Vanya V Valindria, Ioannis Lavdas, Wenjia Bai, Konstantinos Kamnitsas, Eric O Aboagye, Andrea G Rockall, Daniel Rueckert, and Ben Glocker. Domain adaptation for mri organ segmentation using reverse classification accuracy. *arXiv preprint arXiv:1806.00363*, 2018.

[13] Jianfei Yu and Jing Jiang. A hassle-free unsupervised domain adaptation method using instance similarity features. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 168–173, 2015.

[14] Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. Ninth IEEE Int. Conf. on Comput. Vis.*, pages 1470–1477. IEEE, 2003.

[15] H. Jégou and O. Chum. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *Proc. Europ. Conf. in Comput. Vis. (ECCV)*, pages 774–787. Springer, 2012.

[16] Ondřej Chum and Jiří Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *Proc. IEEE Int. Conf. on Comput. Vis. and Pat. Recogn. (CVPR)*, pages 3416–3423. IEEE, 2010.

[17] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. IEEE, 2010.

[18] J. Philbin et al. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Rec.*, pages 1–8, 2008.

[19] Jan C Van Gemert, Jan-Mark Geusebroek, Cor J Veenman, and Arnold WM Smeulders. Kernel codebooks for scene categorization. In *Computer Vision–ECCV 2008*, pages 696–709. Springer, 2008.

[20] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Rec.*, pages 3304–3311, 2010.

[21] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 34(9):1704–1716, 2012.

[22] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[23] R. Arandjelovic and A. Zisserman. All about VLAD. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 1578–1585, 2013.

[24] Zixuan Wang, Wei Di, Anurag Bhardwaj, Vignesh Jagadeesh, and Robinson Piramuthu. Geometric vlad for large scale image search. *arXiv preprint arXiv:1403.3829*, 2014.

[25] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, 2007.

[26] Gabriela Csurka and Florent Perronnin. An efficient approach to semantic segmentation. *International Journal of Computer Vision*, 95(2):198–212, 2011.

[27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 248–255. IEEE, 2009.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[29] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Europ. Conf. in Comput. Vis. (ECCV)*, pages 584–599. Springer, 2014.

[30] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1790–1802, 2016.

[31] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 3384–3391, 2010.

[32] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Europ. Conf. in Comput. Vis. (ECCV)*, pages 392–407. Springer, 2014.

[33] Aaron Chadha and Yiannis Andreopoulos. Voronoi-based compact image descriptors: Efficient region-of-interest retrieval with vlad and deep-learning-based descriptors. *arXiv preprint arXiv:1611.08906*, 2016.

[34] Mattis Paulin, Julien Mairal, Matthijs Douze, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Convolutional patch representations for image retrieval: an unsupervised approach. *arXiv preprint arXiv:1603.00438*, 2016.

[35] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–61, 2015.

[36] K. Mikolajczyk et al. A comparison of affine region detectors. *Int. J. of Comput. Vis.*, 65(1-2):43–72, 2005.

[37] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

[38] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.

[39] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[40] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*, pages 3–20. Springer, 2016.

[41] Yu-An Chung and Wei-Hung Weng. Learning deep representations of medical images using siamese cnns with application to content-based image retrieval. *arXiv preprint arXiv:1711.08490*, 2017.

[42] Liang Zheng, Shengjin Wang, Ziqiong Liu, and Qi Tian. Fast image retrieval: Query pruning and early termination. *IEEE Trans. Multimedia*, 17(5):648–659, 2015.

[43] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[44] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[45] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 1–8. IEEE, 2008.

[46] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[47] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007.

[48] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[49] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[50] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 33(1):117–128, 2011.

[51] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 3169–3176. IEEE, 2011.

[52] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 3551–3558, 2013.

[53] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 33(3):500–513, 2011.

[54] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proc. Scand. Conf. on Image Anal.*, pages 363–370. Springer, 2003.

[55] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 1725–1732, 2014.

[56] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 4489–4497, 2015.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402 and CRCV-TR-12-01*, Nov. 2012.

[59] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. *arXiv preprint arXiv:1712.08416*, 2017.

[60] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. Advances in Neural Inf. Process. Syst. (NIPS)*, pages 568–576, 2014.

[61] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 1933–1941, 2016.

[62] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, volume 2, page 3, 2017.

[63] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *Proc. Europ. Conf. on Comp. Vis. (ECCV)*, pages 20–36. Springer, 2016.

[64] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE Trans. Patt. Anal. Mach. Intel.*, to appear.

[65] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Proc. Int. Worksh. Human Behav. Underst.*, pages 29–39. Springer, 2011.

[66] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 2625–2634, 2015.

[67] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 4694–4702, 2015.

[68] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[69] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 843–852, 2015.

[70] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2593–2600, 2014.

[71] Bowen Zhang et al. Real-time action recognition with enhanced motion vector CNNs. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 2718–2726, 2016.

[72] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. *arXiv preprint arXiv:1712.00636*, 2017.

[73] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 2593–2600, 2014.

[74] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. IEEE, 2017.

[75] Hildegard Kuehne et al. HMDB: a large video database for human motion recognition. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 2556–2563. IEEE, 2011.

[76] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271, 2007.

[77] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.

[78] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[79] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[80] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[81] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, 2016.

[82] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[83] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017.

[84] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018.

[85] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[86] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[87] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[88] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[89] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[90] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[91] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017.

[92] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. *arXiv preprint arXiv:1711.08561*, 2017.

[93] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *arXiv preprint arXiv:1712.02560*, 3, 2017.

[94] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[95] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

[96] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

[97] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[98] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. *arXiv preprint arXiv:1808.00948*, 2018.

[99] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 1–8, 2007.

[100] S. Lazebnik et al. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Rec.*, volume 2, pages 2169–2178, 2006.

[101] Eleni Mantziou, Symeon Papadopoulos, and Yiannis Kompatsiaris. Large-scale semi-supervised learning by approximate laplacian eigenmaps, vlad and pyramids. In *14th Int. Workshop on Image Anal. for Mult. Interactive Services (WIAMIS)*, pages 1–4. IEEE, 2013.

[102] Renhao Zhou, Qingsheng Yuan, Xiaoguang Gu, and Dongming Zhang. Spatial pyramid vlad. In *IEEE Vis. Comm. and Image Proc. Conf.*, pages 342–345. IEEE, 2014.

[103] Hossein Azizpour, Ali Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. 2014.

[104] Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn. Workshops*, pages 806–813, 2014.

[105] H. Jegou et al. Hamming embedding and weak geometric consistency for large scale image search. In *Europ. Conf. in Comput. Vis.*, pages 304–317. Springer, 2008.

[106] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, volume 2, pages II–264, 2003.

[107] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[108] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. Europ. Conf. in Comput. Vis. (ECCV)*, pages 128–142. Springer, 2002.

[109] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Europ. Conf. in Comput. Vis.*, pages 430–443. Springer, 2006.

[110] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[111] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[112] Jonathan Brandt. Transform coding for fast approximate nearest neighbor search in high dimensions. In *Proc. IEEE Int. Conf. on Comput. Vis. and Patt. Recogn.*, pages 1815–1822. IEEE, 2010.

[113] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 36(4):744–755, 2014.

[114] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[115] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Ioannis Yiannis Kompatsiaris, Grigorios Tsoumakas, and Ioannis Vlahavas. A comprehensive study over vlad and product quantization in large-scale image retrieval. *IEEE Trans. Multimedia*, 16(6):1713–1728, 2014.

[116] Jan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE Int. Conf. on Comput. Vis. Workshops (ICCVW)*, pages 554–561. IEEE, 2013.

[117] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3743–3752, 2015.

[118] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[119] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, pages 1–9, 2015.

[120] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circ. and Syst. for Video Technol.*, 13(7):560–576, 2003.

[121] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding HEVC standard. *IEEE Trans. Circ. and Syst. for Video Technol.*, 22(12):1649–1668, 2012.

[122] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. The latest open-source video codec vp9-an overview and preliminary results. In *Picture Coding Symposium (PCS), 2013*, pages 390–393. IEEE, 2013.

[123] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. Europ. Conf. on Comp. Vis. (ECCV)*, pages 25–36. Springer, 2004.

[124] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *Proc. IEEE Conf. Comp. Vis. Pattern Rec. (CVPR)*, 2017.

[125] Miguel T Coimbra and Mike Davies. Approximating optical flow within the MPEG-2 compressed domain. *IEEE Trans. Circ. and Syst. for Video Technol.*, 15(1):103–107, 2005.

[126] FFMPEG LibAVCodec documentation. *online, available at: http://ffmpeg.org/libavcodec.html*.

[127] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1026–1034, 2015.

[128] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. Europ. Conf. on Comp. Vis. (ECCV)*, pages 818–833. Springer, 2014.

[129] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[130] Ke Xu, Xinghao Jiang, and Tanfeng Sun. Two-stream dictionary learning architecture for action recognition. *IEEE Trans. on Circ. and Syst. for Video Technol.*, 27(3):567–576, 2017.

[131] Shichao Zhao, Yanbin Liu, Yahong Han, Richang Hong, Qinghua Hu, and Qi Tian. Pooling the convolutional layers in deep convnets for video action recognition. *IEEE Trans. on Circ. and Syst. for Video Technol.*, to appear.

[132] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015.

[133] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[134] Yuhuang Hu, Hongjie Liu, Michael Pfeiffer, and Tobi Delbruck. DVS benchmark datasets for object tracking, action recognition, and object recognition. *Front. Neurosc.*, 10, 2016.

[135] Tobi Delbruck. Neuromorphic vision sensing and processing. In *Proc. Europ. Solid-State Dev. Res. Conf.*, pages 7–14. IEEE, 2016.

[136] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbrück. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *Proc. Int. Conf. Event-based Contr., Comm., and Sig. Process., EBCCSP*, pages 1–8. IEEE, 2016.

[137] K. D Fischl et al. Neuromorphic self-driving robot with retinomorphic vision and spike-based processing/closed-loop control. In *Information Sciences and Systems (CISS), 2017 51st Annual Conference on*, pages 1–6. IEEE, 2017.

[138] Iulia-Alexandra Lungu, Federico Corradi, and Tobi Delbrück. Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–1. IEEE, 2017.

[139] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *arXiv preprint arXiv:1610.08336*, 2016.

[140] Cheston Tan, Stephane Lallee, and Garrick Orchard. Benchmarking neuromorphic vision: lessons learnt from computer vision. *Front. Neurosc.*, 9:374, 2015.

[141] Jose Antonio Perez-Carrasco, Bo Zhao, Carmen Serrano, Begona Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabe Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets. *IEEE Trans. Patt. Anal. Mach. Intel.*, 35(11):2706–2719, 2013.

[142] Garrick Orchard et al. HFirst: a temporal approach to object recognition. *IEEE Trans. on Patt. Anal. Mach. Intel.*, 37(10):2028–2040, 2015.

[143] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosc.*, 9, 2015.

[144] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[145] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[146] Matthew L Katz, Konstantin Nikolic, and T Delbruck. Live demonstration: Behavioural emulation of event-based vision sensors. In *Proc. IEEE Int. Symp. on Circ. and Syst.*, pages 736–740. IEEE, 2012.

[147] Yin Bi and Yiannis Andreopoulos. PIX2NVS: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams. *Proc. IEEE Int. Conf. Image Process., ICIP*, 2017.

[148] G. Pineda García, P. Camilleri, Q. Liu, and S. Furber. pyDVS: An extensible, real-time dynamic vision sensor emulator using off-the-shelf hardware. In *Proc. IEEE Symp. Ser. Comp. Intel. (SSCI), 2016*, pages 1–7. IEEE, 2016.

[149] Necip Sayiner, Henrik V Sorensen, and Thayamkulangara R Viswanathan. A level-crossing sampling scheme for a/d conversion. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(4):335–339, 1996.

[150] Basabdatta Sen Bhattacharya and Stephen B Furber. Biologically inspired means for rank-order encoding images: A quantitative analysis. *IEEE Trans. Neural Netw.*, 21(7):1087–1099, 2010.

[151] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comp. Vis.*, 40(2):99–121, 2000.

[152] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[153] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pages 3468–3476, 2016.

[154] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017.

[155] Gul Varol, Ivan Laptev, and Cordelia Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[156] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[157] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[158] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[159] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[160] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[161] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[162] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

[163] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[164] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[165] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[166] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[167] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[168] Cheston Tan, Stephane Lallee, and Garrick Orchard. Benchmarking neuromorphic vision: lessons learnt from computer vision. *Front. Neurosc.*, 9:374, 2015.

[169] Tobi Delbruck. Neuromorphic vision sensing and processing. In *Solid-State Device Research Conference (ESSDERC), 2016 46th European*, pages 7–14. IEEE, 2016.

[170] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time asl fingerspelling recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1114–1119. IEEE, 2011.

[171] Yin Bi and Yiannis Andreopoulos. Pix2nvs: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 1990–1994. IEEE, 2017.

[172] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.