# EFFICIENCY IN DEEP LEARNING

IMAGE AND VIDEO DEEP MODEL EFFICIENCY

# EFFICIENCY IN DEEP LEARNING

## IMAGE AND VIDEO DEEP MODEL EFFICIENCY

**Dissertation**

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 16, May 2024 at 10:00 o'clock

by

**Xin LIU**

Master of Science in Embedded Systems, Delft University of Technology, The Netherlands
born in Sichuan, China

This dissertation has been approved by the promotors.

ISBN 978-94-6469-947-0

For my grandfather

For my parents

For Yucheng

# Contents

# SUMMARY

Deep learning is the core algorithmic tool for automatically processing large amounts of data. Deep learning models are defined as a stack of functions (called layers) with millions of parameters, that are updated during training by fitting them to data. Deep learning models have show remarkable accuracy gains on visual problems in video and images. Yet at the same time, this comes at a considerable computational cost that raises concerns about energy consumption. The escalation in the number of parameters and the surging demand for extensive data exacerbate these concerns. This thesis delves into the core of these concerns, proposing innovative techniques to enhance the efficiency of deep learning models. This thesis starts with exploring efficient deep learning models for video data, followed by efficient models for image data.

For video data, efficiently identifying and localizing objects across consecutive frames in videos – video object detection, has useful applications in everyday life such as safer traffic, or medical assisted systems. This thesis leverages the inherent motion within videos, introducing a method that anticipates object locations from a single video frame over multiple future frames. By limiting the frames processed in the deep learning model to only a subset of frames, it markedly reduces computational overhead. This approach leads to not only enhanced efficiency, but also higher precision by implicitly incorporating motion smoothness constraints. Furthermore, while still in the context of video data, this thesis investigates action recognition in video sequences. Existing prior methods often resort to sub-sampled video frames to reduce deep learning computations. In contrast, this thesis advocates using of all video frames during deep learning model fitting. This novel approach reduces model computational demands by making the observations that similar video frames should contribute similarly to the deep model parameter updates, and therefore their function responses can be accumulated. My work on efficient video analysis has led to great speed improvements, and serves as a good starting point for future work.

For image data, this thesis focuses on two tasks: cross-domain image matching, and image classification. Cross-domain image matching is the task of matching images coming from two different recording settings (domains) such as day-time and night-time. Deep models for realistic image data need to be able to recognize semantics in images across domains. Moreover deep models rely on annotations — tags naming the categories present in the images: *e.g.* 'dog', 'car', 'person', *etc.*, which are expensive to obtain. This thesis addresses both these challenges. The proposed method introduces a robust strategy blending deep models for image matching with methods that adapt across multiple domains. And it deftly handles domain disparities and outliers, and more notably is data-efficient as it only needs sparse image annotations. Finally, in the context of recognizing image categories, this thesis defines a new type of deep model layer to be used for normalization that is data-efficient. This stands in contrast to standard normalization layers such as BatchNorm, which makes deep model fitting efficient, however require many data samples at once. The proposed new normalization layer sidesteps the dependency on sample statistics and operates independently of the number of input data samples yielding memory efficiency. My work on efficient image analysis has demonstrated great data-efficiency, and it represents valuable inspiration for future work.

Taken together, this thesis is a comprehensive exploration of deep learning efficiency, spanning image and video data. The proposed approaches have shown valuable reduction in computational and data demands. The work presented here has raised the standards of what useful deep models should accomplish, and helped the field move in a better, more responsible direction.

# SAMENVATTING

Diep leren is de belangrijkste algoritmische tool voor het automatisch verwerken van grote hoeveelheden gegevens. Deep learning-modellen worden gedefinieerd als een stapel functies (lagen genoemd) met miljoenen parameters, die moeten worden bijgewerkt door ze aan te passen aan de gegevens. Dit ontwerp maakt ze opmerkelijk, vooral voor het oplossen van visuele problemen met video- en beeldgegevens. Maar tegelijkertijd brengt dit aanzienlijke rekenkosten met zich mee die zorgen baren over het energieverbruik. De escalatie van het aantal parameters en de stijgende vraag naar uitgebreide gegevens verergeren deze zorgen. Dit proefschrift gaat dieper in op de kern van deze zorgen en stelt innovatieve technieken voor om de efficiëntie van deep learning-modellen te verbeteren. Dit proefschrift begint met het verkennen van efficiënte deep learning-modellen voor videodata, gevolgd door efficiënte modellen voor beelddata.

Voor videogegevens heeft het efficiënt identificeren en lokaliseren van objecten over opeenvolgende frames in video's - detectie van video-objecten, nuttige toepassingen in het dagelijks leven, zoals veiliger verkeer of medische ondersteunde systemen. Dit proefschrift maakt gebruik van de inherente beweging in video's en introduceert een methode die objectlocaties anticipeert vanuit een enkel videoframe over meerdere toekomstige frames. Door de frames die in het deep learning-model worden verwerkt te beperken tot slechts een subset van frames, wordt de rekenkundige overhead aanzienlijk verminderd. Deze aanpak leidt niet alleen tot verbeterde efficiëntie, maar ook tot hogere precisie door impliciet beperkingen voor de soepelheid van bewegingen op te nemen. Bovendien, nog steeds in de context van videodata, onderzoekt dit proefschrift de herkenning van menselijke acties in videosequenties. Bestaande eerdere methoden nemen vaak hun toevlucht tot sub-sampled videoframes om deep learning-berekeningen te verminderen. In tegenstelling hiermee pleit dit werk voor het gebruik van alle videoframes tijdens het aanpassen van deep learning-modellen. Deze nieuwe benadering vermindert de rekeneisen van het model door de waarnemingen te doen dat vergelijkbare videoframes op dezelfde manier zouden moeten bijdragen aan de diepe modelparameterupdates, en daarom kunnen hun functiereacties worden geaccumuleerd. Mijn werk aan efficiënte video-analyse heeft geleid tot grote snelheidsverbeteringen en dient als een goed startpunt voor toekomstig werk.

Voor beelddata richt dit proefschrift zich op twee taken: cross-domein beeldmatching en beeldclassificatie. Cross-domain image matching is de taak om beelden te matchen die afkomstig zijn van twee verschillende opname-instellingen (domeinen), zoals overdag en 's nachts. Diepe modellen voor realistische beeldgegevens moeten semantiek in beelden over domeinen heen kunnen herkennen. Bovendien vertrouwen diepe modellen op annotaties — tags die de categorieën in de afbeeldingen benoemen: *bijv.*. hond, auto, persoon, *etc.*, die duur zijn om te verkrijgen. Dit proefschrift behandelt beide uitdagingen. De voorgestelde methode introduceert een robuuste strategie die diepe modellen voor het matchen van afbeeldingen combineert met methoden die zich over meerdere domeinen aanpassen. En

het gaat behendig om met domeinverschillen en uitschieters, en is met name data-efficiënt omdat het slechts schaarse beeldannotaties nodig heeft. Tot slot, in de context van het herkennen van afbeeldingscategorieën, definieert dit proefschrift een nieuw type diepe modellaag die gebruikt kan worden voor normalisatie en die data-efficiënt is. De standaard normalisatielagen versterken de diepe modelaanpassing, maar ze vereisen veel gegevensmonsters tegelijk. De voorgestelde nieuwe normalisatielaag omzeilt de afhankelijkheid van steekproefstatistieken. Door dit te doen, werkt de methode onafhankelijk van het aantal invoergegevensmonsters en bereikt zowel betrouwbaarheid als geheugenefficiëntie. Mijn werk op het gebied van efficiënte beeldanalyse heeft een grote gegevensefficiëntie aangetoond en vormt een waardevolle inspiratie voor toekomstig werk.

Alles bij elkaar genomen, is dit proefschrift een uitgebreide verkenning van de efficiëntie van diep leren, verspreid over het ingewikkelde tapijt van beeld- en videogegevens. De voorgestelde benaderingen hebben geleid tot een waardevolle vermindering van de reken- en gegevensvereisten. Het hier gepresenteerde werk heeft de normen verhoogd van wat bruikbare diepe modellen zouden moeten bereiken, en heeft het veld geholpen om in een betere, meer verantwoorde richting te gaan.

# 1 Introduction

Automatically processing visual data (image and video) has countless applications in every day life: from medical image analysis, to assisting systems and autonomous driving. For analyzing visual data, researchers have started from edge analysis [1–3] and grouping based on Gestalt principles [4–6] and extended this to probabilistic models [7, 8] and image descriptors [9–12]. However, in the past ten years a new set of algorithms [13–15] decisively showed great accuracy gains on visual data. These algorithms learn from example and are inspired by neural networks in the human brain and are generally referred to as: Deep learning. These deep networks allow vision researchers to move away from pre-defined visual cues, and learn to automatically extract the useful visual information directly from the visual input data. This thesis focuses exclusively on solving vision tasks with deep networks.

Deep networks have been used primarily for solving visual tasks such as: (1) *image recognition* [16–18] – recognizing the category of the objects present in an image, out of a predefined set of categories, called *annotations* or *labels*: *e.g.* 'car', 'person', 'plant', *etc.*; (2) *object recognition* [19–21] – localizing objects in images or frames of a video, typically by drawing a bounding-box around the objects, and recognizing the category of the object; (3) *action recognition* [13, 22–24] – recognizing the action present in a video out of a set of predefined action categories: *e.g.* 'jumping', 'running', 'swimming', *etc.* This thesis discusses image recognition in Chapter 4 and Chapter 5, object recognition on video-data in Chapter 2, and action recognition in Chapter 3, where I focus both on network training strategies and network architectures. I shortly introduce below the typical deep network building blocks (network architecture), and the deep learning training procedure.

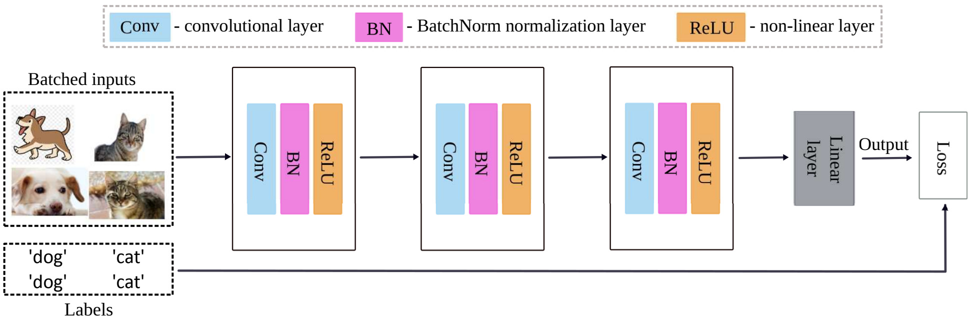


Fig. 1.1: An example of typical building blocks of a deep network architecture. The network starts from batches of inputs, and computes convolutions (in *convolutional layers*) or linear functions of the inputs (in the *linear layers*). The linear/convolutional layers contain the learnable parameters. These layers are typically followed by *normalization layers* such as BatchNorm [25] and *non-linear layers* such as ReLU [26].

A typical convolutional deep network architecture starts from input samples (here, images or video sequences), split into groups of a fixed size, called *batches*. The network is composed of multiple layers, as in Figure 1.1. The layers compute convolutions over the layer inputs (*convolutional layer*), or compute a linear function of the layer inputs (*linear layer*). The convolutional/linear layers are typically followed by layers that compute input statistics, knows as *normalization layers* (such as BatchNorm [25], LayerNorm [27], InstanceNorm [28], and GroupNorm [29]). And the normalization layers are followed by non-linear functions (*activation functions/non-linear layers*), of which the most popular is the ReLU [26]. The convolutional layers and linear layers contain learnable parameters, also called *weights*. The parameters are fitted to the training data, and the more layers, the larger the number of parameters. The output of each layer is called an *activation* or *feature map*. The network predicts an output which, then, is optimized with respect to the labels, by minimizing a function called the *loss*. In this thesis, I analyze the normalization layers in Chapter 5.

With regard to deep network training, this involves optimizing the deep network parameters so that they fit well to the training data, after which such a trained network is evaluated on unseen test data. The training data consists of examples images or videos and associated labels. When not every training image has an associated label, the annotations are called *sparse*. This is generally the case with data for real-world applications, and this thesis analyzes the sparse data annotations in Chapter 2. Additionally, when the training data is recorded in a different setting than the test data (*e.g.* day-time and night-time), then the training setting is called the *source domain* and the test setting is called the *target domain*. In Chapter 4 I focus on deep learning methods that can extract information useful for both source and target domain (called *cross-domain*), and can adapt the source domain to the target domain (known as *domain adaptation*). The deep learning training procedure is composed of 2 steps called *passes*: the *forward pass* computes the results of all the network operations starting from the inputs and going towards the outputs; the *backward pass* computes the gradients of the loss function with respect to the parameters of the network and updates the network parameters. In Chapter 3 I focus on speeding up the training procedure, and rethink the complete forward and backward pass as well as how activations are computed.

### 1.0.1. Efficiency in deep learning

Deep networks achieve impressive results on visual problems [16, 17, 31, 32]. However, these results come at a great computational cost, leading to great energy consumption being associated with deep learning. Deep learning models have become increasingly complex, with growing numbers of layers, requiring substantial computational resources when fitted to the training data (training) and when used to make predictions on unseen testing data (inference), as shown in Figure 1.2. The training of large-scale models often involves running computations on high-performance hardware, such as graphics processing units (GPUs) or specialized accelerators. These hardware devices consume significant amounts of power, which contribute to a high carbon footprint [33]. Additionally, the increasing demand for large-scale training datasets and cloud-based infrastructure further adds to the energy consumption and environmental impact. To address the efficiency and energy consumption concerns, researchers are exploring new techniques such as removing (pruning) redundant network parameters [34–36], or quantizing the network parameters [37–39], or devising efficient training strategies [40–42]. Given its urgency, this thesis also focuses on efficient deep learning and specifically on the efficiency of deep models for image and video data. Below I review the current efficient deep learning approaches for image and video data.

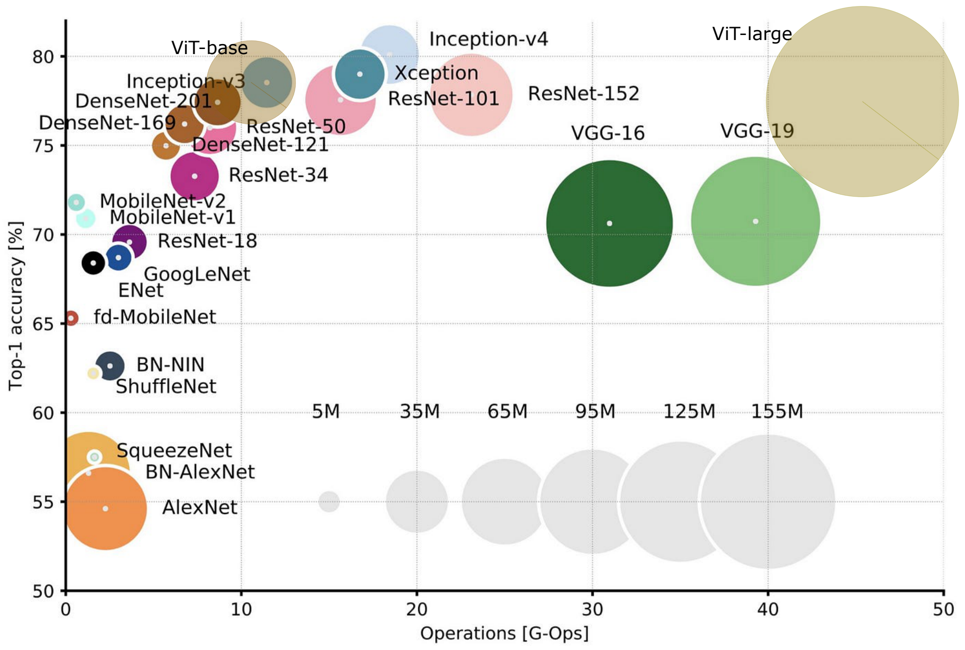


Fig. 1.2: Top-1 accuracy v.s. model sizes defined by the number of operations on the standard ImageNet dataset [30] for deep visual networks over the years 2014-2023. Deep learning models have become increasingly complex, needing more computational resources both when fitted to the data (training) and when making predictions (inference) to reach a high accuracy.

Deep learning efficiency in images is primarily done by network pruning [34–36] and parameter low-rank factorization [43–45]. These methods aim to reduce the number of parameters of deep neural networks by eliminating redundant parameters and exploiting sparsity. Lightweight architectures, such as MobileNet [46] and EfficientNet [47], have been designed to strike a balance between model size (number of parameters) and accuracy, making them suitable for deployment on mobile and edge devices. Knowledge distillation [48, 49] is another effective approach where a large model (the teacher) is trained to transfer its activations to a smaller, more efficient model (the student) by exploiting the teacher's predictions during training. Additionally, domain adaptation [50–52] techniques facilitate the efficient use of pre-trained models by adapting the knowledge learned from a source domain to a target domain, and thus reducing the training data requirements. Furthermore, methods for learning sample statistics (e.g. mean and variance), such as GroupNorm [29], have been proposed to enhance model efficiency by decoupling the model's behavior from batch statistics, allowing for faster training. In the context of deep learning efficiency on image data, this thesis focuses on domain adaptation in Chapter 4 and normalization layers that are efficient by being sample-independent in Chapter 5.

While efficiency is well explored for images, it is not as well investigated for videos.

Videos typically contain around 20-30 images per second, thus making deep models for videos computationally intensive. Architectural efficiency plays a crucial role in deep models for videos, where 3D convolutional networks, like C3D [53], extend traditional 2D convolutions with an extra dimension: time. 3D convolutional networks model the time dimension but increase the computation cost and introduce extra parameters. To make 3D convolutions efficient shifting activations over the temporal dimension in TSM (temporal shift module) [54] is extremely efficient. Recent advances in efficient video architectures, such as SlowFast [55], propose to process the videos on a fast path (where the video frames are greatly subsampled) and a slow path (processing dense frames). In terms of training strategy, augmenting the training data with different transformations, such as RandAugment [56], provide efficient yet diverse data augmentations that boost model generalization. Also, frame sampling strategies, like Temporal Segment Network (TSN) [24], efficiently sample video segments during training, striking a balance between computational efficiency and model performance. Building on these successful prior works, in this thesis, I propose two training strategies to address the training efficiency for video object detection Chapter 2 and video action recognition in Chapter 3.

The following sub-sections start by giving a brief introduction of each task in each of the individual thesis chapters. Finally, I conclude with the contributions of the thesis.

## 1.1. EFFICIENCY IN VIDEOS

Here, I briefly introduce this thesis's focus on building and training efficient deep learning models for video data. For this, I focus on two video understanding tasks: video object detection, and video action recognition.

### 1.1.1. CHAPTER 2: EFFICIENT VIDEO OBJECT DETECTION

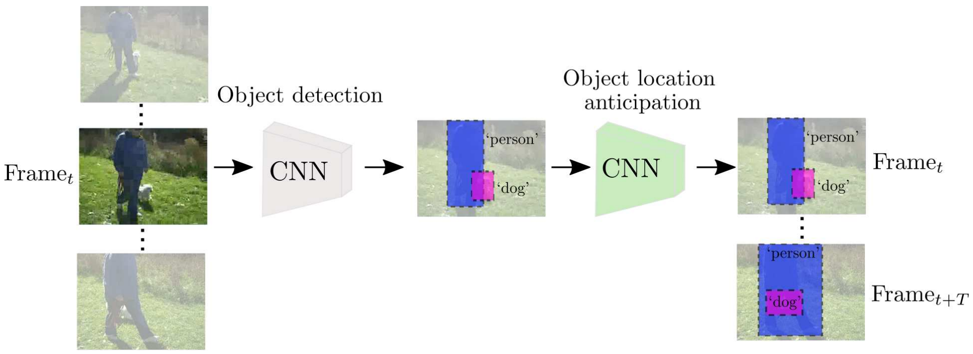


Fig. 1.3: **Chapter 2: Efficient video object detection.** The proposed video object detection method starts from individual video frames at timestep $t$ and detects the object locations via a convolutional neural network (CNN). Subsequently, the novelty of the method comes from anticipating the future object locations (highlighted in green, here) over the next $T$ frames. Thus, the frames between timestep $t$ and $t+T$ do not need to be processed in the network, leading to efficiency.

Video object detection involves identifying and localizing objects of interest within a sequence of video frames. Unlike image object detection, where objects are detected in individual images, video object detection focuses on detecting objects across consecutive frames in a video, as shown in Figure 1.3. Previous works on video object detection either perform temporal post-processing of single-frame object detectors [57–60] to align the object detection results from each frame along the temporal dimension, or aggregate temporal information in different ways [61–65] through feature maps of sampled frames from a video. These post-processing steps and featuremaps aggregations make these methods computationally expensive. In contrast, I propose a method in Chapter 2 to start from individual frames sampled at different timesteps, $t$, and detect the objects present in these frames and then subsequently anticipate the detected object locations over the subsequent $T$ frames, as shown in Figure 1.3. By anticipating future object locations over $T$ frames from a single static frames at time $t$, the network does not need to process all the frames between $t$ and $t+T$, leading to efficiency.

### 1.1.2. Chapter 3: Efficient Video Action Recognition

Action recognition entails identifying and classifying human actions or activities from a sequence of video frames, as illustrated in Figure 1.4 This task is challenging due to the ambiguity of describing activities at the appropriate timescales [66]. Even though many actions can be recognized without reasoning about the long-term temporal relations, such as video actions in datasets such as UCF101 [67] and THUMOS [68], deep neural networks still struggle in situations where data and observations are limited, or where the underlying structure is characterized by long-term temporal relations, rather than the appearance of certain entities [69, 70]. To represent the long-term temporal relations of a full video, prior methods train by using sub-sampled video frames [13, 24, 71, 72]. Sub-sampling is needed because realistic videos contain thousands of frames and these cannot be fit in the GPU memory. Furthermore, instead of computationally-intensive representations learned from 3D convolutions or optical flow, recent video action recognition methods [13, 22–24, 42, 71–73] rely on efficient 2D CNNs. However, even with 2D CNNs, the number of input sub-sampled frames that can be fit in the memory is only 16 to 64 frames, due to the large networks that also occupy GPU memory. Furthermore, since not all video frames are equally informative for recognizing an action, sub-sampling can miss crucial frames for video action recognition.

To address the issues above, I propose to do away with sub-sampling heuristics and argue for leveraging all video frames: use all video frames during training, as shown in Figure 1.4. My method overcomes the memory limitation by reconsidering the forward pass and backward pass computations. The method starts by computing frame activations ($f_{maps}$ in Figure 1.4) for all frames, and then aggregates the activations over frames such that fewer gradients are computed. If the network would be linear, a huge memory reduction could be gained by first summing all frame activations in the forward pass, which would reduce to having just a single frame-gradient in the backward pass. Yet, deep networks are non-linear and have non-linearities in the activation function and in the loss function. The differences between the linear assumption and the actual non-linearity of the network introduce gradient updating errors. I propose to reduce these errors by aggregating the approximately linear parts of the activations along the

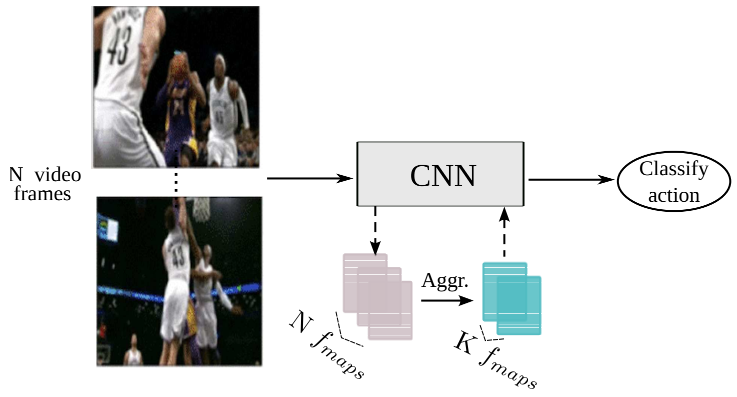


Fig. 1.4: Chapter 3: **Efficient video action recognition.** The proposed method aggregates the feature maps ($f_{maps}$) of the input $N$ frames and reduces the number of feature maps from $N$ to $K$ (where $K \ll N$) in the forward pass. It largely saves computation memory for video action recognition and makes it possible to input more frames from a video.

temporal dimension (video frames), leading to memory savings in the backward pass while still approximating the full video gradient.

## 1.2. EFFICIENCY IN IMAGES

My work on the efficiency on image data focuses on domain adaptation with limited annotations, and on learning a normalization layer that is independent of data sample statistics. I briefly discuss these below.

### 1.2.1. CHAPTER 4: DATA EFFICIENT CROSS DOMAIN IMAGE MATCHING

Cross domain image matching implies matching two images that are collected in different settings (*i.e.* domains): for instance, photos of the same building captured at day-time and at night-time. Here I focus on the challenging setting where the source domain is labeled while the target domain data is unlabeled, thus paired-image examples from the two domains are not available during training. Therefore, I consider both image matching and domain adaptation techniques for solving this task. For image matching, prior work uses a special type of network where the weights are shared between 2 branches, called 'siamese networks' [74]. In Chapter 4, I also adopt the siamese network architecture as part of my framework. To match the unlabeled target domain images with the source domain images, I employ domain adaptation. Domain adaptation has been widely researched in diverse classification tasks, and the empirical multi-kernel maximum mean discrepancy (MK-MMD) [75] is the most used metric, which is a statistical method for comparing and measuring the similarity between two sets of data. In my proposed approach I also use this metric to assess the similarity of the source domain and the target domain. Additionally in Chapter 4 I address another challenge:

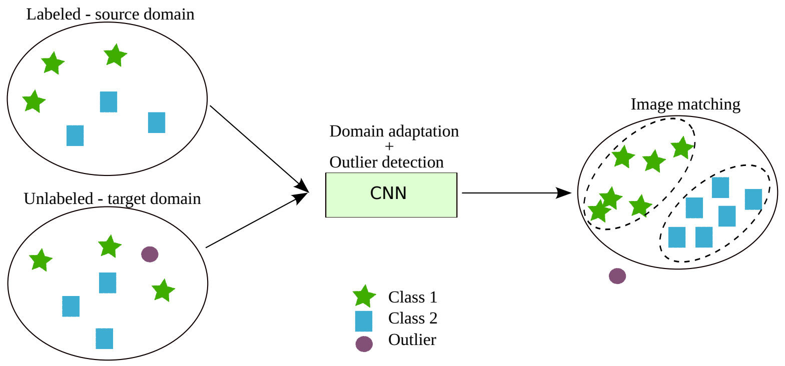


Fig. 1.5: Chapter 4: **Data efficient cross-domain image matching.** The proposed method learns domain invariant and representations for matching the images in the target domain with images in the source domain. By doing this it efficiently uses the source and target domain data without extra annotation needed in the target domain.

the image samples in the two domains may not fully overlap due to the existence of outlier images, as shown by the example in Figure 1.5. If left undetected, the outliers may affect the matching performance. Putting everything together, I propose a network using three constraints incorporating solutions for the three challenges mentioned above. The proposed method efficiently uses the source and target domain data without extra effort in labeling the target domain data.

### 1.2.2. CHAPTER 5: DATA EFFICIENT ACTIVATION NORMALIZATION INDEPENDENT OF SAMPLE STATISTICS

Normalization layers in deep networks are designed to stabilize the training process by avoiding gradients that have too large or too small magnitudes. They do this by normalizing the input activations within the layers. The most popular normalization layer is Batch Normalization [25] (BatchNorm). BatchNorm computes featuremap statistics (mean and standard deviation) across the sample dimension, $N$, and then normalizes the feature maps with these statistics. Thus, the performance of BatchNorm greatly depends on the number of input samples. Using large batch sizes is not feasible in applications such as object detection, segmentation, and video recognition, where just one or a few samples can be fit in the GPU memory due to the high-resolution/large number of frames. Using only a few or just one sample per batch is possible if the normalization is independent of the sample statistics. Various normalization techniques have been proposed to overcome the dependency on large batches, such as Layer Normalization [27], Instance Normalization [28], and Group Normalization [29]. These methods are applied per sample and are efficient, but they are less reliable than BatchNorm. Thus, there is a need for a method that is both reliable and efficient.

In Chapter 5, I propose WeightAlign, as shown in Figure 1.6, which normalizes activations without using sample statistics. This chapter proposes re-parameterizing the weights within a filter to arrive at correctly normalized activations. The proposed method

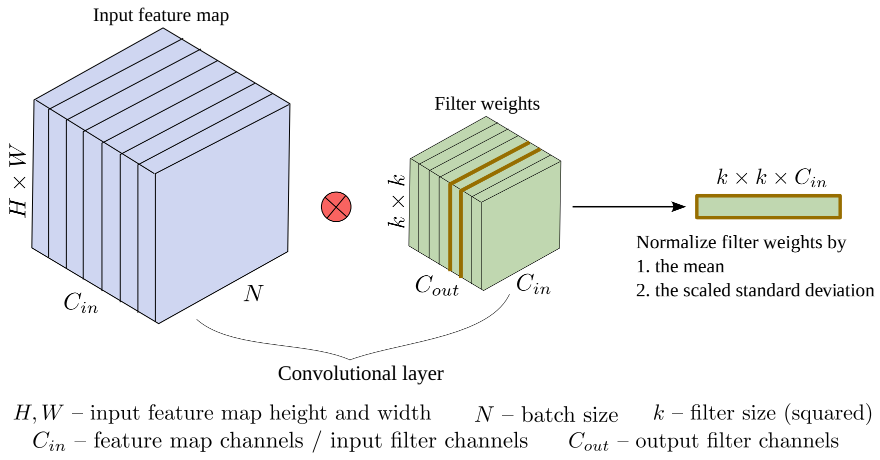

Input feature map
$H \times W$
$C_{in}$
$N$
Filter weights
$k \times k$
$C_{out}$
$C_{in}$
$k \times k \times C_{in}$
Normalize filter weights by
1. the mean
2. the scaled standard deviation
Convolutional layer
$H, W$ – input feature map height and width
$N$ – batch size
$k$ – filter size (squared)
$C_{in}$ – feature map channels / input filter channels
$C_{out}$ – output filter channels

Fig. 1.6: **Chapter 5: Data efficient activation normalization independent of sample statistics.** The proposed WeightAlign arrives at correct featuremap normalization by normalizing the weights by the mean and scaled standard deviation computed within a filter. WeightAlign achieves the same purpose as normalizing the featuremap but does not rely on the featuremap. WeightAlign is independent of the sample statistics because it only normalizes the filter weights, which leads to data efficiency.

is independent of batch size, and achieves stable performance over a wide range of tasks with various batch sizes. Because the proposed method is independent of the sample statistics, it is data efficient.

## 1.3. CONTRIBUTION

The main contribution of this thesis is developing efficient deep learning models for video understanding tasks, and using image data in an efficient way towards the tasks. I propose novel methods for efficient video understanding: *i.e.* memory-efficient video action recognition, and computationally efficient video object detection. And I explore image data efficiency with a cross-domain image matching task on unlabeled target domain data, and sample-statistics independent activation normalization layers.

Chapter 2 exploits the continuous smooth motion of objects in videos for video object detection in two ways: 1) Improved accuracy by exploiting object motion as an additional source of supervision, which is exploited by anticipating motion from a static keyframe. And 2) Improved efficiency by only performing the expensive feature computations on a small subset of all frames. Because neighboring video frames are often redundant, features can be computed only for a single static keyframe, and the object positions can be predicted in subsequent frames. In addition, this work also explore a sparse annotation setting here, where only keyframe annotations are used and between keyframes the model relies on smooth pseudo-motion for training. The proposed model demonstrates computational efficiency, annotation efficiency, and improved mean average precision when compared to state-of-the-art methods.

Chapter 3 proposes an efficient method to use all video frames during training for video action recognition. Assuming the network would be linear, then a huge memory reduction is gained by first summing all frame activations in the forward pass. This reduces to just a single update in the backward pass. Yet, deep networks have non-linearities in the activation function and in the loss function. Thus, treating the non-linear network as linear would introduce considerable approximation errors. However, subsequent frames in a video are strongly correlated, and it's this correlation that makes it possible for the proposed method to process all frames and exploit the frame correlations to create groups of frames, where the network is approximately linear. By aggregating the approximately linear parts in a video in the forward pass, the method achieves large memory savings while still approximating the full video gradient.

Chapter 4 addresses the problem of domain adaptation for feature learning in a cross domain matching task when outliers are present. As is common in domain adaptation, only labeled image pairs from the source domain are available, but no labels from the target domain. To resolve the domain disparity between the train and the test data, the model relies on Siamese network [74] for image matching and domain adaptation used in image classification [76–80]. I propose a triplet constraints network to learn the domain invariant and identity distinguishable representations of the samples. This is made possible by utilizing the paired-image information from the source domain, a weighted multi-kernel maximum mean discrepancy (weighted MK-MMD) method and an entropy loss. Given the lack of publicly available datasets for the problem setting, this work introduces two new synthetic datasets. The proposed method gives a practical solution for the cross domain image matching task with outliers present.

Chapter 5 studies the normalization of the network activation. Unlike BatchNorm, whose performance critically depends on the quality of the sample statistics, this work proposes WeightAlign: normalizing activations without using sample statistics. Instead of relying on sample statistics, the model re-parameterizes the weights within a filter to arrive at correctly normalized activations. The proposed method is based on weight statistics and is thus orthogonal to sample statistics. This allows it to exploit two orthogonal sources to normalize activations: the normalization based on sample statistics in combination with the proposed new one based on weight statistics. The proposed WeightAlign is data efficient because it is a sample statistics independent normalization. WeightAlign also achieves stable performance over a wide range of tasks with various batch sizes.

# REFERENCES

[1] L. S. Davis. “A survey of edge detection techniques”. In: *Computer graphics and image processing* 4.3 (1975), pp. 248–270.

[2] D. Marr and E. Hildreth. “Theory of edge detection”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 207.1167 (1980), pp. 187–217.

[3] V. Torre and T. A. Poggio. “On edge detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1986), pp. 147–163.

[4] A. Desolneux, L. Moisan, and J.-M. Morel. “Gestalt theory and computer vision”. In: *Seeing, thinking and knowing: Meaning and self-organisation in visual cognition and thought*. Springer, 2004, pp. 71–101.

[5] J. W. Jaromczyk and G. T. Toussaint. “Relative neighborhood graphs and their relatives”. In: *Proceedings of the IEEE* 80.9 (1992), pp. 1502–1517.

[6] D. Marr. “Early processing of visual information”. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 275.942 (1976), pp. 483–519.

[7] B. Moghaddam and A. Pentland. “Probabilistic visual learning for object representation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), pp. 696–710.

[8] A. R. Pope and D. G. Lowe. “Probabilistic models of appearance for 3-D object recognition”. In: *International Journal of Computer Vision* 40 (2000), pp. 149–167.

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-up robust features (SURF)”. In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[10] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. Ieee. 2005, pp. 886–893.

[11] D. G. Lowe. “Local feature view clustering for 3D object recognition”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE. 2001, pp. I–I.

[12] D. G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60 (2004), pp. 91–110.

[13] J. Carreira and A. Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. 2012, pp. 1097–1105.

[15] S. Ren, K. He, R. Girshick, and J. Sun. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. “Densely connected convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[18] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *ICLR* (2015).

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik. “Region-based convolutional networks for accurate object detection and segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2015), pp. 142–158.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. “Large-scale video classification with convolutional neural networks”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[23] K. Simonyan and A. Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems* 27 (2014).

[24] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. “Temporal segment networks: Towards good practices for deep action recognition”. In: *European conference on computer vision*. Springer. 2016, pp. 20–36.

[25] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[26] A. F. Agarap. “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375* (2018).

[27] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[28] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

[29] Y. Wu and K. He. “Group normalization”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, pp. 248–255.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR* (2021).

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[33] E. Strubell, A. Ganesh, and A. McCallum. “Energy and policy considerations for modern deep learning research”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 09. 2020, pp. 13693–13696.

[34] Y. Chen, Z. Ma, W. Fang, X. Zheng, Z. Yu, and Y. Tian. “A Unified Framework for Soft Threshold Pruning”. In: *arXiv preprint arXiv:2302.13019* (2023).

[35] E. Frantar and D. Alistarh. “Optimal brain compression: A framework for accurate post-training quantization and pruning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 4475–4488.

[36] A. Peste, E. Iofinova, A. Vladu, and D. Alistarh. “Ac/dc: Alternating compressed/decompressed training of deep neural networks”. In: *Advances in neural information processing systems* 34 (2021), pp. 8557–8570.

[37] H. Qin, M. Zhang, Y. Ding, A. Li, Z. Cai, Z. Liu, F. Yu, and X. Liu. “Bibench: Benchmarking and analyzing network binarization”. In: *arXiv preprint arXiv:2301.11233* (2023).

[38] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. “Gptq: Accurate post-training quantization for generative pre-trained transformers”. In: *arXiv preprint arXiv:2210.17323* (2022).

[39] J. Martinez, J. Shewakramani, T. W. Liu, I. A. Bârsan, W. Zeng, and R. Urtasun. “Permute, quantize, and fine-tune: Efficient compression of neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15699–15708.

[40] J. Launay, I. Poli, F. Boniface, and F. Krzakala. “Direct feedback alignment scales to modern deep learning tasks and architectures”. In: *Advances in neural information processing systems* 33 (2020), pp. 9346–9360.

[41] H. Cai, C. Gan, L. Zhu, and S. Han. “Tinytl: Reduce memory, not parameters for efficient on-device learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11285–11297.

[42] Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, and Z. Wang. “E2-train: Training state-of-the-art cnns with over 80% energy savings”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[43] S. Han, H. Mao, and W. J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *arXiv preprint arXiv:1510.00149* (2015).

[44] Y. Idelbayev and M. A. Carreira-Perpinán. "Low-rank compression of neural nets: Learning the rank of each layer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8049–8059.

[45] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. Li, and Y. Chen. "Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 678–679.

[46] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[47] M. Tan and Q. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[48] P. Chen, S. Liu, H. Zhao, and J. Jia. "Distilling knowledge via knowledge review". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5008–5017.

[49] G. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[50] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.

[51] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann. "Contrastive adaptation network for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4893–4902.

[52] Y. Zhang, T. Liu, M. Long, and M. Jordan. "Bridging theory and algorithm for domain adaptation". In: *International conference on machine learning*. PMLR. 2019, pp. 7404–7413.

[53] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[54] J. Lin, C. Gan, K. Wang, and S. Han. "TSM: Temporal shift module for efficient and scalable video understanding on edge devices". In: *IEEE transactions on pattern analysis and machine intelligence* 44.5 (2020), pp. 2760–2774.

[55] C. Feichtenhofer, H. Fan, J. Malik, and K. He. "Slowfast networks for video recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2019, pp. 6202–6211.

[56] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.

[57] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. “Seq-nms for video object detection”. In: *CoRR* (2016).

[58] K. Kang, W. Ouyang, H. Li, and X. Wang. “Object detection from video tubelets with convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 817–825.

[59] G. Bertasius and L. Torresani. “Classifying, segmenting, and tracking object instances in video with mask propagation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9739–9748.

[60] C.-C. Lin, Y. Hung, R. Feris, and L. He. “Video instance segmentation tracking with a modified vae architecture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13147–13157.

[61] H. Wu, Y. Chen, N. Wang, and Z. Zhang. “Sequence level semantics aggregation for video object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9217–9225.

[62] Y. Chen, Y. Cao, H. Hu, and L. Wang. “Memory enhanced global-local aggregation for video object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10337–10346.

[63] Y. Cui, L. Yan, Z. Cao, and D. Liu. “TF-Blender: Temporal Feature Blender for Video Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8138–8147.

[64] M. Han, Y. Wang, X. Chang, and Y. Qiao. “Mining inter-video proposal relations for video object detection”. In: *European conference on computer vision*. Springer. 2020, pp. 431–446.

[65] Z. Jiang, P. Gao, C. Guo, Q. Zhang, S. Xiang, and C. Pan. “Video object detection with locally-weighted deformable neighbors”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8529–8536.

[66] G. A. Sigurdsson, O. Russakovsky, and A. Gupta. “What actions are needed for understanding human actions in videos?” In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2137–2146.

[67] K. Soomro, A. R. Zamir, and M. Shah. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *arXiv preprint arXiv:1212.0402* (2012).

[68] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. “The thumos challenge on action recognition for videos “in the wild””. In: *Computer Vision and Image Understanding* 155 (2017), pp. 1–23.

[69] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. “A simple neural network module for relational reasoning”. In: *Advances in neural information processing systems* 30 (2017).

[70] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. “Building machines that learn and think like people”. In: *Behavioral and brain sciences* 40 (2017), e253.

[71] J. Lin, C. Gan, and S. Han. "Tsm: Temporal shift module for efficient video understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7083–7093.

[72] M. Zolfaghari, K. Singh, and T. Brox. "Eco: Efficient convolutional network for online video understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

[73] M. Jain, J. C. Van Gemert, and C. G. Snoek. "What do 15,000 object categories tell us about classifying and localizing actions?" In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 46–55.

[74] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE. 2005, pp. 539–546.

[75] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. "Optimal kernel choice for large-scale two-sample tests". In: *Advances in neural information processing systems* 25 (2012).

[76] M. Long, Y. Cao, J. Wang, and M. Jordan. "Learning transferable features with deep adaptation networks". In: *International conference on machine learning*. PMLR. 2015, pp. 97–105.

[77] K. Saito, Y. Ushiku, and T. Harada. "Asymmetric tri-training for unsupervised domain adaptation". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2988–2997.

[78] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[79] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. "Deep hashing network for unsupervised domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5018–5027.

[80] X. Zhang, F. X. Yu, S.-F. Chang, and S. Wang. "Deep transfer network: Unsupervised domain adaptation". In: *arXiv preprint arXiv:1503.00591* (2015).

# 2 Objects do not disappear: Video object detection by single-frame object location anticipation

*Objects in videos typically have continuous smooth motion. We exploit continuous smooth motion in two ways: 1) Improved accuracy by exploiting object motion as an additional source of supervision, which we can exploit by anticipating motion from a static keyframe. And 2) Improved efficiency by only doing the expensive feature computations on a small subset of all frames. Because neighboring video frames are often redundant we only compute features for a single static keyframe and predict object positions in subsequent frames. In addition, we also explore a sparse annotation setting, where we only annotate the keyframe and use smooth pseudo-motion between keyframes. We demonstrate computational efficiency, annotation efficiency, and improved mean average precision when compared to state-of-the-art methods on three common datasets: ImageNet VID, EPIC KITCHENS-55, and YouTube-BoundingBoxes. We will make our code publicly available.*

This chapter is published as: **Xin Liu**, Fatemeh Karimi Nejadasl, Jan C Van Gemert, Olaf Booij, Silvia L Pintea. *Objects do not disappear: Video object detection by single-frame object location anticipation*. International Conference on Computer Vision (ICCV), 2023. arXiv: `https://arxiv.org/abs/2308.04770`

## 2.1. INTRODUCTION

Humans assume object permanence; blink your eyes: and the world is still there. Similarly, video frames are redundant, and missing some frames when watching a movie does not drastically change the scene. Actually, for the parts that *did* change, if these parts changed coherently, they might hint at a sense of objectness, as hypothesized by the Gestalt law of common fate. As illustrated in Figure 2.1 we here explore these observations in the context of video object detection in two ways. 1) Improved accuracy by the law of common fate: By predicting object motion from a static image, we exploit an additional source of supervision which improves the video object detector accuracy because coherent motion hints at objectness. 2) Improved efficiency by exploiting redundancy to reduce computational cost and to reduce annotation cost. Instead of processing all frames in a video, we sample keyframes and predict object motion for the missing frames in-between; saving much computation and annotation time by simply skipping the feature computation and/or the annotation for a large majority of the frames.

We make the following contributions: (i) We propose a video object detection method that samples static keyframes and predicts object motion for unseen future frames. (ii) We are compute efficient, as we only extract features for sampled static keyframes; (iii) We demonstrate data efficiency, as we allow sparse annotations for only the sampled keyframes, hallucinating motion in-between sparse annotations. (iv) Our experimental results on ImageNet-VID [1], EPIC KITCHENS-55 [2], and YouTube-BoundingBoxes [3] show that our approach improves accuracy over the current state of the art video object detection methods, while being faster at both training and inference time.

## 2.2. RELATED WORK

**Video object detection.** Several methods perform temporal post-processing of still-image object detectors such as post-processing Faster-RCNN [4, 5] or post-processing Mask R-CNN [6, 7]. Alternatives include recurrent blocks [8, 9] or optical flow [10, 11]. This can be further extended with instance and pixel-level calibrations over time [12], or using a space-time lattice [13]. More recently, the field has achieved notable momentum by aggregating temporal information: either by defining detection correlations as a graph in SELSA [14], or by using global and local temporal pooling in MEGA [15]. MEGA is further extended by considering all pairwise frames in TF-Blender [16], while HVRNet [17] integrates inter-video and intra-video object proposal relations. IFF-Net[18] uses a feature flow estimating module to indicate the feature displacement for video object detection. LWDN [19] does not simply aggregate features, but aligns the features between keyframes by adopting a memory mechanism. In our paper, in contrast, we do not aggregate neighboring frames, nor any other temporal heuristics to post-process still-image detections. We avoid computationally demanding optical flow and recurrent blocks. Instead, we anticipate future object locations over time from a single static keyframe, placing temporal prediction at the heart of our model.

**Single image motion prediction versus tracking.** Object tracking aims to predict object location in a video. In classical methods, the bounding box around the object to track is given a priori [20], but can also be estimated by an object detector [21–24]. The

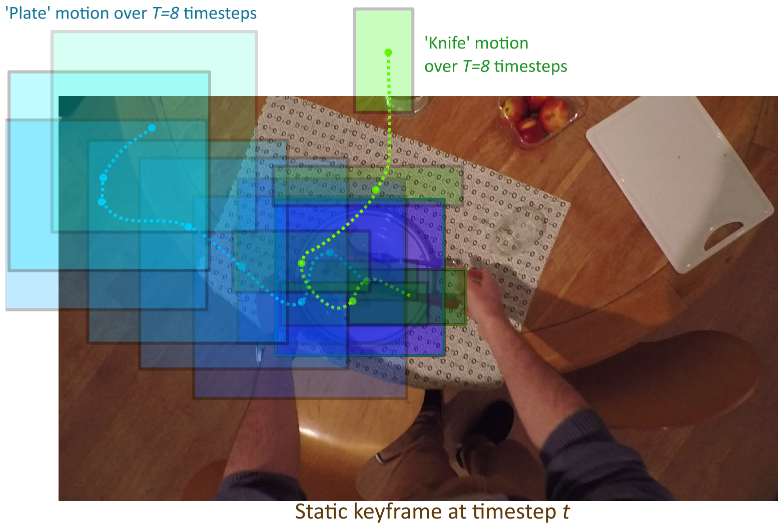

'Plate' motion over *T=8* timesteps
'Knife' motion
over *T=8* timesteps
Static keyframe at timestep *t*

Fig. 2.1: Anticipating future object locations from a static image is efficient and exploits motion cues as an additional form of supervision for video object detection. By sampling a single static keyframe at time $t$ and anticipating the object locations over the following $T$ timesteps, we incorporate temporal consistency and smoothness of object motion. Moreover, we are efficient because we only do the computationally expensive feature extraction on a small subset of keyframes, while still using bounding box locations for all video frames.

object location in the next frame can be estimated with siamese networks [21], or object detections at every frame are linked into tracks by motion regression [22]. And a recent overview of object tracking is given by [25]. Our method is inspired by tracking, yet we are different because we make predictions of the object location in future frames from a single static input image, without actually using the image features of the future frame.

**Anticipating motion from a static single image.** A single static frame is rich enough to allow predicting of future appearance [26–29], actions [30–33], and motion [34–36]. Such motion predictions, in turn, can then be used for predicting pedestrians' behaviors [37, 38] or other road agents [39]. Moreover, motion prediction can be used as a self-supervision cue [40–44] to improve feature learning. Inspired by these works, we also predict the future from a still-image: future object locations. We use motion prediction as an additional source of supervision to improve the accuracy of the still-image detector, while we also exploit it for efficiency as it allows us to only compute features on a subset of still-images, avoiding expensive feature computations on all frames.

**Efficiency in video.** Because videos typically sample several frames per second (FPS) it is important to have efficient video analysis methods. Successful prior work proposes network architecture adaptations to reduce computations [45–49]. For video object detection, it is efficient to adapt the detector online [50] or to transfer an image detector to video [51]. Alternatively, computations can be reduced by focusing only on specific video regions [52–54]. Similarly, we also focus on efficiency in video object detection. We are efficient by predicting object trajectories a few frames ahead and therefore saving computations by not processing those frames.

**Sparse annotations in video.** Training a model on sparse video annotations can be done by iteratively updating the model in a boosting fashion [55], or propagating groups of pixels through time [56]. More recent work generates dense object masks in videos from sparse bounding boxes [57]. Rather than focusing on improving the model accuracy on sparsely annotated videos, prior work has analyzed what is the accuracy vs. annotation effort trade-off [58]. Specifically for object locations in video, annotations can be generated through a combination of tracking, frame selection and active learning [59–62]. Inspired by these works we also explore a variant of our model that allows sparse annotations. We require bounding box annotations on the static keyframes, and we will experimentally show that we can hallucinate the motion between keyframes, removing the need to annotate all frames.

## 2.3. ANTICIPATING OBJECT LOCATIONS

**Object detection backbone.** We illustrate our method in Figure 2.2. We start from a standard static image object detection backbone, and input static frames that uniformly sub-sampled from the video over time with a time step $T$, we call such static frames "keyframes". For each keyframe at time $t$, the object detector detects a set of $N$ bounding boxes $B_t = \{B_t^i\}_{i \in N}$ where $B_t^i = (x_t^i, y_t^i, w_t^i, h_t^i)$, $i \in \{1,..N\}$ and $x_t^i, y_t^i$ are the top-left corner and $w_t^i, h_t^i$ are the width and height of the bounding box. Each $B_t^i$ corresponds to a possible object and class $c_t^i$: $\{(B_t^i, c_t^i)\}_{i \in N}$.

**Trajectory subnetwork.** Starting from the keyframe detection bounding boxes $B_t$, we anticipate object trajectories – defined as the future bounding box locations of an object over the following $T$ frames. The trajectory subnetwork is highlighted in Figure 2.2. For each keyframe indexed by $t$, we define a batch of length $T$ and we input into the trajectory network three types of inputs: (a) the time index of the future trajectory location relative to the keyframe index $t$ batched from $[1,..,T]$; (b) the set of bounding boxes detected over the keyframe $B_t = \{B_t^i\}_{i \in N}$ repeated over all $T$ time steps: $[\{B_t^i\}_{i \in N}] \times T$; and (c) the static image feature maps extracted from the area of the keyframe bounding boxes $\{f_t^i \mid f_t^i = f(B_t^i)\}_{i \in N}$, also broadcast for each of the $T$ time steps: $[\{f_t^i\}_{i \in N}] \times T$. Note that by inputting into the trajectory subnetwork the future trajectory-frame indices in (a), we add temporal ordering information: each future box prediction has an associated frame index. We map all three inputs: boxes, features and time indices of trajectory length, through fully connected (FC) layers of equal size. We concatenate the output features, and pass them through two additional fully connected layers.

The output of the trajectory subnetwork is a list of trajectories, one trajectory for each N detected objects indexed by $i$. Each trajectory starts at the keyframe at time $t$ and extends over the future $t+T$ frames. Concretely, our trajectory predictions are: $\{\mathbb{T}_t^i \mid \mathbb{T}_t^i = (B_t^i, B_{t+1}^i, .., B_{t+T}^i)\}_{i \in N}$, where we also add the keyframe bounding box $B_t$ to the trajectory.

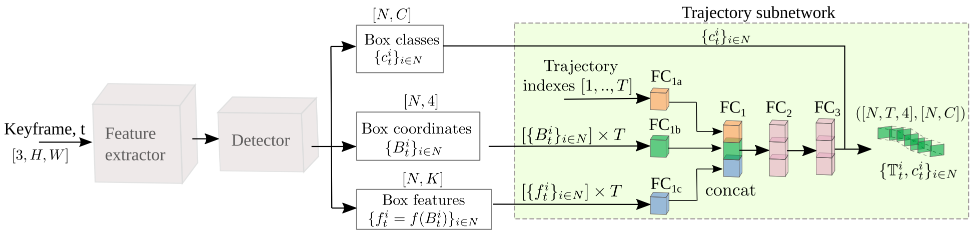


Fig. 2.2: **Overview:** The network starts from sampled video keyframes at timestep $t$. A feature extractor backbone is followed by an object detector. The object detector outputs for the keyframe at time $t$ a set of $N$ bounding boxes $\{B_t^i\}_{i\in N}$, together with their associated class probabilities $\{c_t^i\}_{i\in N}$ and features extracted from each box area $\{f_t^i\}_{i\in N}$. Next, our trajectory subnetwork takes as input: (a) a batch of trajectory indexes $[1,..,T]$, where T is the trajectory length; (b) the keyframe bounding boxes repeated $T$ times and batched, $[\{B_t^i\}_{i\in N}]\times T$; and (c) the box features also repeated $T$ times $[\{f_t^i\}_{i\in N}]\times T$. These are projected through linear layers of equal size (FC$_{1a}$, FC$_{1b}$, FC$_{1c}$) and the output is concatenated and passed through two additional linear layers. The output of the trajectory network is a set of $N$ trajectories $\{\mathbb{T}_t^i\}_{i\in N}$ of length $T$ and their associated classes $\{c_t^i\}_{i\in N}$.

## 2.3.1. ASSOCIATING TRAJECTORIES TO GROUND TRUTH

To optimize our trajectory predictions, we need to associate ground truth boxes with all predicted boxes along the trajectory and an object class to each trajectory. Each trajectory starts with a keyframe bounding box $B_t^i$, which has a corresponding object class $c_t^i$. We assume that the object class remains the same over the next $T$ frames along each trajectory and thus let each trajectory inherit the class of its starting keyframe bounding box, yielding: $\{(\mathbb{T}_t^i, c_t^i)\}_{i\in N}$.

To define the box-regression loss we need to associate trajectories to ground truth bounding boxes, $B^*$. We consider a set of trajectory boxes : $\{\mathbb{T}_t^i = (B_t^i, B_{t+1}^i, .., B_{t+T}^i)\}_{i\in N}$. Following the standard procedure [63, 64] we rank the predicted boxes $B_{t+l}$ based on their overlap with the ground truth boxes $B_{t+l}^*$ at each frame $t+l$. We associate each of the $N$ predicted boxes with the best matching IoU score of the ground truth box.

## 2.3.2. TRAJECTORY LOSS

**Bag of boxes loss.** We want to optimize for each object indexed by $i$ its associated trajectory, starting at keyframe $t$: $\{\mathbb{T}_t^i\}_{i\in N}$. For readability, we ignore the index $i$ from here on. The standard loss $L_{\text{bag}}$ for performing box regression, considers the trajectory boxes as an unordered bag and computes a smooth $L_1$ loss $\mathscr{L}_1(\cdot)$ [64, 65], for each predicted box $B_{t+l}$, to its associated ground truth box $B_{t+l}^*$:

$$L_{\text{bag}}(B^*, \{B_t, .., B_{t+T}\}) = \sum_{l=0}^{T} \mathscr{L}_1(B_{t+l}^* - B_{t+l}). \tag{2.1}$$

**Trajectory cumulative loss.** The downside of Equation (3.2) is that it treats each prediction $B_{t+l}$ as if it were independent of its neighboring predictions along the

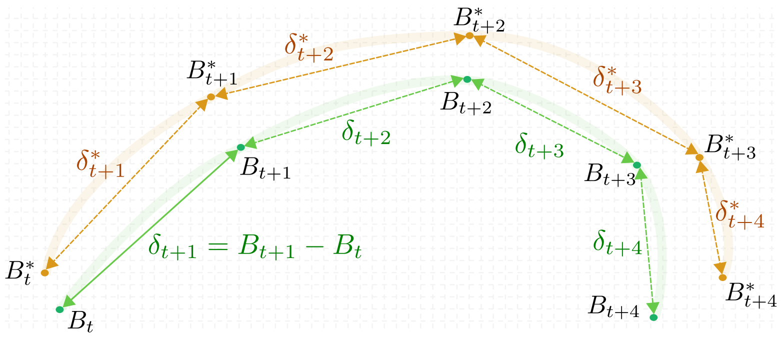


Fig. 2.3: Object trajectories are piecewise continuous: an object cannot disappear between two neighboring frames $t$ and $t+1$. We incorporate this continuity, by noting that the loss between a box $B_{t+l}$ along the trajectory and its associated ground truth $B^*_{t+l}$ is defined as the sum of pairwise offsets of neighboring boxes, $\delta_{t+l}$, starting from the keyframe box $B_t$. The orange line is the true trajectory and the green line is the predicted trajectory. (Here, for simplicity we discard the width and height of the bounding boxes and only show $(x, y)$ coordinates and $l \in \{1,..,4\}$.)

trajectory, $B_{t+l-1}$ and $B_{t+l+1}$. Therefore, there is no temporal ordering enforced in the $L_{\text{bag}}$ loss. Not enforcing the ordering of the predictions along the trajectory could lead to discontinuous trajectories. We want to enforce smoothness in the predictions over time: objects cannot disappear or appear at random locations, between neighboring frames. Specifically, the trajectory is piecewise continuous: for an object to move from a location $B_{t+1}$ to a location $B_{t+4}$ it has to travel through the intermediate locations $B_{t+2}$ and $B_{t+3}$, as illustrated in Figure 2.3. To add this insight, we define a loss that constrains the pairwise offsets along the trajectory $\delta_{t+k} = (B_{t+k} - B_{t+k-1}), k \in \{1,..,l\}$, from frame $t$ up to every frame $t+l$, to add up to the offset from the ground truth at the frame $t+l$ to the keyframe prediction $(B^*_{t+l} - B_t)$. Concretely:

$$L_{\Sigma}(B^*, \overleftrightarrow{\mathbb{T}}_t) = \sum_{l=0}^{T} \mathscr{L}_1 \left( (B^*_{t+l} - B_t) - \sum_{k=1}^{l} \delta_{t+k} \right), \tag{2.2}$$

where we redefine the trajectory to predict $\delta_{t+l}$ values describing pairwise offsets instead of bounding boxes: $\overleftrightarrow{\mathbb{T}}_t = (\delta_{t+1},..,\delta_{t+T})$. Note, that if we would predict bounding boxes $B_{t+l}$ in the trajectory network, instead of offsets between pairs of bounding boxes $\delta_{t+l}$, Equation (2.2) would reduce to Equation (3.2) and the temporal ordering would not be enforced, see the derivation in the supplementary material.

In Equation (2.2) the inner loop over pairwise offsets $\delta_{t+k}$ accumulates the errors in the predictions over time from frame $t+1$ to frame $t+l$. To make sure the errors do not accumulate along the trajectory, and the change from frame to frame is smooth, we add another loss $L_{\text{bag}(\delta)}$ constraining the pairwise offsets $\delta_{t+l}$ at every timestep $t+l$ to map back to ground truth offsets: $\delta^*_{t+l} = (B^*_{t+l} - B^*_{t+l-1})$:

$$L_{\text{bag}(\delta)}(B^*, \overleftrightarrow{\mathbb{T}}_t) = \sum_{l=1}^{T} \mathscr{L}_1 \left( \delta^*_{t+l} - \delta_{t+l} \right). \tag{2.3}$$

Our final loss $L_{\text{traj}}$ is then a combination of the two losses where the cumulative trajectory loss $L_{\sum}$ enforces piecewise continuity in the predictions and the bag of offsets loss $L_{\text{bag}(\delta)}$ discourages errors from accumulating along the trajectory and makes the trajectory smooth:

$$L_{\text{traj}} = L_{\sum} + L_{\text{bag}(\delta)}. \tag{2.4}$$

We investigate the effect of each loss term in the experimental section, together with the effect of predicting offsets $\delta_{t+l}$ instead of box coordinates $B_{t+l}$ in the trajectory network.

**Sparse annotation loss.** When there are no annotations in-between keyframes then we cannot optimize the trajectory anticipating. Since the task in the sparse annotation cases is object detection on annotated keyframes, we can hypothesize that the precise true location of an object along a trajectory $B^*_{t+l}$, is not essential, as long as the trajectory is piecewise continuous and smooth, and the starting and ending points of the trajectory are known. Therefore, we can rewrite our losses in Equation (2.4) to a sparsely-annotated variant $L^{(\text{sa})}_{\text{traj}}$ by changing the way in which we define the box-supervision. Explicitly, we replace the set of ground truth boxes $\{B^*_t, B^*_{t+1}, ..B^*_{t+T}\}$ with a pseudo box trajectory. The pseudo-box trajectory is defined by a continuous function, $r_t(\cdot)$ describing the trajectory at every timestep as: $\mathbb{T}^r_t = (B^*_t, r_{t+1}(B^*_t), ..r_{t+T}(B^*_t))$, relative to the true keyframe location $B^*_t$. Because the next keyframe is the last trajectory location, we also constrain the pseudo trajectory to match the true bounding box at the end of the trajectory: $r_{t+T}(B^*_t) = B^*_{t+T}$. In practice, we choose $r_t(\cdot)$ to be either linear interpolated box annotations or a parabola as it is continuous and does not assume linear object trajectories. Here, we only need bounding-box annotations every $T$ frames, for correcting the starting point and end-point of our predicted trajectory. Our sparsely-annotated loss $L^{(\text{sa})}_{\text{traj}}$ variants are useful when the dataset only has sparse annotations available.

## 2.4. EXPERIMENTS

**Datasets and evaluation setup.** We carefully test our hypotheses on a fully controlled MovingDigits dataset. We ablate model choices on a subset of ImageNet VID [1]. We show the practical benefits of our approach on the full ImageNet VID [1] and on EPIC KITCHENS-55 [2]. As a realistic sparsely annotated dataset we evaluate on YouTube-BoundingBoxes [3], which has approximately 1 keyframe annotation per second. For the ImageNet VID experiments, we train our model on a combination of ImageNet VID and DET datasets as a common practice in [10, 14, 16, 17]. To quantify detection accuracy we adopt the common approach [12, 14–16] by computing mean Average Precision (mAP), where a detection is correct if its Intersection over Union (IoU) with the ground truth is sufficiently large.

Note that although our method samples keyframes during training, we do evaluate on all available frames at test time, unless stated otherwise.

**Implementation details.** We use Faster R-CNN [64] with an ImageNet pre-trained ResNet base as the object detection backbone. Our trajectory prediction sub-network contains three fully-connected layers with 1024 dimensions for the middle layer, see Figure 2.2. We train our network for 120k iterations on ImageNet VID and 173k iterations on Epic Kitchens with the SGD optimizer, on 4 GPUs. For ImageNet VID, the initial learning rate is $10^{-3}$ and is divided by 10 at 80K iterations. For Epic Kitchens, the initial learning rate is $5 \times 10^{-4}$ and is divided by 10 at 120K iterations. For YouTube-BoundingBoxes, the initial learning rate is $5 \times 10^{-4}$ and is divided by 10 at 100k iterations.

### 2.4.1. HYPOTHESIS TESTING

We test our hypotheses by creating a fully controlled dataset, MovingDigits, where we pair each of the 10 MNIST digit classes with a linear motion of 2px per frame, see Figure 2.4. Each video has 32 frames with a frame size of 64×64 px. We created 200 videos for training and 80 videos for testing, with an equal number of videos per class. We use a trajectory length of $T$=8, train for 1.25k iterations and use a ResNet-18 feature extractor.

**[H1]: Can the model anticipate motion trajectories?** To verify if our model can anticipate trajectories from a single static input frame $t$, we train on MovingDigits, where each digit has its own linear motion. We calculate the average IoU over the predicted trajectories for the test videos. The IoU is 0.95, which is near-perfect compared to the IoU of 0.79 for no motion anticipation. We show an example of the predicted bounding boxes (Bbox) by our method and the ground truth bounding boxes (GT Bbox) from time steps $t$ to $t+3$ in Figure 2.4. We conclude that our model can successfully learn motion by anticipating trajectories from a static input frame.

**[H2]: Anticipating improves static detection.** The motion cues in-between the static keyframes offer an additional source of supervision. Here, we investigate how the motion anticipating affects the static object detector where we evaluate at test-time on keyframes only. We explore what types of motion to predict between the static keyframes. We consider four types of motion supervision for predicting box trajectories: (1) *Ground truth motion*: trained on true bounding-box trajectories annotated at every frame; (2) *Simulated smooth motion*: bounding boxes move between keyframes according to a smooth parabola. (Details and examples are in the supplementary material); (3) *Randomized positions*: there is motion, but it is not smooth, the boxes in-between the keyframes can occur at any random position in the image; (4) *No motion*: without motion prediction, the static object detector baseline trained at every video frame.

Table 2.1 shows the keyframe mAP scores for IoU@[0.50:.05:0.95]. The *No motion* static object detector is the baseline, which uses no motion. The *Randomized positions* as supervision is detrimental for object detection because the motion is random, and unpredictable. This likely overfits to the random position prediction on the training set, which negatively affects the static detections as well. Interestingly, both *Ground truth motion* and *Simulated smooth motion* motion supervision improve the static keyframe detection. We analyzed the results, and notice an improvement in the IoU. We speculate that the anticipating loss encourages detecting static regions that are most likely to move coherently, with consistent motion offsets: i.e. same direction. We conclude that, for non-random motions, adding motion anticipating as an additional source of supervision improves static object detection at keyframes, even when we do not use the ground truth motion between keyframes.

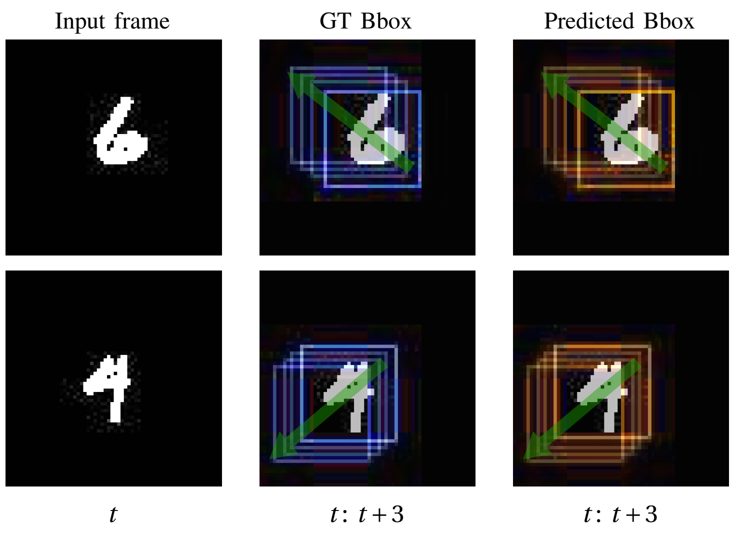


Fig. 2.4: **[H1]: Trajectory anticipation on MovingDigits.** We show a single static input frame of digits 6 and 4 and their time-accumulated ground truth and predicted bounding boxes in timesteps $t\colon t+3$. Each digit class has an associated linear motion (green arrow). The match of our predicted bounding boxes and the ground truth bounding boxes shows that our model can predict trajectories from a static frame.

| Motion | Keyframe mAP (%) |
|---|---|
| Randomized positions | 62.68 |
| No motion | 73.51 |
| Simulated smooth motion | 76.57 |
| Annotated motion | 79.31 |

Tab. 2.1: **[H2] Influence of motion anticipating on static object detection.** Static keyframe detection mAP on MovingDigits for varying motion type supervision. For non-random motions, adding motion anticipating improves static object detection at keyframes.

### 2.4.2. ABLATION OF MODEL COMPONENTS

We run all ablation experiments on an ImageNet VID [1] subset containing the classes ‘dog’, ‘giant_panda’, and ‘hamster’. We use a ResNet-101 for feature extraction.

**[A1]: The effect of the trajectory loss.** We evaluate each term in our trajectory loss $L_{\text{traj}} = L_{\sum} + L_{\text{bag}(\delta)}$ in Equation (2.4), compared to the standard loss $L_{\text{bag}}$ in Equation (3.2). The mAP scores for trajectory lengths $T$=4 and $T$=8 are in Table 2.2. The $L_{\text{traj}}$ loss has a higher mAP than the bag loss $L_{\text{bag}}$ for both trajectory lengths. $L_{\text{traj}}$ enforces continuity and smoothness in the predictions over time thus leads to more precise predictions. Furthermore, the improvement of $L_{\text{traj}}$ over $L_{\text{bag}}$ is larger for longer trajectories, e.g. for $T$=4 and $T$=8, $L_{\text{traj}}$ outperforms $L_{\text{bag}}$ by 1.66% and 2.18% respectively. Our $L_{\text{traj}}$

loss is defined using offsets, without offsets the $L_{\sum}$ would be equivalent to $L_{\text{bag}}$. We observe that the effect of predicting offsets between neighboring boxes $\delta_{t+l}$, instead of bounding-box coordinates $B_{t+l}$ gives an improvement of 0.81% and 0.89%. When considering $L_{\text{bag}(\delta)}$ or $L_{\sum}$ individually, their mAP is lower or on par with $L_{\text{traj}}$. This is because $L_{\text{bag}(\delta)}$ cannot enforce trajectory continuity, while $L_{\sum}$ cannot ensure that trajectories do not diverge by accumulating errors over time. Predicting offsets instead of box coordinates results in better accuracy.

**[A2]: The effect of the sparse annotation loss.** We test our model using the $L_{\text{traj}}^{(\text{sa})}$ loss for sparse annotations. To evaluate this in a controlled setting, we sub-sample keyframe annotations from the fully annotated ImageNet VID subset to mimic a sparse annotation scenario, and evaluate only on keyframes. We compare ground truth motion result with the result of anticipating linearly interpolated trajectories between keyframes. Table 2.3 shows that the $L_{\text{bag}(\delta)}^{(\text{sa})}$ component does not contribute much for the keyframe detection. This is because $L_{\text{bag}(\delta)}^{(\text{sa})}$ constrains the predicted offsets to match the pairwise offsets of the pseudo trajectory at every frame, which is not useful here since the motion is simulated. The $L_{\sum}^{(\text{sa})}$ component performs on par with the $L_{\text{traj}}^{(\text{sa})}$. And the keyframe detection accuracy with the sparse annotation loss $L_{\text{traj}}^{(\text{sa})}$ is close to that with the proposed fully-supervised loss $L_{\text{traj}}$.

**[A3]: Inference speed vs. accuracy trade-off.** We can control the inference speed by sampling fewer keyframes, and thus predicting longer trajectories. We analyse the speed vs. accuracy trade-off of our method on the subset of ImageNet VID. Figure 2.5 shows we can reach an mAP of 89.2% with a runtime of 39.6 FPS. Moreover a noticeable drop in mAP occurs at trajectories longer than 10 frames. However, when the trajectory is too long, the object motion may vary or the object may leave the frame or new object may enter the frame. These changes result in the decrease of our model's performance. The prediction trajectory length can be chosen according to the speed-accuracy trade-off.

### 2.4.3. STATE-OF-THE ART COMPARISONS

**[S1]: Experiments on ImageNet VID.** We compare our method with state-of-the-art video object detection methods on ImageNet VID in Table 2.4 using a ResNet-101 as the feature extractor. We mark methods without post-processing with ✓. Methods with post-processing add extra computational cost. Our method does not use any post-processing. We use a prediction trajectory of length $T$=4, as this already allows a considerable reduction in computation speed. Among all methods, our method is the most accurate with a 87.2% mAP, which has a 2.7% improvement over the leading MEGA [15]. We also report runtime (FPS) and train-time (hrs/epoch) to show that our method is fast and efficient. We measure the efficiency using the code provided by the original papers, and where the code is not available we mark this with '-'. Note that for the training time this is a rough estimate, when considering the same settings (batch-size, GPU) for all methods. We tested the inference runtime speed on a single NVIDIA GTX 1080 Ti. In terms of both training time and runtime our method is most efficient. Our method has fast inference time with 39.6 FPS, which is $\approx 1.8\times$ faster

| Loss | Trajectory length (mAP %) T=4 | T=8 |
|---|---|---|
| $L_{\mathbf{bag}}$ | 87.54 ± 0.16 | 83.97 ± 0.71 |
| $L_{\Sigma}$ | 87.95 ± 0.27 | 84.09 ± 0.67 |
| $L_{\text{bag}(\delta)}$ | 85.19 ± 0.66 | 79.73 ± 1.01 |
| $L_{\text{traj}}$ | **89.20** ± **0.21** | 86.15 ± 0.75 |

Tab. 2.2: **[A1]: Loss choice.** Compared to $L_{\text{bag}}$, the $L_{\text{bag}(\delta)}$ loss does worse, whereas $L_{\Sigma}$ performs on par. Their combination in $L_{\text{traj}} = L_{\Sigma} + L_{\text{bag}(\delta)}$ outperforms $L_{\text{bag}}$. Predicting offsets instead of bounding-box coordinates in the trajectory network gives better results. These patterns are consistent over both trajectory lengths.

| | Keyframe mAP% | | | |
|---|---|---|---|---|
| | $L_{\text{traj}}$ | Sparse annotation loss | | |
| | | $L_{\Sigma}^{(\text{sa})}$ | $L_{\text{bag}(\delta)}^{(\text{sa})}$ | $L_{\text{traj}}^{(\text{sa})}$ |
| T=4 | 91.03 ± 0.19 | 90.35 ± 0.34 | 74.61 ± 0.72 | 90.76 ± 0.26 |
| T=8 | 87.94 ± 0.66 | 86.98 ± 0.73 | 68.76 ± 0.99 | 87.12 ± 0.69 |

Tab. 2.3: **[A2]: Sparse annotation loss analysis.** We sub-sample keyframes of the fully annotated ImageNet VID subset to mimic the sparse annotations, and evaluate our sparse annotation loss on keyframe detection. The $L_{\text{bag}(\delta)}^{(\text{sa})}$ does not improve the detection accuracy, while $L_{\Sigma}^{(\text{sa})}$ performs on par with the $L_{\text{traj}}^{(\text{sa})}$. The sparse annotation loss achieves a comparable accuracy on keyframe detection with our fully-supervised loss $L_{\text{traj}}$.

than existing fast methods like LWND [19] and ST-Lattice [13]. Our efficiency comes from predicting object locations for the next $T$ frames, while processing only temporally sub-sampled keyframes.

**Comparison on different motion speeds.** We also compare on different motion speeds with other methods as shown in Table 2.5. The category of object motion speeds in ImageNet VID follows FGFA [10]. Our method improves mAP significantly on slow and medium motion speeds and achieves comparable results to the previous best method on fast motion speed, which shows the effectiveness of our method across different motion speeds.

**[S2]: Experiments on EPIC KITCHENS-55.** In EPIC KITCHENS-55, each frame contains avg/max 1.7/9 objects, which is more challenging compared to ImageNet VID. The Epic Kitchens video object detection task consists of 32 different kitchens and 290 classes. The training set has 272 video sequences captured in 28 kitchens. For evaluation, 106 sequences collected in the same 28 kitchens (S1) and 54 sequences

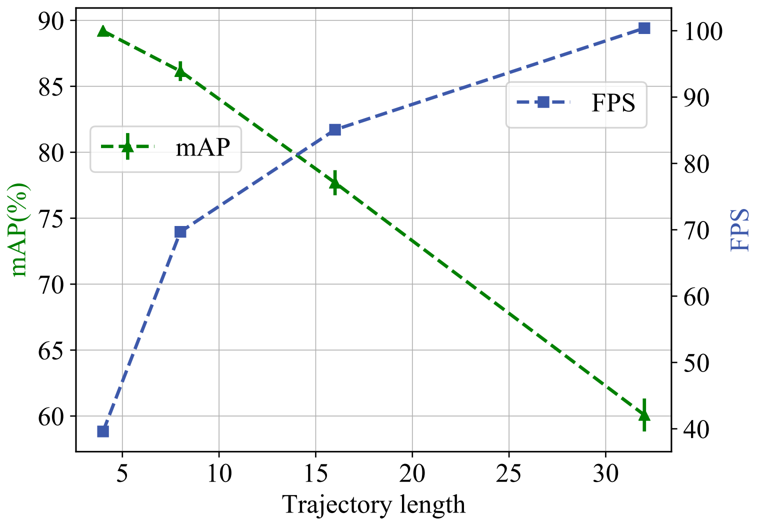


Fig. 2.5: **[A3] Inference speed vs. accuracy trade-off.** We can increase inference speed (FPS) by sampling fewer keyframes and predicting longer trajectories. Yet, with increasing trajectory length the mAP decreases. The prediction trajectory length can be chosen according to the required speed-accuracy trade-off.

collected in 4 other unseen kitchens (S2) are used. We use a prediction trajectory length of $T$=4 and evaluate for two IoU thresholds of 0.5 and 0.75.

As summarized in Table 2.6, our method is more accurate than previous state-of-the-art methods for both Seen/Unseen splits. This indicates that our method is applicable to more complex video detection tasks.

### 2.4.4. SPARSELY ANNOTATED VIDEOS

The YouTube-BoundingBoxes dataset has sparse annotations: the video frame rate is 30 frames per second, and on average it only has annotations at 1 fps. We evaluate our method compared to the Faster R-CNN [64] baseline on YouTube-BoundingBoxes in Table 2.7. The networks receive as input keyframes sampled with a step 60 and 30, which is every 60 frames and 30 frames in the video, respectively. During training, for the Faster R-CNN we use the labels at the keyframes, while for our method we use labels with a step of 30 or 1 over the input keyframes. For a label step of 30 and a keyframe step of 60, we use 2× more labels than input frames, while for a label step of 1 and keyframe step of 30, we use 30× more labels than frames. Since the video annotations are sparse we do not have labels at every frame, therefore for a label step of 1, we use our sparse annotation loss. For the $L_{\text{traj}}^{(\text{sa})}$ we define the box trajectories to be linearly interpolated pseudo trajectories. For keyframes sampled with a step of 60, our method achieves higher accuracy than Faster R-CNN by using 2× more labels, while processing the same input keyframes. With the same input keyframes every 30 frames,

| Methods | Backbone | No Post-proc. | mAP (%) | Train-time (hrs/epoch) | Runtime (FPS) |
|---|---|---|---|---|---|
| Faster-RCNN [64] | R101 | ✓ | 73.6 | 1.55 | 21.2 |
| LWND [19] | R101 | ✓ | 76.3 | - | 20.0 |
| FGFA [10] | R101 |  | 78.4 | 6.59 | 5.0 |
| THP [11] | R101+DCN | ✓ | 78.6 | - | - |
| ST-Lattice [13] | R101 |  | 79.6 | 1.40 | 20.0 |
| D&T [22] | R101 |  | 80.2 | 6.56 | 5.0 |
| MANet [12] | R101 |  | 80.3 | 6.88 | 4.9 |
| STSN [66] | R101+DCN |  | 80.4 | - | - |
| STMN [67] | R101 |  | 80.5 | 2.49 | 13.2 |
| TROI [68] | R101 |  | 80.5 | 5.18 | 6.4 |
| SELSA [14] | R101 |  | 80.5 | 3.15 | 10.6 |
| OGEMN [69] | R101+DCN |  | 81.6 | - | 8.9 |
| SparseVOD [70] | R101 | ✓ | 81.9 | - | 14.4 |
| BoxMask [71] | R101 | ✓ | 83.2 | - | 6.1 |
| RDN [72] | R101 |  | 83.8 | - | - |
| HVRNet [17] | R101 |  | 83.8 | - | - |
| TF-Blender [16] | R101 | ✓ | 83.8 | - | 4.9 |
| MEGA [15] | R101 |  | 84.5 | 6.34 | 5.3 |
| Ours | R101 | ✓ | **87.2** | **0.78** | **39.6** |

Tab. 2.4: **[S1]: Experiments on ImageNet VID.** We indicate the methods without video-level post-processing (e.g. SeqNMS, Tube Rescoring, BLR) with ✓in the table. 'No Post-proc.' means no post-processing. R101 here is ResNet-101. The runtime is measured on a NVIDIA GTX 1080 Ti. Our method achieves the best performance and fastest runtime among all the methods.

| Methods | Backbone | mAP (%) | mAP (%) (slow) | mAP (%) (medium) | mAP (%) (fast) |
|---|---|---|---|---|---|
| FGFA [10] | R101 | 78.4 | 83.5 | 75.8 | 57.6 |
| MANet [12] | R101 | 80.3 | 86.9 | 76.8 | 56.7 |
| SELSA [14] | R101 | 80.5 | 86.9 | 78.9 | 61.4 |
| OGEMN [69] | R101+DCN | 81.6 | 86.2 | 78.7 | 61.1 |
| HVRNet [17] | R101 | 83.8 | 88.7 | 82.3 | **66.6** |
| IFFNet [18] | R101 | 79.7 | 87.5 | 78.7 | 60.6 |
| Ours | R101 | **87.2** | **92.2** | **86.1** | 66.5 |

Tab. 2.5: mAP on ImageNet VID across different motion speeds. Our method improves mAP on different motion speeds.

our method even achieves a 1.1% higher mAP than Faster R-CNN while optimizing a simulated motion between these frames. This result indicates that the precise location of an object along a trajectory is not essential for keyframe detection.

| | S1 | | S2 | |
|---|---|---|---|---|
| Methods | mAP@.5 | mAP@.75 | mAP@.5 | mAP@.75 |
| EPIC [2] | 34.2 | 8.5 | 32.0 | 7.9 |
| Faster-RCNN [14] | 36.6 | 9.9 | 31.9 | 7.4 |
| SELSA [14] | 37.9 | 9.8 | 34.8 | 8.1 |
| SELSA-ReIm + TROI [68] | 42.2 | - | 39.6 | - |
| BoxMask [71] | 44.3 | 18.5 | 41.3 | 15.7 |
| Ours | **44.9** | **18.7** | **41.7** | **16.0** |

Tab. 2.6: **[S2]: Experiments on EPIC KITCHENS-55.** S1 and S2 represent Seen and Unseen splits respectively. Our method achieves promising results for both test sets and IoU thresholds.

| Methods | Keyframe step | Label step | mAP (%) |
|---|---|---|---|
| Faster-RCNN [64] | 60 | 60 | 47.6 |
| Faster-RCNN [64] | 30 | 30 | 58.7 |
| Ours $L_{\text{traj}}$ | 60 | 30 | 51.3 |
| Ours $L_{\text{traj}}^{(\text{sa})}$ | 30 | 1 | **59.8** |

Tab. 2.7: **Sparsely annotated data.** We report mAP on the sparsely annotated YouTube-BoundingBoxes. The sparse annotation loss $L_{\text{traj}}^{(\text{sa})}$ using interpolated pseudo trajectory labels improves keyframe detection. Our method outperforms Faster R-CNN by making effective use of annotations, even if these are not present in the dataset.

### 2.4.5. Method limitations

Despite our state-of-the-art results, with successful predictions as in Figure 2.6(a), we also identify several limitations. In practice, multiple motion patterns can be associated with the exactly same appearance: e.g people can walk or jump. If a specific object's appearance contains multiple motion patterns, our method may fail to predict the correct object locations. Another failure case is if objects appear or disappear in the middle of a trajectory. In these cases, we will either miss the objects or over-predict. Yet another limitation of our method is the assumption that the motion changes smoothly. If the object trajectories have extremely large variations in a short timestep because of the video frame rate, or other reasons, for example, the dog in Figure 2.6(b) suddenly changed its moving direction, our trajectory network may not be able to learn such complex motion patterns. We might miss some detection of intermediate frames as shown in Figure 2.6(b), but our method can recover the detection by the next keyframe detection. While these hypothetical limitations might exist, they can only influence our method minimally, since we miss only a few hundred milliseconds, which is reflected in our state-of-the-art results compared to other methods for video object detection.

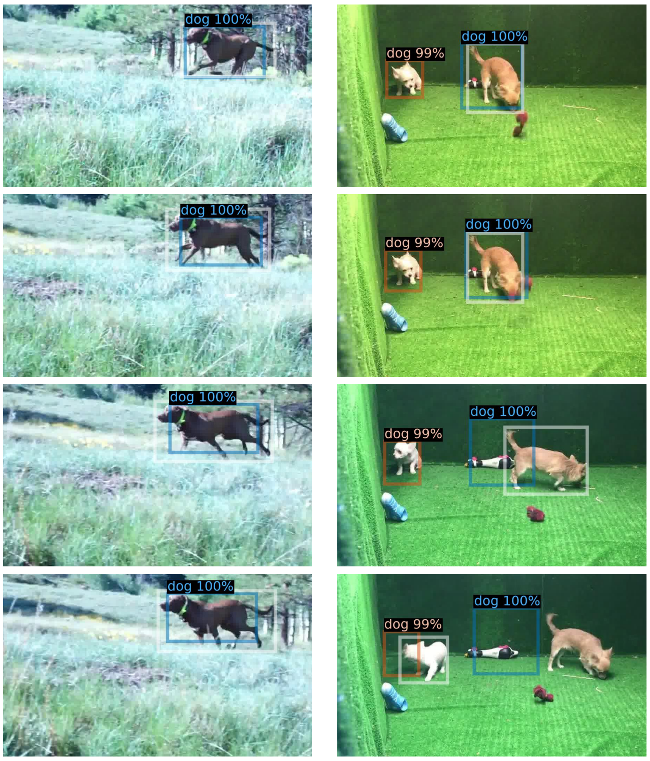


Fig. 2.6: Example predictions. The white boxes are the ground truth. We show one success case in (a) containing a 'dog' running at a fast speed. In (b) the failure case shows a 'dog' suddenly changes its moving direction, we missed the detection in the third and fourth frames. Yet we recovered the detection at the next keyframe detection. When objects move smoothly our detections are accurate, yet when the motion largely varies, is unpredictable, and objects enter and leave the frame, our method fails.

## 2.5. Conclusion

We propose a method to efficiently detect objects in videos by predicting their future locations from a static input keyframe and the ground truth locations of all frames. Our method associates appearance with motion. Different motion contexts have different appearances, thus we can model various motion patterns from the keyframes with different appearances. Because we predict the future object locations over multiple frames, we do not need to process every frame of the video, but only a subset of the keyframes, which makes our method efficient. Moreover, by learning to predict object trajectories we improve the object detection accuracy when compared to the state-of-the-art on multiple datasets. Finally, by using pseudo object trajectories defined by smooth continuous functions, we can improve object detection accuracy on sparsely annotated videos.

## 2.6. APPENDIX

### 2.6.1. DERIVATION FOR LOSS EQUIVALENCE

In this appendix section, we provide the derivation to show that if we would predict bounding boxes $B_{t+l}$ in the trajectory network, instead of offsets between pairs of bounding boxes $\delta_{t+l}$, Equation (2.2) would reduce to Equation (3.2) and the temporal ordering would not be enforced.

If we predict bounding boxes $B_{t+l}$ and use $\delta_{t+k} = (B_{t+k} - B_{t+k-1}), k \in \{1,..,l\}$, the sum $\sum_{k=1}^{l} \delta_{t+k}$ can be rewritten as follows:

$$\begin{aligned}
\sum_{k=1}^{l} \delta_{t+k} &= \sum_{k=1}^{l} (B_{t+k} - B_{t+k-1}), \\
&= \sum_{k=1}^{l} (B_{t+k}) - \sum_{k=1}^{l} (B_{t+k-1}), \\
&= \sum_{k=1}^{l} (B_{t+k}) - \sum_{k=0}^{l-1} (B_{t+k}), \\
&= \sum_{k=1}^{l-1} (B_{t+k}) + B_{t+l} - \sum_{k=1}^{l-1} (B_{t+k}) - B_t, \\
&= B_{t+l} - B_t.
\end{aligned}$$

And we fill the above in Equation (2.2). Then we have,

$$\begin{aligned}
L_{\Sigma}(B^*, \overleftrightarrow{\mathbb{T}}_t) &= \sum_{l=0}^{T} \mathscr{L}_1 \left( (B^*_{t+l} - B_t) - \sum_{k=1}^{l} \delta_{t+k} \right), \\
&= \sum_{l=0}^{T} \mathscr{L}_1 \left( B^*_{t+l} - B_t - \sum_{k=1}^{l} \delta_{t+k} \right), \\
&= \sum_{l=0}^{T} \mathscr{L}_1 \left( B^*_{t+l} - B_t - B_{t+l} + B_t \right), \\
&= \sum_{l=0}^{T} \mathscr{L}_1 \left( B^*_{t+l} - B_{t+l} \right).
\end{aligned}$$

which is the same as Equation (3.2):

$$L_{\text{bag}}(B^*, \{B_t, .., B_{t+T}\}) = \sum_{l=0}^{T} \mathscr{L}_1 (B^*_{t+l} - B_{t+l}).$$

### 2.6.2. DETAILS FOR THE *Simulated smooth motion*

In this section, we describe how we create the *Simulated smooth motion*: bounding boxes move between keyframes according to a smooth parabola, and the change of width and height is linearly interpolated. Given the center points of two keyframe digits $(x_t, y_t)$ and $(x_{t+T}, y_{t+T})$, we choose the focus $F = (0, f), f = 8$ for the parabola, then the

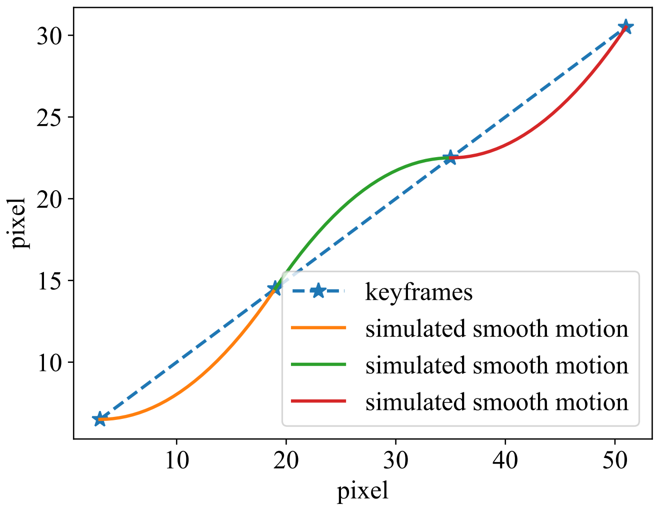


Fig. 2.7: An example of simulated smooth motion generated by parabola functions. The parabola represents the trajectory of intermediate digit locations between every two keyframes. The simulated smooth motion is smooth and continuous.

parabola can be written as,

$$y = \frac{1}{4f}x^2 - \frac{v_1}{2f}x + \frac{v_1^2}{4f} + v_2, \tag{2.5}$$

where the vertex is $V = (v_1, v_2)$. By filling in $(x_t, y_t)$ and $(x_{t+T}, y_{t+T})$, we can get the value of $v_1, v_2$. For every pairwise neighbouring keyframes, we can have a parabola that acts as a simulated smooth trajectory for intermediate locations of digits. Here we show an example of having four keyframes and the simulated smooth motion presents as a parabola in Figure 2.7. Because the digits move linearly in MovingDigits dataset, we have the digits of keyframes staying in a linear line. We choose the focus of every second parabola sequence to be $F = (0, -8)$ to make all the parabola trajectories smoothly connected.

# REFERENCES

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[2] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, *et al.* “Scaling egocentric vision: The epic-kitchens dataset”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 720–736.

[3] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. “Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5296–5305.

[4] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. “Seq-nms for video object detection”. In: *CoRR* (2016).

[5] K. Kang, W. Ouyang, H. Li, and X. Wang. “Object detection from video tubelets with convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 817–825.

[6] G. Bertasius and L. Torresani. “Classifying, segmenting, and tracking object instances in video with mask propagation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9739–9748.

[7] C.-C. Lin, Y. Hung, R. Feris, and L. He. “Video instance segmentation tracking with a modified vae architecture”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13147–13157.

[8] Y. Lu, C. Lu, and C.-K. Tang. “Online video object detection using association LSTM”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2344–2352.

[9] S. Tripathi, Z. C. Lipton, S. Belongie, and T. Nguyen. “Context matters: Refining object detection in video with recurrent neural networks”. In: *CoRR* (2016).

[10] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. “Flow-guided feature aggregation for video object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 408–417.

[11] X. Zhu, J. Dai, L. Yuan, and Y. Wei. “Towards high performance video object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7210–7218.

[12] S. Wang, Y. Zhou, J. Yan, and Z. Deng. “Fully motion-aware network for video object detection”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 542–557.

[13] K. Chen, J. Wang, S. Yang, X. Zhang, Y. Xiong, C. C. Loy, and D. Lin. “Optimizing video object detection via a scale-time lattice”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7814–7823.

[14] H. Wu, Y. Chen, N. Wang, and Z. Zhang. “Sequence level semantics aggregation for video object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9217–9225.

[15] Y. Chen, Y. Cao, H. Hu, and L. Wang. “Memory enhanced global-local aggregation for video object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10337–10346.

[16] Y. Cui, L. Yan, Z. Cao, and D. Liu. “TF-Blender: Temporal Feature Blender for Video Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8138–8147.

[17] M. Han, Y. Wang, X. Chang, and Y. Qiao. “Mining inter-video proposal relations for video object detection”. In: *European conference on computer vision*. Springer. 2020, pp. 431–446.

[18] R. Jin, G. Lin, C. Wen, J. Wang, and F. Liu. “Feature flow: In-network feature flow estimation for video object detection”. In: *Pattern Recognition* 122 (2022), p. 108323.

[19] Z. Jiang, P. Gao, C. Guo, Q. Zhang, S. Xiang, and C. Pan. “Video object detection with locally-weighted deformable neighbors”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8529–8536.

[20] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. “Visual tracking: An experimental survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1442–1468.

[21] P. Bergmann, T. Meinhardt, and L. Leal-Taixe. “Tracking without bells and whistles”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 941–951.

[22] C. Feichtenhofer, A. Pinz, and A. Zisserman. “Detect to track and track to detect”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3038–3046.

[23] D. Held, S. Thrun, and S. Savarese. “Learning to track at 100 fps with deep regression networks”. In: *European conference on computer vision*. Springer. 2016, pp. 749–765.

[24] P. Sun, J. Cao, Y. Jiang, R. Zhang, E. Xie, Z. Yuan, C. Wang, and P. Luo. “Transtrack: Multiple object tracking with transformer”. In: *CoRR* (2020).

[25] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera. “Deep learning in video multi-object tracking: A survey”. In: *Neurocomputing* 381 (2020), pp. 61–88.

[26] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles. “Learning to decompose and disentangle representations for video prediction”. In: *Advances in neural information processing systems* 31 (2018).

[27] B. Liu, Y. Chen, S. Liu, and H.-S. Kim. “Deep learning in latent space for video prediction and compression”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 701–710.

[28] M. Narasimhan, S. Ginosar, A. Owens, A. A. Efros, and T. Darrell. “Contrastive Video Textures”. In: (2020).

[29] V. Vukotić, S.-L. Pintea, C. Raymond, G. Gravier, and J. C. v. Gemert. “One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network”. In: *International conference on image analysis and processing*. Springer. 2017, pp. 140–151.

[30] Y. Abu Farha, A. Richard, and J. Gall. “When will you do what?-anticipating temporal occurrences of activities”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5343–5352.

[31] R. Girdhar and K. Grauman. “Anticipative video transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13505–13515.

[32] T. Mahmud, M. Hasan, and A. K. Roy-Chowdhury. “Joint prediction of activity labels and starting times in untrimmed videos”. In: *Proceedings of the IEEE International conference on Computer Vision*. 2017, pp. 5773–5782.

[33] G. Singh, S. Saha, M. Sapienza, P. H. S. Torr, and F. Cuzzolin. “Online Real-Time Multiple Spatiotemporal Action Localisation and Prediction”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[34] R. Gao, B. Xiong, and K. Grauman. “Im2flow: Motion hallucination from static images for action recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5937–5947.

[35] S. Pintea, J. v. Gemert, and A. Smeulders. “Deja vu: Motion prediction in static images”. In: *Proceedings of the European Conference on Computer V ision (ECCV)*. 2014.

[36] J. Walker, C. Doersch, A. Gupta, and M. Hebert. “An uncertain future: Forecasting from static images using variational autoencoders”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851.

[37] A. Rasouli, I. Kotseruba, T. Kunic, and J. K. Tsotsos. “Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6262–6271.

[38] Z. Zhang, J. Gao, J. Mao, Y. Liu, D. Anguelov, and C. Li. “Stinet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11346–11355.

[39] J. Gu, C. Sun, and H. Zhao. "Densetnt: End-to-end trajectory prediction from dense goal sets". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15303–15312.

[40] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He. "A large-scale study on unsupervised spatiotemporal representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3299–3309.

[41] T. Han, W. Xie, and A. Zisserman. "Video representation learning by dense predictive coding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[42] T. Han, W. Xie, and A. Zisserman. "Memory-augmented dense predictive coding for video representation learning". In: *European conference on computer vision*. Springer. 2020, pp. 312–329.

[43] R. Li, Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei. "Motion-focused contrastive learning of video representations". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2105–2114.

[44] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. "Learning and using the arrow of time". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8052–8060.

[45] C. Feichtenhofer. "X3d: Expanding architectures for efficient video recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 203–213.

[46] D. Kondratyuk, L. Yuan, Y. Li, L. Zhang, M. Tan, M. Brown, and B. Gong. "Movinets: Mobile video networks for efficient video recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16020–16030.

[47] J. Lin, C. Gan, and S. Han. "Tsm: Temporal shift module for efficient video understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7083–7093.

[48] X. Liu, S. L. Pintea, F. K. Nejadasl, O. Booij, and J. C. van Gemert. "No frame left behind: Full video action recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14892–14901.

[49] M. Zolfaghari, K. Singh, and T. Brox. "Eco: Efficient convolutional network for online video understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

[50] P. Sharma and R. Nevatia. "Efficient detector adaptation for object detection in a video". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3254–3261.

[51] V. Kalogeiton, V. Ferrari, and C. Schmid. "Analysing domain shift factors between videos and images for object detection". In: *IEEE transactions on pattern analysis and machine intelligence* 38.11 (2016), pp. 2327–2334.

[52] Z. Jiang, Y. Liu, C. Yang, J. Liu, P. Gao, Q. Zhang, S. Xiang, and C. Pan. "Learning where to focus for efficient video object detection". In: *European conference on computer vision*. Springer. 2020, pp. 18–34.

[53] G. Sun, Y. Hua, G. Hu, and N. Robertson. "Efficient One-Stage Video Object Detection by Exploiting Temporal Consistency". In: *European Conference on Computer Vision*. Springer. 2022, pp. 1–16.

[54] Z. Xu, E. Hrustic, and D. Vivet. "Centernet heatmap propagation for real-time video object detection". In: *European conference on computer vision*. Springer. 2020, pp. 220–234.

[55] K. All, D. Hasler, and F. Fleuret. "FlowBoost—Appearance learning from sparsely annotated video". In: *CVPR 2011*. IEEE. 2011, pp. 1433–1440.

[56] S. D. Jain and K. Grauman. "Supervoxel-consistent foreground propagation in video". In: *European conference on computer vision*. Springer. 2014, pp. 656–671.

[57] Y. Xu, Y. Wu, S. Nobuhara, K. Nishino, *et al.* "Video region annotation with sparse bounding boxes". In: *BMVC* (2020).

[58] P. Mettes, J. C. v. Gemert, and C. G. Snoek. "Spot on: Action localization from pointly-supervised proposals". In: *European conference on computer vision*. Springer. 2016, pp. 437–453.

[59] K. Dai, J. Zhao, L. Wang, D. Wang, J. Li, H. Lu, X. Qian, and X. Yang. "Video Annotation for Visual Tracking via Selection and Refinement". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10296–10305.

[60] K. G. Ince, A. Koksal, A. Fazla, and A. A. Alatan. "Semi-Automatic Annotation For Visual Object Tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1233–1239.

[61] A. Kuznetsova, A. Talati, Y. Luo, K. Simmons, and V. Ferrari. "Efficient video annotation with visual interpolation and frame selection guidance". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 3070–3079.

[62] C. Vondrick and D. Ramanan. "Video annotation and tracking with active learning". In: *Advances in Neural Information Processing Systems* 24 (2011).

[63] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[64] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015).

[65] R. Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[66] G. Bertasius, L. Torresani, and J. Shi. "Object detection in video with spatiotemporal sampling networks". In: 2018, pp. 331–346.

[67] F. Xiao and Y. J. Lee. “Video object detection with an aligned spatial-temporal memory”. In: 2018, pp. 485–501.

[68] T. Gong, K. Chen, X. Wang, Q. Chu, F. Zhu, D. Lin, N. Yu, and H. Feng. “Temporal ROI align for video object recognition”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2. 2021, pp. 1442–1450.

[69] H. Deng, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan. “Object guided external memory network for video object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6678–6687.

[70] K. A. Hashmi, D. Stricker, and M. Z. Afzal. “Spatio-Temporal Learnable Proposals for End-to-End Video Object Detection”. In: *arXiv preprint arXiv:2210.02368* (2022).

[71] K. A. Hashmi, A. Pagani, D. Stricker, and M. Z. Afzal. “BoxMask: Revisiting Bounding Box Supervision for Video Object Detection”. In: *arXiv preprint arXiv:2210.06008* (2022).

[72] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei. “Relation distillation networks for video object detection”. In: 2019, pp. 7023–7032.

# 3

# No frame left behind: Full video action recognition

*Not all video frames are equally informative for recognizing an action. It is computationally infeasible to train deep networks on all video frames when actions develop over hundreds of frames. A common heuristic is uniformly sampling a small number of video frames and using these to recognize the action. Instead, here we propose* full video action recognition *and consider all video frames. To make this computational tractable, we first cluster all frame activations along the temporal dimension based on their similarity with respect to the classification task, and then temporally aggregate the frames in the clusters into a smaller number of representations. Our method is end-to-end trainable and computationally efficient as it relies on temporally localized clustering in combination with fast Hamming distances in feature space. We evaluate on UCF101, HMDB51, Breakfast, and Something-Something V1 and V2, where we compare favorably to existing heuristic frame sampling methods.*

This chapter is published as: **Xin Liu**, Silvia L Pintea, Fatemeh Karimi Nejadasl, Olaf Booij, Jan C Van Gemert. *No frame left behind: Full video action recognition*. Conference on Computer Vision and Pattern Recognition (CVPR), 2021. arXiv: `https://arxiv.org/abs/2103.15395`

## 3.1. Introduction

Videos have arbitrary length with actions occurring at arbitrary moments. Current video recognition methods use CNNs on coarsely sub-sampled frames [1–10] because using all frames is computationally infeasible. Sub-sampling, however, can miss crucial frames for action recognition. For example, as shown in Figure 4.1, sampling the frame with the dish in the pan is crucial for correct recognition. We propose to do away with sub-sampling heuristics and argue for leveraging all video frames: Full video action recognition.

It is worth analyzing why training CNNs on full videos is computationally infeasible in terms of memory and calculations. The calculations in the forward pass yield activations, while the backward pass calculations give gradients which are summed over all frames to update the weights. Many of these calculations can be done in parallel and thus are well-suited for modern GPUs. When treating videos as a large collection of image frames, the amount of calculations are not too different from those on large image datasets [11]. Regarding memory, however, there is a crucial difference between videos and images: The video loss function is not per-frame but on the full video. Hence, to do the backward pass, all activations for each frame, for each filter in each layer need to be stored in memory. This even doubles for storing their gradients. With 10-30 frames per second, this quickly becomes infeasible for even just a few minutes of video. Existing approaches can trade off memory for compute [12–14] by not storing all intermediate layers, yet they do not scale to video as they would still need to store each frame. The main computational bottleneck for training video CNNs is memory for frame activations.

Here, we propose an efficient method to use all video frames during training. The forward pass computes frame activations and the backward pass sums the gradients over the frames to update the weights. Now, if only the network was linear, then a huge memory reduction could be gained by first summing all frame activations in the forward pass, which would reduce to just a single update in the backward pass. Yet, deep networks are infamously non-linear, and have non-linearities in the activation function and in the loss function. Thus, if all frames were independent, treating the non-linear network as linear would introduce considerable approximation errors. However, subsequent frames in a video are strongly correlated, and it's this correlation that makes it possible for existing approaches to use sub-sampling. Instead of sub-sampling, we propose to process all frames and exploit the frame correlations to create groups of frames where the network is approximately linear. We use the *ReLU* (Rectified Linear Unit) activation function, which is linear when the signs of two activations agree, to estimate which parts of the video are approximately linear. This allows us to develop an efficient clustering algorithm based on Hamming distances of frame activations as illustrated in Figure 4.1. By then aggregating the approximately linear parts in a video in the forward pass, we make large memory savings in the backward pass while still approximating the full video gradient.
We summarize the contributions of our work as follows:

- We propose a method that allows us to use most or even all video frames for training action recognition by approximated individual frame gradients with the gradients of temporally aggregated frame activations;

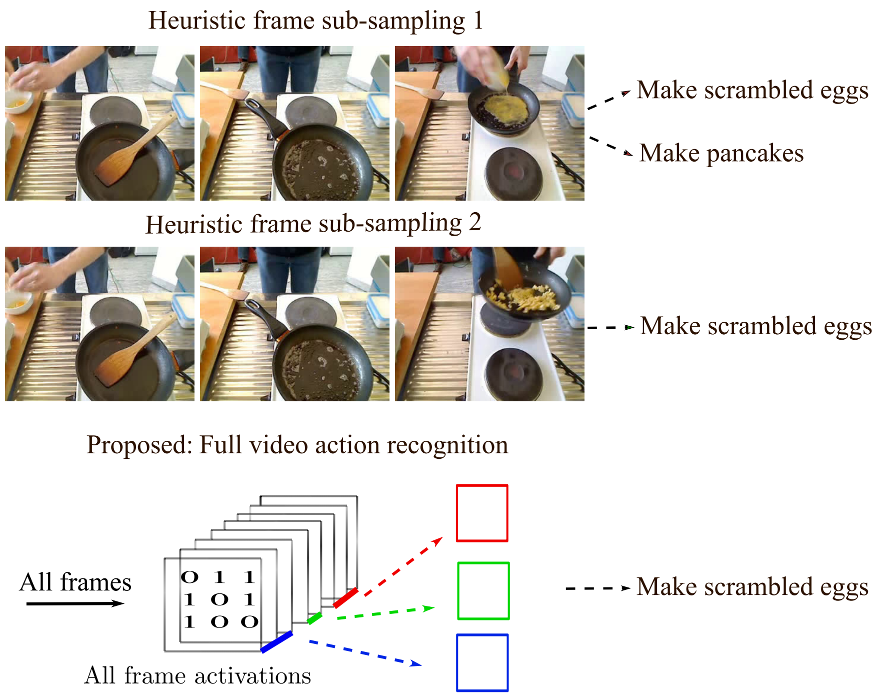


Fig. 3.1: Sub-sampling can miss crucial frames in videos and may cause confusion for action recognition: e.g. compare the two sub-samplings heuristics in row 1 and row 2: Without sampling the dish in the pan it is difficult to classify. Instead, as shown in row 3, we propose to efficiently use all frames during training by clustering frame activations along the temporal dimension and aggregating each cluster to a single representation. The temporal clustering is based on Hamming distances over frame activations, which is computationally fast. With the assumption that similar activations have similar gradients, the aggregated representations approximate the individual frame activations. We efficiently utilize all frames for training without missing important information.

- We devise an end-to-end trainable approach for efficient grouping of video frames based on temporally localized clustering and Hamming distances;
- Extensive experiments demonstrate that our method compares well to state-of-the-art methods on several benchmark datasets such as UCF101, HMDB51, Breakfast, and Something-Something V1 and V2.

## 3.2. RELATED WORK

**Action recognition architectures.** Actions in video involve motion, leading to deep networks which include optical flow [15–17], 3D convolutions [1, 18–20] and recurrent connections [16, 21–24]. Instead of heavy-weight motion representations, a single 2D image can reveal much of an action [7, 17, 25, 26]. 2D CNNs are extremely efficient, and by adding motion information by concatenating a 3D module in ECO [2], modeling temporal relations in TSN [27] or simply shifting filter channels over time in TSM [4] their efficiency is complemented by good accuracy. For this reason, we build on the TSM [4] architecture and modify it for full video action recognition.

**Frame sampling for action recognition.** Realistic videos contain more frames than can fit in memory. To address this, current methods train by using sub-sampled video frames [1, 2, 4, 7]. Additionally, the SlowFast [28] network also explores the resolution trade-off across temporal, spatial and channel dimension. Rather than using a fixed frame sampling strategy, the sampling can be adaptive [4, 6, 8–10], or learned to select the best frame [5], or rely on clip sampling [3]. In our work we do not sub-sample frames, but use all frames of the videos, however our clustering is adaptive as it dynamically adapt to the task and the loss function.

Using a subset of frames is computationally more efficient. Using 5-7 frames is sufficient for state-of-the-art action recognition on short videos [29]. Aiming for training efficiency, the work in [30] uses stochastic mini-batch dropping which drops complete batches rather than frames, with a certain probability. Similarly, [31] uses variable mini-batch shapes with different spatio-temporal resolutions varied according to a schedule. Unlike these methods, we do not focus on training efficiency, but propose a method that allows the network to see all video frames during training.

**Temporal pooling.** To integrate frame-level features, TSN [7] uses average pooling in the late layers of the network. ActionVLAD [32] integrates two-stream networks with a learnable spatio-temporal feature aggregation. Instead of performing temporal pooling or aggregation at a late stage of the network, in [33] RankSVM is used to rank frames temporally and then pool them together. As a follow-up, in [34] a 'dynamic image' is introduced, which is a compact representation of the videos frames using the 'rank pool' operation. In [35, 36] temporal aggregation via pooling and attention is used. Similar to these methods, our proposal performs a temporal pooling of the network activations, however this aggregation is done over clustered activations and it allows us to process all video frames.

**Efficient backpropagation.** Given that 2/3 of the training computations and memory are spent in the backward pass, existing work focuses on approximations. It is more memory efficient to recompute activations from the previous layer instead of storing them [14], however this comes at the cost of increased training time. In [37] gradient approximations are used where activations are overwritten when new frames are seen without waiting for the backward pass to be performed. Also for efficient backpropagation, randomized automatic differentiation can be used [38], gradients can be reused during training [39], or even quantized during backpropagation [40]. Similar to these works, we use all frames to approximate the full video gradient.

## 3.3. AGGREGATED TEMPORAL CLUSTERS

### 3.3.1. APPROXIMATING GRADIENTS

We enable the use of all frames of a video during training. To this end, we calculate a single gradient to approximate the gradients of a group of frames. Our hypothesis is that nearby frames in a video are alike, and thus have similar activations, leading to congruent updates. When using the *ReLU* (Rectified Linear Unit) activation function, we know that for activations with agreeing signs, the activation function is linear. Assuming that similar frames are approximately linear, the standard computation of the sum of gradients over all frames, becomes equivalent to first summing all frame activations and then computing a single gradient. This is computationally and memory efficient. Mathematically, for frames $i$, this can be formulated as:

$$\sum_i \nabla_{\mathbf{w}} L(h(\mathbf{x}_i \mathbf{w})) = \nabla_{\mathbf{w}} L\left(\sum_i h(\mathbf{x}_i \mathbf{w})\right), \tag{3.1}$$

where $\mathbf{x}$ are frame activations, $\mathbf{w}$ are the network weights, $h(\cdot)$ is an activation function, and $L(\cdot)$ is the loss function. Note that Equation (3.1) only holds in the ideal case when the activation function $h$ is linear for similar frames and the loss function $L$ is also linear. This is not generally the case, and this approximation introduces an error.

With the above ideal scenario in mind, we can use all video frames without calculating the gradient for each frame, by grouping frames that agree in the sign of their activations $\mathbf{x}$. Over these grouped activations we calculate a single gradient $\nabla_{\mathbf{w}} L(\sum_i h(\mathbf{x}_i \mathbf{w}))$. However, for similar frames the sign of their activation values may not be in complete agreement. Therefore, we aim to find which frames can be safely grouped together, to minimize the error introduced by our approximation in Equation (3.1).

### 3.3.2. ERROR BOUND FOR THE APPROXIMATION

For ease of explanation, we consider two input video frames and their activations $\mathbf{x}$=$\{\mathbf{x}_1, \mathbf{x}_2\}$, and a convolutional operation with parameters $\mathbf{w}$, denoted by $\mathbf{xw}$. The two frames have the same class label, $y$, since they come from the same video. We consider a multi-class setting using the cross-entropy loss in combination with the softmax function $q$, which for these two samples is:

$$L(\mathbf{x}, y) = -\frac{1}{2}\left(\log q_y(\mathbf{x}_1) + \log q_y(\mathbf{x}_2)\right), \tag{3.2}$$

where $q_c(\mathbf{x}_i)$=$\frac{\exp(h(\mathbf{x_i w}_c))}{\sum_{j=1}^{C} \exp(h(\mathbf{x_i w}_j))}$, $c$∈$\{1,..,C\}$ indexes video classes and $h(\cdot)$ is the *ReLU* activation function. The gradient of the loss with respect to $\mathbf{w}$ is:

$$\nabla_{\mathbf{w}} L(\mathbf{x}, y) = \frac{\mathbf{x}_1 (q_c(\mathbf{x}_1) - \delta_{yc}) + \mathbf{x}_2 (q_c(\mathbf{x}_2) - \delta_{yc})}{2}, \tag{3.3}$$

where $\delta_{yc}$ is the Dirac function which is 1 when $c$=$y$. In our method, we first average the two activations after the convolution and before the *ReLU*. We can do this, because

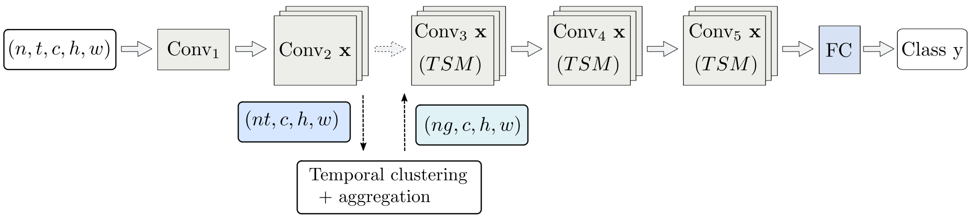

$(n, t, c, h, w)$
$\text{Conv}_1$
$\text{Conv}_2$ x
$\text{Conv}_3$ x $(TSM)$
$\text{Conv}_4$ x $(TSM)$
$\text{Conv}_5$ x $(TSM)$
FC
Class y
$(nt, c, h, w)$
$(ng, c, h, w)$
Temporal clustering + aggregation

Fig. 3.2: We adopt 2D ResNet-50 with TSM [4] a backbone. The input batch size is $n$ with $t$ frames. We cluster the activations of the first block of size $(nt, c, h, w)$ which groups $t$ frames into $g$ clusters and outputs new activations of size $(ng, c, h, w)$, as input to the next network blocks. Our full video method efficiently utilizes all frames and is end-to-end trainable.

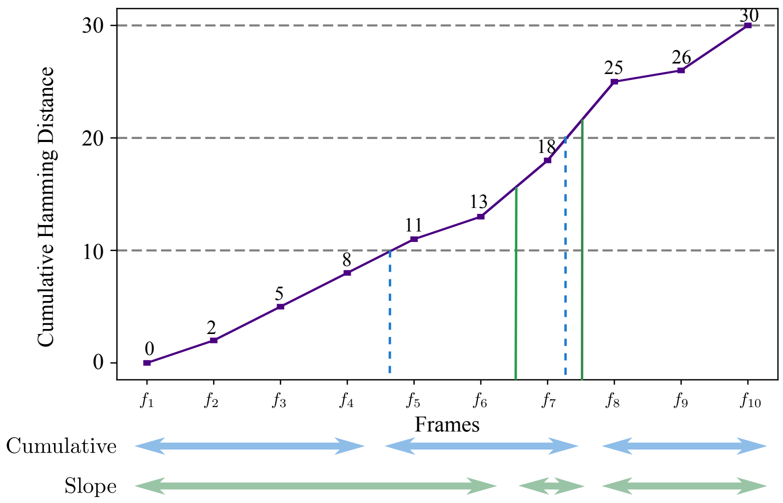


Fig. 3.3: An illustration of our two clustering algorithms. The numbers on the solid line are pair-wise Hamming distances and the solid line is the cumulative Hamming distance of frame $f_1$ to $f_{10}$. For $g$=3 clusters, the *cumulative clustering* groups frames by dividing the total cumulative distance on the $y$-axis into 3 equally distanced segments, as shown with the dashed lines resulting in the 3 clusters $(f_1 - f_4)$, $(f_5 - f_7)$ and $(f_8 - f_{10})$. The *slope clustering* algorithm is based on the slope of the curve and here selects the top-2 largest slopes, as shown with the solid green lines, which results in the 3 clusters: $(f_1 - f_6)$, $(f_7)$, $(f_8 - f_{10})$.

if we assume the activations have agreeing signs sign$(\mathbf{x}_1\mathbf{w})$=sign$(\mathbf{x}_2\mathbf{w})$, then it holds that: $\frac{h(\mathbf{x}_1\mathbf{w})+h(\mathbf{x}_2\mathbf{w})}{2} = h(\frac{\mathbf{x}_1\mathbf{w}+\mathbf{x}_2\mathbf{w}}{2})$. In this case the cross-entropy loss becomes:

$$\hat{L}(\mathbf{x}, y) = -\log q_y\left(\frac{\mathbf{x}_1 + \mathbf{x}_2}{2}\right). \tag{3.4}$$

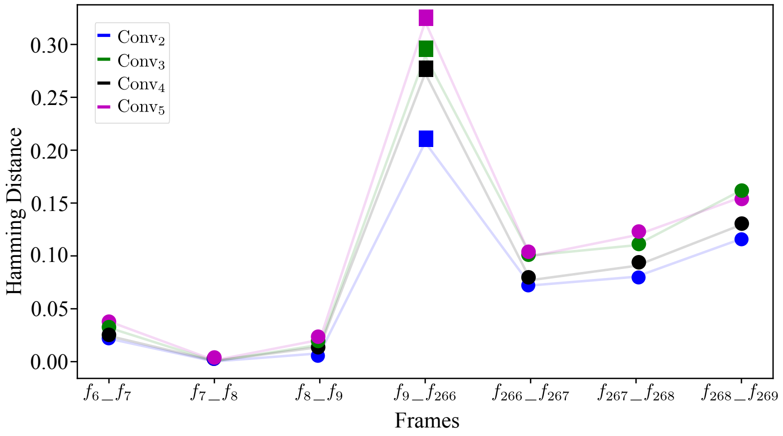


Fig. 3.4: Hamming distances between similar frames and dissimilar frames across 4 blocks of ResNet. The frames are taken from a single Breakfast [41] video. We denote the frames that are similar to their neighbors with circles and the dissimilar ones with squares. Hamming distances are consistent across blocks.

In the backward pass, we calculate a single gradient of the averaged activations as follows:

$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y) = \frac{\mathbf{x}_1 + \mathbf{x}_2}{2} \left( q_c \left( \frac{\mathbf{x}_1 + \mathbf{x}_2}{2} \right) - \delta_{yc} \right), \tag{3.5}$$

We now estimate the relative error introduced by our approximation by comparing equations Equation (3.3) and Equation (3.5) using Jensen's inequality. We start from the softmax function $q_c(\cdot)$ and we recover back equations Equation (3.3) and Equation (3.5). The softmax function $q_c(\cdot)$ is convex, therefore we can apply to it Jensen's inequality for the samples $\mathbf{x}_1$ and $\mathbf{x}_2$: $q_c\left(\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2}\right) \leq \frac{q_c(\mathbf{x}_1)+q_c(\mathbf{x}_2)}{2}$. We start by considering the case $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2} > 0$. If we multiply both sides of this inequality with $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2}$ we obtain that:

$$\begin{aligned} \frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} q_c \left( \frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} \right) &\leq \frac{\mathbf{x}_1 q_c(\mathbf{x}_1) + \mathbf{x}_2 q_c(\mathbf{x}_2)}{2} \\ &- \frac{1}{4}(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2)). \end{aligned} \tag{3.6}$$

In the left hand side of the inequality we recover precisely the $\nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y)$ given by Equation (3.5), while in the right hand side we recover Equation (3.3) minus the approximation error as $\nabla_{\mathbf{w}} L(\mathbf{x}, y) - \frac{1}{4}(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2))$. Note that for the case $y{=}c$ the additional Dirac terms in $\mathbf{x}$ cancel out. We now consider also the case $\frac{(\mathbf{x}_1+\mathbf{x}_2)}{2} \leq 0$, which together with the previous case leads to the following bound on the absolute difference between the gradients in Equation (3.3) and Equation (3.5):

$$|\nabla_{\mathbf{w}} L(\mathbf{x}, y) - \nabla_{\mathbf{w}} \hat{L}(\mathbf{x}, y)| \leq \frac{1}{4} |(\mathbf{x}_1 - \mathbf{x}_2)(q_c(\mathbf{x}_1) - q_c(\mathbf{x}_2))|. \tag{3.7}$$

Thus, the difference between the two gradient updates is bounded by a function depending on the difference between the activations and their softmax responses. The closer to 0 the difference between the activations the smaller the difference between their gradient updates. We show in the experimental section that, indeed, small differences in the activations entail small differences in the loss. The inequality in Equation (3.6) holds under the condition that the sign of activations agree. Therefore, we want to group frames based on the sign similarity of their activations.

### 3.3.3. Temporal clustering and aggregation

Using our proposal in Equation (3.5) allows training on all video frames. We group frames based on the sign agreement of their activations. An efficient way to do this, is to binarize the activation values by using the $sign$ function and compute a fast Hamming distance between binarized activations to determine which frames to group.

Consecutive frames in a video are more likely to be similar in appearance and are thus more likely to have similarly signed activations. Therefore, we explore two variants of a temporal clustering algorithm based on Hamming distances, where we allow a fixed number of clusters $g$ to match the available memory. We employ the temporal order of video frames and calculate Hamming distances only with neighboring frames. Figure 3.3 illustrates the two temporal clustering algorithms we consider here: *cumulative clustering* and *slope clustering*. We start by calculating the cumulative Hamming distance $\mathscr{C}(\mathbf{x})$ for neighboring frames along the temporal order:

$$\mathscr{C}_N(\mathbf{x}) = \sum_{i=1}^{N-1} H(\mathbf{x}_i, \mathbf{x}_{i+1}), \tag{3.8}$$

where $\mathbf{x}_i$ is the binarized activation of frame $i$, $H(\cdot,\cdot)$ is the Hamming distance, and $N$ is the total number of frames. For *cumulative clustering*, we divide the total cumulative Hamming distance, $\mathscr{C}(\mathbf{x})$, into $g$ even segments, where the cluster id for frame $i$ is $\lceil g\frac{\mathscr{C}_i(\mathbf{x})}{\mathscr{C}_N(\mathbf{x})}\rceil$. For the *slope clustering*, the boundaries of the segments are defined by the frame indexes corresponding to the top-$g$ largest slopes where the cumulative distance increases the most.

For efficiency, we cluster early on in the network, and input to the subsequent layers only aggregated activations. We assume that the sign of the activations corresponding to two similar frames, approximately agree throughout the network. To validate this, we visualize in Figure 3.4 the Hamming distance over activations corresponding to similar and dissimilar frames. The distances corresponding to similar frames remain consistent across different layers.

Putting everything together, we input a set of $n$ videos into our TSM-based [4] network architecture. After the first block, we apply temporal clustering and average the activations within each cluster, giving rise to $g$ activations per video. These aggregated activations are input to the subsequent blocks of the network. Our method efficiently utilizes all frames for training and it is end-to-end trainable, as the gradients propagate directly through the aggregated feature-maps. Figure 3.2 depicts the outline of our method.

## 3.4. EXPERIMENTS

We evaluate our method on the action recognition datasets Something-Something V1 & V2 [42], UCF-101 [43], HMDB51 [44] and Breakfast [41]. The consistent improvements show the effectiveness and generality of our method. We validate and analyze our method on a fully controlled Move4MNIST dataset we created. We also include ablation studies of the components of our method.

**Datasets**. Something-Something V1 [42] consists of 86k training videos and 11k validation videos belonging to 174 action categories. The second release V2 of Something-Something increase the number of videos to 220k. The UCF101 [43] dataset contains 101 action classes and 13,320 video clips. The HMDB51 [44] dataset is a varied collection of movies and web videos with 6,766 video clips from 51 action categories. Breakfast [41] has long videos of human cooking activities with 10 categories with 1,712 videos in total, with 1,357 for training and 335 for testing. Our fully controlled dataset Move4MNIST has four action classes *{move up, move down, move left, move right}*, and 1,800 videos for training and 600 videos for testing. Each video has 32 frames, with a digit from MNIST [45] moving on a UCF-101 video background. To obtain a per-frame ground truth of which frames are relevant we randomly inserted a consecutive chunk of UCF-101 background frames, black frames and frames with MNIST digits that are not part of the target classes. An example is shown in Figure 3.7.

**Training & Inference**. Following the setting in TSM [4], our models are fine-tuned from Kinetics [46] pre-trained weights and we freeze the Batch Normalization [47] layers for HMDB51 [44] and UCF101 [43] datasets. For other datasets, our models are fine-tuned from ImageNet [11] pre-trained weights. To optimize the GPU we train with a fixed number of frames per batch. If the video has less frames, we pad it repeatedly with the last video frame. We compare and cluster the activations of all the frames in each video, and get $g$ average activations for each video, from the first block of our model. We set the number of clusters to $g = \{8, 16\}$ to align with the sub-sampling methods using 8 or 16 frames. During testing, we follow the setting of TSM and sample one clip per video and use the full resolution image with the shorter side 256.

**Backbone architecture.** For a fair comparison with the state-of-the-art, we evaluate our method on the TSM [4] backbone replying on the ResNet-50 [48] architecture. We use TSM with a ResNet-18 as the backbone for the experiments on our toy dataset Move4MNIST and for model analysis on the Breakfast dataset.

### 3.4.1. ARE MORE FRAMES BETTER?

To make it computationally possible to use all individual frames we evaluate on the fully controlled Move4MNIST to test if using more frames during training is better than sub-sampling. We use here the ResNet-18 [48] backbone pretrained on ImageNet [11] and compare with TSM [4]. We evalute slope clustering and cumulative clustering, and a cluster-free uniform grouping of evenly distributed segments and then aggregating them (*Ours-uniform*).

Table 3.1 shows that TSM trained on all the 32 frames of a video in Move4MNIST significantly outperforms TSM trained on 8 and 16 sub-sampled frames. The uniform grouping of evenly distributed segments does not much better than random sub-sampling,

| Model | #Frames | #Clusters | FLOPs /Video | Runtime Mem./Video | Top-1 |
|---|---|---|---|---|---|
| TSM | 8 | - | 14.56G | 1.04GB | 90.13 ± 0.38 |
| TSM | 16 | - | 29.12G | 1.72GB | 93.78 ± 0.33 |
| TSM | all | - | 58.24G | 3.15GB | **98.83 ± 0.16** |
| Ours-uniform | all | 8 | 28.61G | 1.56GB | 90.25 ± 0.28 |
| Ours-slope | all | 8 | 28.61G | 1.56GB | 93.33 ± 0.19 |
| Ours-cumulative | all | 8 | 28.61G | 1.56GB | **94.08 ± 0.25** |
| Ours-uniform | all | 16 | 38.51G | 1.79GB | 92.73 ± 0.25 |
| Ours-slope | all | 16 | 38.51G | 1.79GB | 94.06 ± 0.18 |
| Ours-cumulative | all | 16 | 38.51G | 1.79GB | **95.27 ± 0.21** |

Tab. 3.1: Training with all frames gives best accuracy. Our method with slope or cumulative clustering outperforms the uniform grouping of evenly distributed segments and frame sub-sampling. Our method has less FLOPs and runtime memory usage than TSM training with all frames.

and uniform grouping performs worse than random sub-sampling when the frame and cluster numbers increased from 8 to 16. This can be explained since the videos in the Move4MNIST contain black frames, UCF-101 background frames, and frames containing another digits at random positions, which can make sub-sampling miss frames related to the task and evenly distributed segments group frames wrongly. Both our clustering approaches with 8 and 16 clusters do better than evenly distributed segments or sub-sampling with 8 or 16 frames as they can adapt to the content and dynamically choose which frames to group. In addition, our method has significantly reduced FLOPs and runtime memory when compared to the baseline on all frames.

### 3.4.2. Do similar frames have similar gradients?

In this experiment, we evaluate our assumption that similar frame activations have similar gradients. The activations and gradients are taken from the 1st block of ResNet-18. We show the Euclidean and the Hamming activation distance versus the gradient Euclidean distance between all $32 * 31/2 = 496$ frame pairs for three videos in Move4MNIST in Figure 3.5. For both the Euclidean distance and the Hamming distance the relation between activations and gradients is close to linear. It validates our assumption that frames having similar activations with respect to the task have similar gradients.

We compare the ground truth gradients when training truly on all frames to our efficient approximation. We use 16 clusters and compare our approximate gradients to the real gradients which are from 3rd block of ResNet-18 for a video in Move4MNIST. We compare the results of our method with cumulative clustering, slope clustering and uniform grouping. Results in Figure 3.6 show that compared to uniform grouping, cumulative clustering and slope clustering give smaller Euclidean distance between the single gradient from each cluster and the sum of gradients of frames in the corresponding cluster. And cumulative clustering gives even smaller gradients differences than slope

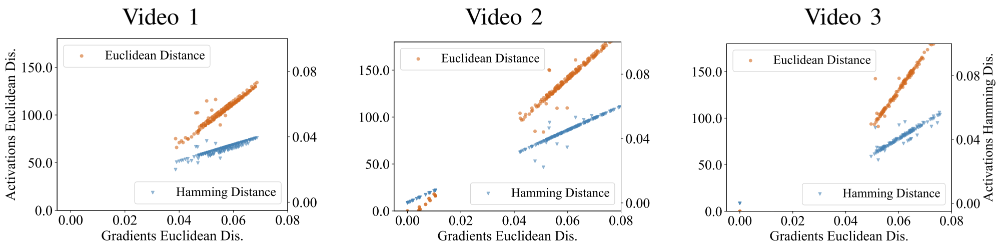


Fig. 3.5: An illustration of activation distance versus gradient distance for frames from three videos in Move4MNIST dataset. For frames that are similar with respect to recognizing the action, the activation distance and the gradients distance between them have a nearly linear relation for both the Euclidean distance and the Hamming distance. Our assumption that frames having similar activations with respect to the task have similar gradients is validated.

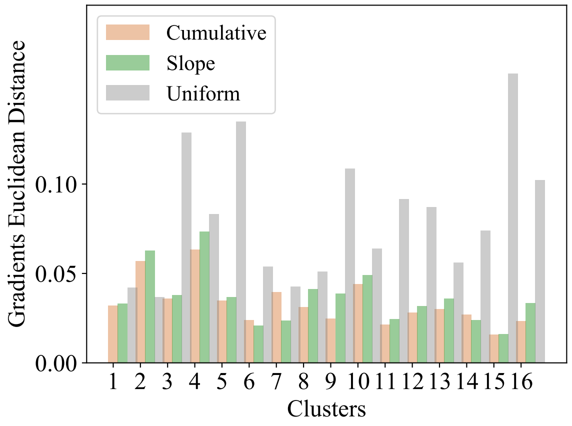


Fig. 3.6: Comparing the Euclidean distance between gradients of the ground truth of truly using all frames to our efficient approximation per cluster for cumulative clustering, slope clustering and uniform grouping on Move4MNIST. Compared to uniform grouping and slope clustering, cumulative clustering results in smaller gradients difference and thus a better approximation.

clustering. In other words, it means that our method with cumulative clustering (the right hand side of Equation (3.1)) approximates the standard gradients calculation (the left hand side of Equation (3.1)) in the network with a small difference.

### 3.4.3. ANALYZING MODEL PROPERTIES

We evaluate the clustering methods, the number of clusters, and the training time efficiency on Breakfast and Move4MNIST with a ResNet-18 backbone.

**Different temporal clustering methods.** We compare slope clustering, cumulative

| Model | #Frames | #Clusters | Tr. sec/epoch | Top-1 |
|---|---|---|---|---|
| TSM | 8 | - | 97.6 | 59.1 |
| TSM | 16 | - | 113.7 | 61.4 |
| Ours-uniform | all | 8 | 100.1 | 58.3 |
| Ours-slope | all | 8 | 99.6 | 60.7 |
| Ours-cumulative | all | 8 | 101.3 | 62.0 |
| Ours-uniform | all | 16 | 114.0 | 60.2 |
| Ours-slope | all | 16 | 114.5 | 63.7 |
| Ours-cumulative | all | 16 | 115.2 | **64.4** |

Tab. 3.2: With 8 and 16 clusters we consistently outperform TSM with 8 and 16 frames for comparable training time on the Breakfast dataset.

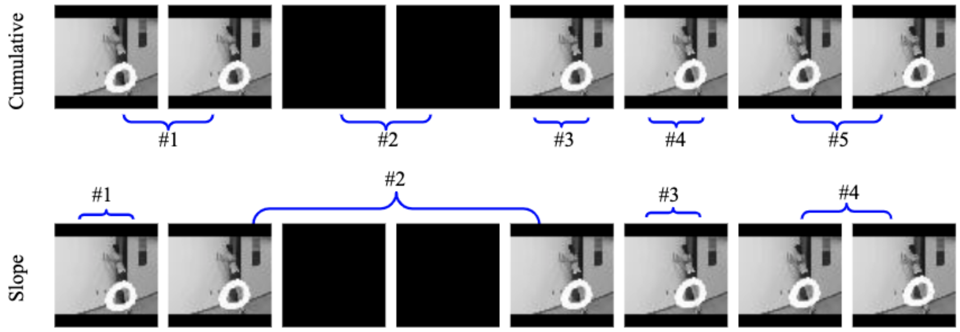


Fig. 3.7: Temporal clustering results for a video in Move4MNIST. Cumulative temporal clustering groups frames more accurately than slope temporal clustering.

clustering, and uniform grouping where the videos are split into equal segments. From Table 3.2, cumulative clustering outperforms slope clustering, while uniform grouping has the lowest top-1 accuracy. This is because equal temporal grouping merges non similar frames together leading to linear approximations of non-linear information and incorrect network updates, resulting in a low action recognition accuracy. A similar trend is also visible on the Move4MNIST dataset in Table 3.1. In Figure 3.7, we show the temporal clustering results for a small number of frames of a Move4MNIST video. Cumulative clustering correctly groups similar frames together, while slope clustering groups moving zero frames and black frames together.

**Number of clusters.** We conduct experiments using 8 and 16 clusters for our method, which follows the protocol of TSM with 8 and 16 frames for training. Table 3.2 shows that using 16 clusters consistently outperforms using 8 clusters for all clustering methods. A larger number of clusters improves accuracy. In the extreme case, the cluster numbers equal the number of frames in a video, which is equivalent with using all frames for training. From the table we can also see that our cumulative temporal clustering implementation improves the top-1 accuracy by 2.9% and 3.0%, separately for

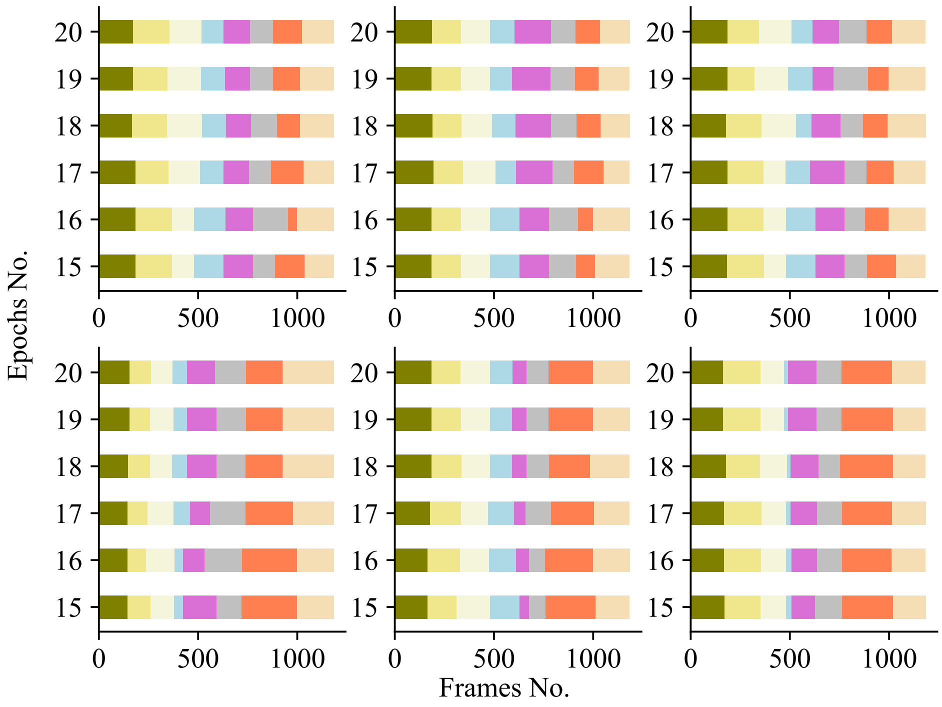


Fig. 3.8: Cumulative temporal clustering results over epochs for six videos in the Breakfast dataset. Each cluster is shown in a different color. Clusters contains segments with different lengths. Our cumulative temporal clustering groups frames with similar activations together. The cluster lengths change according to the changes in the frame activations during training.

8 clusters and 16 clusters comparing to TSM with 8 and 16 frames.

To show that our cumulative temporal clustering algorithm is different from the naive uniform grouping, we visualize the 8 clusters obtained from cumulative temporal clustering for six videos over different epochs in the Breakfast dataset in Figure 3.8. Different videos have different segment lengths in the cumulative temporal clustering, which takes the similarity of frame activations into consideration. In Figure 3.8, we also show that the cluster length changes over epochs during training, since the activations change during training.

**Efficiency of training time.** Table 3.2 gives the training time per epoch for all the models. Our method with 8 clusters and 16 clusters only has an increase of 3.7 seconds and 1.5 seconds in training time per epoch, when compared to TSM with 8 frames and 16 frames. The results show that our method is efficient during training time, while using all video frames.

| Model | Backbone | #Frames | #Clusters | Top-1 |
|---|---|---|---|---|
| ResNet-152[49] | ResNet152 | 64 | - | 41.1% |
| ActionVLAD [49] | ResNet152 | 64 | - | 55.5% |
| VideoGraph [49] | ResNet152 | 64 | - | 59.1% |
| TSM [4] (our impl.) | ResNet50 | 16 | - | 72.1% |
| Ours-slope | ResNet50 | all | 16 | 74.9% |
| Ours-cumulative | ResNet50 | all | 16 | **76.6%** |

Tab. 3.3: Our method using either slope temporal clustering or cumulative temporal clustering compared to existing works on the Breakfast dataset. Our proposal outperforms TSM, and significantly exceeds in top-1 accuracy methods using the deeper backbone architecture, ResNet-152. By using all frames our method has an advantage on long-term video action recognition.

| Model | Backbone | Pre-train | #Frames | #Clusters | Top-1 | |
|---|---|---|---|---|---|---|
| | | | | | UCF-101 | HMDB51 |
| TSM [4] (our impl.) | ResNet50 | Kinetics | 1 | - | 91.2% | 65.1% |
| TSN [4] | ResNet50 | Kinetics | 8 | - | 91.7% | 64.7% |
| SI+DI+OF +DOF [34] | ResNeXt50 | Imagenet | dynamic images | - | 95.0% | 71.5% |
| TSM [4] | ResNet50 | Kinetics | 8 | - | 95.9% | 73.5% |
| STM [50] | ResNet50 | ImageNet +Kinetics | 16 | - | 96.2% | 72.2% |
| Ours-slope | TSM-ResNet50 | Kinetics | all | 8 | 96.2% | 73.3% |
| Ours-cumulative | TSM-ResNet50 | Kinetics | all | 8 | **96.4%** | **73.4%** |

Tab. 3.4: Top-1 accuracy on UCF-101 and HMDB51. Our method performs only slightly better than the state-of-the-art on the scene-related datasets UCF-101 and HMDB51. These datasets do not have much frame diversity per video, thus, the improvement of our method over sampling methods is limited.

### 3.4.4. Comparison with the state-of-the-art

We compare our method with the state-of-the-art on Something-Something V1&V2, Breakfast, UCF-101 and HMDB51. All methods use ResNet-50 pre-trained on ImageNet as a backbone, unless specified otherwise.

**Comparison on the Breakfast dataset.** We compare our method with existing work on the Breakfast dataset, which contains long action videos. Our method using either slope temporal clustering or cumulative temporal clustering largely outperforms the three methods using ResNet-152 as a backbones, in Table 3.3. Compared to TSM using 16 sub-sampled frames, our method improves the top-1 accuracy by 2.8% and 4.5% with slope temporal clustering and cumulative temporal clustering, respectively. Methods using sub-sampling can easily miss important frames for the recognition task on long action videos. Our method has an advantage on the long videos for action recognition,

| Model | #Frames | #Clusters | Top-1 V1 | Top-1 V2 |
|---|---|---|---|---|
| TSN [4] | 8 | - | 19.7% | 30.0% |
| TRN-Multiscale [4] | 8 | - | 38.9% | 48.8% |
| TSM [4] | 8 | - | 45.6% | 59.1% |
| TSM [4] | 16 | - | 47.2% | 63.4% |
| STM [50] | 8 | - | 49.2% | 62.3% |
| STM [50] | 16 | - | 50.7% | 64.2% |
| Ours-slope | all | 8 | 46.7% | 60.2% |
| Ours-cumulative | all | 8 | 49.5% | 62.7% |
| Ours-cumulative | all | 16 | **51.4%** | **65.1%** |

Tab. 3.5: Top-1 accuracy on Something-Something V1 and V2 datasets. Our method using cumulative temporal clustering outperforms the state-of-the-art methods on both Something-Something V1 and V2. Our method achieves limited accuracy improvement for shorter videos.

by efficiently utilizing all the frames.

**Comparison on the Something-Something dataset.** In Table 3.5, we list the results of our method compared to other methods on the Something-Something V1&V2 datasets. We achieve state-of-the-art performance on both V1 and V2, with outperforming STM of 8 frames by 0.3% and 0.4% for V1 and V2, and STM of 16 frames by 0.7% and 0.9% for V1 and V2 respectively. Comparing to TSM, we significantly improve the top-1 accuracy of 8 frames by 3.9% and 3.5%, and the top-1 accuracy of 16 frames by 4.2% and 1.7% for the V1 and V2 datasets. Although the Something-Something dataset is characterized by temporal variations, the video clips are short compared to the Breakfast dataset. The methods using frame sampling heuristics can capture the main movement in videos. Therefore, our accuracy improvement is not as pronounced as for the Breakfast dataset.

**Comparison on the UCF-101 and HMDB51 datasets.** We train with 8 clusters and evaluate over three splits and report averaged results in Table 3.4. Our performance is on par with state-of-the-art methods on both datasets. The UCF-101 and HMDB51 have a scene-bias, where motion plays a limited role and just a few number of frames –or even a single frame– is sufficient. Thus, methods relying on sampling heuristics can correctly classify the actions and our method using all frames is not expected to improve results. To test this, we show results with a single frame in Table 3.4 which shows that TSM with 1 frame achieves comparable accuracy to TSN with 8 frames on UCF-101 and outperforms TSN with 8 frames on HMDB51. For scene-biased datasets, using all frames does not bring accuracy benefits.

## 3.5. CONCLUSION

We propose an efficient method for training action recognition deep networks without relying on sampling heuristics. Our work offers a solution to using all video frames during training based on the assumption that similar frames have similar gradients, leading to similar parameter updates. To this end, we efficiently find frames that are similar with respect to the classification task, by using a cumulative temporal clustering algorithm based on Hamming distances. The clustering based on Hamming distances enforces that activations in a cluster agree in signs, which is a requirement entailed by our assumption that we can approximate the gradients of multiple frames with a single gradient of an aggregated frame. We accumulate the activations within each cluster to create new representations used to classify the actions. Our proposed method shows competitive results on large datasets when compared to existing work.

Despite our state of the art results, we identify several limitations. One limitation is that the number of clusters is fixed and thus not well-suited for inhomogeneous videos with more semantic (shot) changes than clusters. This could create a dependency for action proposals or other approaches to pre-segment a video in homogeneous segments which somewhat counters the philosophy of using full video action recognition. Another limitation is that for grouping frames the only non-linearity we consider is the activation function and do not use the non-linearity in the loss. This limitation seems insurmountable, as memory constraints prevent us to store all frame activations for when the loss is computed. Nevertheless, with our current results and analysis, we make a first move for action recognition to go full video.

## APPENDIX

In addition to the comparison with 2D models, we also show results of our method compared to the state-of-the-art 3D models and additional 2D models on Breakfast, Something-Something V1 & V2, UCF-101 and HMDB51. [Nx] denotes the new citations in the tables.

### COMPARISON ON THE BREAKFAST DATASET.

| **Model** | **Backbone** | **#3D** | **#Optical flow** | **#Frames** | **#Clusters** | **Top-1** |
|---|---|---|---|---|---|---|
| ResNet152[49] | ResNet152 | - | - | 64 | - | 41.1% |
| ActionVLAD [49] | ResNet152 | - | - | 64 | - | 55.5% |
| VideoGraph [49] | ResNet152 | - | - | 64 | - | 59.1% |
| TSM [4] (our impl.) | ResNet50 | - | - | 16 | - | 72.1% |
| I3D [49] | 3D Inception-v1 | ✓ | - | 512 | - | 58.6% |
| I3D + ActionVLAD [49] | 3D Inception-v1 | ✓ | - | 512 | - | 65.5% |
| I3D + VideoGraph [49] | 3D Inception-v1 | ✓ | - | 512 | - | 69.5% |
| 3D ResNet-50 + Timeception [51] | 3D ResNet-50 | ✓ | - | 512 | - | 71.3% |
| Ours-slope | ResNet50 | - | - | all | 16 | 74.9% |
| Ours-cumulative | ResNet50 | - | - | all | 16 | **76.6%** |

Tab. 3.6: Our method using either slope temporal clustering or cumulative temporal clustering compared to existing works on the Breakfast dataset. Our proposal outperforms TSM and the 3D model, and significantly exceeds in top-1 accuracy methods using the deeper backbone architecture, ResNet-152. By using all frames our method has an advantage on long-term video action recognition.

### COMPARISON ON THE SOMETHING-SOMETHING DATASET.

| **Model** | **Backbone** | **#3D** | **#Optical flow** | **#Frames** | **#Clusters** | **Top-1 V1** | **Top-1 V2** |
|---|---|---|---|---|---|---|---|
| TSN [4] | ResNet50 | - | - | 8 | - | 19.7% | 30.0% |
| TRN-Multiscale [4] | ResNet50 | - | - | 8 | - | 38.9% | 48.8% |
| TSM [4] | Resnet50 | - | - | 8 | - | 45.6% | 59.1% |
| STM [50] | ResNet50 | - | - | 8 | - | 49.2% | 62.3% |
| MSNet-R50 [52] | TSM-ResNet50 | - | - | 8 | - | **50.9%** | **63.0%** |
| I3D [53] | I3D | ✓ | - | 32 | - | 41.6% | - |
| NL-I3D [53] | I3D | ✓ | - | 32 | - | 44.4% | - |
| NL-I3D+GCN [53] | I3D | ✓ | - | 32 | - | 46.1% | - |
| S3D-G [54] | Inception | ✓ | - | 64 | - | 48.2% | - |
| ECO [2] | BNIncep+3D Res18 | ✓ | - | 8 | - | 39.6% | - |
| ECO [2] | BNIncep+3D Res18 | ✓ | - | 16 | - | 41.4% | - |
| ECO-En *Lite* [2] | BNIncep+3D Res18 | ✓ | - | 92 | - | 46.4% | - |
| ECO-En *Lite*-RGB+Flow [2] | BNIncep+3D Res18 | ✓ | ✓ | 92+92 | - | 49.5% | - |
| DFB-Net [55] | 3D ResNet50 | ✓ | - | 16 | - | 50.1% | - |
| Ours-slope | TSM-ResNet50 | - | - | all | 8 | 46.7% | 60.2% |
| Ours-cumulative | TSM-ResNet50 | - | - | all | 8 | 49.5% | 62.7% |

Tab. 3.7: Top-1 accuracy on Something-Something V1 and V2 datasets. Our method using cumulative temporal clustering outperforms most state-of-the-art methods on both Something-Something V1 and V2, and performs on par with ECO-En *Lite* with both RGB and optical flow while slightly worse than MSNet-R50. Our method achieves limited accuracy improvement for shorter videos.

## Comparison on the UCF-101 and HMDB51 dataset.

| Model | Backbone | Pre-train | #3D | #Optical flow | #Frames | #Clusters | Top-1 UCF-101 | Top-1 HMDB51 |
|---|---|---|---|---|---|---|---|---|
| TSM [4] (our impl.) | ResNet50 | Kinetics | - | - | 1 | - | 91.2% | 65.1% |
| TSN [4] | ResNet50 | Kinetics | - | - | 8 | - | 91.7% | 64.7% |
| SI+DI+OF+DOF [34] | ResNeXt50 | Imagenet | - | ✓ | dynamic images | - | 95.0% | 71.5% |
| TSM [4] | ResNet50 | Kinetics | - | - | 8 | - | 95.9% | 73.5% |
| STM [50] | ResNet50 | ImageNet +Kinetics | - | - | 16 | - | 96.2% | 72.2% |
| MSNet-R50 [52] | TSM-ResNet50 | Kinetics | - | - | 8 | - | - | 75.8% |
| ECO-En *Lite* [2] | BNIncep+3D Res18 | Kinetics | ✓ | - | 8 | - | 94.8% | 72.4% |
| RGB I3D [1] | 3D Inception-v1 | Kinetics | ✓ | - | 64 | - | 95.1% | 74.3% |
| Two-stream I3D [1] | 3D Inception-v1 | Kinetics | ✓ | ✓ | 64+64 | - | **97.8%** | **80.9%** |
| Ours-slope | TSM-ResNet50 | Kinetics | - | - | all | 8 | 96.2% | 73.3% |
| Ours-cumulative | TSM-ResNet50 | Kinetics | - | - | all | 8 | 96.4% | 73.4% |

Tab. 3.8: Top-1 accuracy on UCF-101 and HMDB51. Our method performs only slightly better than the state-of-the-art on the scene-related datasets UCF-101 and HMDB51, and worse than two-stream I3D, which uses both RGB and optical flow with 3D Inception-v1 backbone. Given that these datasets do not have a large number of frames per video, the improvement of our method over sampling methods is limited.

# REFERENCES

[1] J. Carreira and A. Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[2] M. Zolfaghari, K. Singh, and T. Brox. “Eco: Efficient convolutional network for online video understanding”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

[3] B. Korbar, D. Tran, and L. Torresani. “Scsampler: Sampling salient clips from video for efficient action recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6232–6242.

[4] J. Lin, C. Gan, and S. Han. “Tsm: Temporal shift module for efficient video understanding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7083–7093.

[5] J. Ren, X. Shen, Z. Lin, and R. Mech. “Best Frame Selection in a Short Video”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 3212–3221.

[6] S. Sudhakaran, S. Escalera, and O. Lanz. “Gate-Shift Networks for Video Action Recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1102–1111.

[7] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. “Temporal segment networks: Towards good practices for deep action recognition”. In: *European conference on computer vision*. Springer. 2016, pp. 20–36.

[8] W. Wu, D. He, X. Tan, S. Chen, and S. Wen. “Multi-Agent Reinforcement Learning Based Frame Sampling for Effective Untrimmed Video Recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6222–6231.

[9] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis. “AdaFrame: Adaptive Frame Selection for Fast Video Recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[10] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. “End-to-end learning of action detection from frame glimpses in videos”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2678–2687.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, pp. 248–255.

[12] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. “Reversible Architectures for Arbitrarily Deep Residual Neural Networks”. In: *AAAI*. 2018.

[13] T. Chen, B. Xu, C. Zhang, and C. Guestrin. “Training deep nets with sublinear memory cost”. In: *arXiv preprint arXiv:1604.06174* (2016).

[14] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. “The reversible residual network: Backpropagation without storing activations”. In: *Advances in neural information processing systems*. 2017, pp. 2214–2224.

[15] C. Feichtenhofer, A. Pinz, and A. Zisserman. “Convolutional two-stream network fusion for video action recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1933–1941.

[16] H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. “Two stream lstm: A deep fusion framework for human action recognition”. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 177–186.

[17] K. Simonyan and A. Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems* 27 (2014).

[18] T. Du, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. “C3d: Generic features for video analysis”. In: *Corr* 2.8 (2014).

[19] K. Hara, H. Kataoka, and Y. Satoh. “Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?” In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6546–6555.

[20] S. Ji, W. Xu, M. Yang, and K. Yu. “3D convolutional neural networks for human action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[21] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. “A multi-stream bi-directional recurrent neural network for fine-grained action detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1961–1970.

[22] V. Veeriah, N. Zhuang, and G.-J. Qi. “Differential recurrent neural networks for action recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4041–4049.

[23] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik. “Action recognition in video sequences using deep bi-directional LSTM with CNN features”. In: *IEEE Access* 6 (2017), pp. 1155–1166.

[24] Z. Wu, C. Xiong, Y.-G. Jiang, and L. S. Davis. “Liteeval: A coarse-to-fine framework for resource efficient video recognition”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 7780–7789.

[25] M. Jain, J. C. Van Gemert, and C. G. Snoek. “What do 15,000 object categories tell us about classifying and localizing actions?” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 46–55.

[26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. “Large-scale Video Classification with Convolutional Neural Networks”. In: *CVPR*. 2014.

[27] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. “Temporal relational reasoning in videos”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 803–818.

[28] C. Feichtenhofer, H. Fan, J. Malik, and K. He. “Slowfast networks for video recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2019, pp. 6202–6211.

[29] K. Schindler and L. Van Gool. “Action snippets: How many frames does human action recognition require?” In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8.

[30] Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, and Z. Wang. “E2-train: Training state-of-the-art cnns with over 80% energy savings”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[31] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl. “A Multigrid Method for Efficiently Training Video Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 153–162.

[32] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. “Actionvlad: Learning spatio-temporal aggregation for action classification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 971–980.

[33] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. “Rank pooling for action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.4 (2016), pp. 773–787.

[34] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi. “Action Recognition with Dynamic Image Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12 (2018), pp. 2799–2813. DOI: 10.1109/TPAMI.2017.2769085.

[35] F. Sener, D. Singhania, and A. Yao. “Temporal Aggregate Representations for Long-Range Video Understanding”. In: *European Conference on Computer Vision*. 2020, pp. 154–171.

[36] S. Song, N.-M. Cheung, V. Chandrasekhar, and B. Mandal. “Deep adaptive temporal pooling for activity recognition”. In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, pp. 1829–1837.

[37] M. Malinowski, G. Swirszcz, J. Carreira, and V. Patraucean. “Sideways: Depth-Parallel Training of Video Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11834–11843.

[38] D. Oktay, N. McGreivy, J. Aduol, A. Beatson, and R. P. Adams. “Randomized Automatic Differentiation”. In: *CoRR* (2020).

[39] N. Goli and T. M. Aamodt. “ReSprop: Reuse Sparsified Backpropagation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1548–1558.

[40] S. Wiedemann, T. Mehari, K. Kepp, and W. Samek. “Dithered backprop: A sparse and quantized backpropagation algorithm for more efficient deep neural network training”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 720–721.

[41] H. Kuehne, A. B. Arslan, and T. Serre. “The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities”. In: *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*. 2014.

[42] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, *et al.* “The" Something Something" Video Database for Learning and Evaluating Visual Common Sense.” In: *ICCV*. Vol. 1. 4. 2017, p. 5.

[43] K. Soomro, A. R. Zamir, and M. Shah. “A dataset of 101 human action classes from videos in the wild”. In: *Center for Research in Computer Vision* 2.11 (2012).

[44] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. “HMDB: A large video database for human motion recognition”. In: *2011 International Conference on Computer Vision* (2011), pp. 2556–2563.

[45] Y. LeCun, C. Cortes, and C. Burges. “MNIST handwritten digit database”. In: (2010).

[46] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.* “The Kinetics Human Action Video Dataset”. In: *CoRR* (2017).

[47] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[48] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[49] N. Hussein, E. Gavves, and A. W. Smeulders. “Videograph: Recognizing minutes-long human activities in videos”. In: *ICCV 2019, Workshop on Scene Graph Representation and Learning* (2019).

[50] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan. “Stm: Spatiotemporal and motion encoding for action recognition”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2000–2009.

[51] N. Hussein, E. Gavves, and A. W. Smeulders. “Timeception for complex action recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 254–263.

[52] H. Kwon, M. Kim, S. Kwak, and M. Cho. “MotionSqueeze: Neural Motion Feature Learning for Video Understanding”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 345–362.

[53] X. Wang and A. Gupta. “Videos as Space-Time Region Graphs”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[54] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. “Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[55] B. Martinez, D. Modolo, Y. Xiong, and J. Tighe. “Action Recognition With Spatial-Temporal Discriminative Filter Banks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

# 4

# Cross domain image matching in presence of outliers

*Cross domain image matching between image collections from different source and target domains is challenging in times of deep learning due to i) limited variation of image conditions in a training set, ii) lack of paired-image labels during training, iii) the existing of outliers that makes image matching domains not fully overlap. To this end, we propose an end-to-end architecture that can match cross domain images without labels in the target domain and handle non-overlapping domains by outlier detection. We leverage domain adaptation and triplet constraints for training a network capable of learning domain invariant and identity distinguishable representations, and iteratively detecting the outliers with an entropy loss and our proposed weighted MK-MMD. Extensive experimental evidence on Office [1] dataset and our proposed datasets Shape, Pitts-CycleGAN shows that the proposed approach yields state-of-the-art cross domain image matching and outlier detection performance on different benchmarks. The code will be made publicly available.*

This chapter is published as:
**Xin Liu**, Seyran Khademi, Jan Van Gemert. *Cross Domain Image Matching in Presence of Outliers*. International Conference on Computer Vision Workshops (ICCV-w), 2019. arXiv: `https://arxiv.org/abs/1909.03552`

## 4.1. Introduction

Cross domain image matching is about matching two images that are collected from different sources (e.g. photos of the same location but captured in different illuminations, seasons or era). It has wide application value in different areas, with research in location recognition over large time lags [2], e-commerce product image retrieval [3], urban environment image matching for geo-localization [4], etc.

Even using deep feature representation learning, the automated cross domain image matching task remains challenging mainly due to the following difficulties. First, it is difficult to match varying observations of the same location or object, in general. Second, often the paired-image examples from two domains are not available for training neural networks. Third, the image samples in two domains may not fully overlap due to the existing of outlier images, which affects the matching performance if such outliers are not detected.

In this work, we address the problem of domain adaptation for feature learning in a cross domain matching task when outliers are present. As is common in domain adaptation, we only have labeled image pairs from the source domain, but no labels from the target domain. To resolve the domain disparity between the train and the test data, we are inspired from Siamese network [5] for image matching and domain adaptation used in image classification [6–10]. We propose a triplet constraints network to learn the domain invariant and identity distinguishable representations of the samples. This is made possible by utilizing the paired-image information from the source domain, a weighted multi-kernel maximum mean discrepancy (weighted MK-MMD) method and an entropy loss. The setting of the problem and experiment results of our method are depicted on a 2D toy dataset in fig. 4.1.

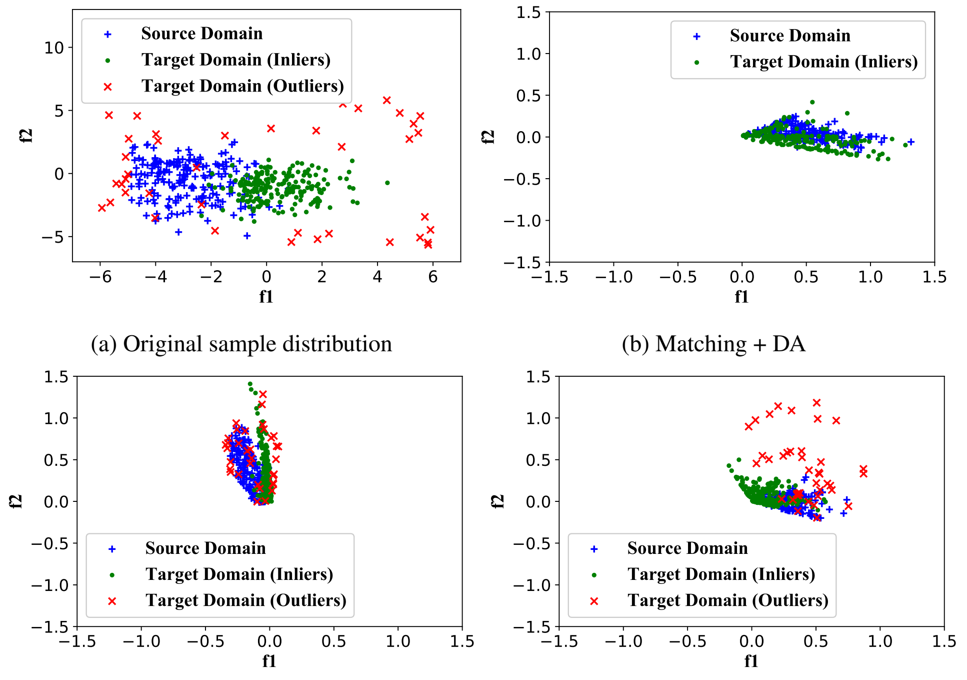
To verify our method, we introduce two new synthetic datasets as there are no publicly available datasets for our problem setting. Moreover, we believe outlier-aware algorithms are essential to design practical domain adaptation algorithms as many real data repositories contain irrelevant samples w.r.t. the source domain. In summary, our main contribution is two-fold:

- Joint domain adaptation and outlier detection.
- Two new datasets, *Pits-CycleGAN* dataset and *Shape* dataset, for cross domain image matching.

## 4.2. Related work

### 4.2.1. Image matching

Feature learning based matching methods become popular due to its improved performance over hand-crafted features (e.g. SIFT [11]). Siamese network architectures [5] are among the most popular feature learning networks, especially for pairs comparison tasks. We also adopt Siamese network as part of our framework. The purpose is to learn feature representations to distinguish matching and unmatching pairs in the source domain, which assists the network in learning to match cross domain images. In the cross-domain image matching context, Lin*et al.* [12] investigated a deep Siamese network to learn feature embedding for cross-view image geo-localization. Kong *et al.* [13] applied Siamese architecture to cross domain footprint matching. Tian *et al.* [4] utilized Siamese network for matching the



(a) Original sample distribution (b) Matching + DA

(c) Matching + DA (d) Our method

Fig. 4.1: Domain adaptation (DA) and image matching applied on a 2D toy dataset generated with domain shift between source and target domains. (a) Original distribution, (b) no outliers, (c) with outliers, (d) our method. The result of (b) and (c) shows that outliers affect the alignment of source samples and inlier target samples. (c) and (d) show that our outlier detection helps separating the outliers from the aligned source samples and inlier target samples.

building images from street view and bird's eye view. Unlike the existing works on cross-domain image matching, we consider labeled paired-image information is only available in the source domain.

### 4.2.2. Domain adaptation

Domain adaptation have been researched over recent years in diverse domain classification tasks, in which adversarial learning and statistic methods are main approaches. Ganin *et al.* [14] proposed domain-adversarial training of neural networks with input of labeled source domain data and unlabeled target domain data for classification. In [10], the authors proposed a deep transfer network (DTN), which achieved domain transfer by simultaneously matching both the marginal and the conditional distributions with adopting the empirical maximum mean discrepancy (MMD) [15], which is a nonparametric metric. Venkateswara *et al.* [9] applied MK-MMD [16] to a deep learning framework that can learn hash codes for domain adaptive classification. In this setting MK-MMD loss promotes nonlinear alignment of data, which generates a nonparametric distance in Reproducing Kernel Hilbert Space (RKHS). The distance between two distributions is the distance between their means in a RKHS. When two data sets belong to the same distribution, their MK-MMD is zero. Based on the successful performance of MK-MMD loss, we also adopt it to adapt different domains, this time for image matching task. This requires the marriage of Siamese network with MK-MMD loss, as we do later in our paper.

### 4.2.3. Outlier detection

Much work exists on outlier detection [17–20]. Chalapathy *et al.* [17] proposed an one-class neural network (OC-NN) encoder-decoder model to detect anomalies. Sabokrou *et al.* [19] also applied the encoder-decoder architecture as part of their network for novelty detection. Zhang *et al.* [20] proposed an adversarial network for partial domain adaptation to deal with outlier classes in the source domain. Their network is for classification task, and they do not have the assumption that outliers originate from low-density distribution. Instead, we are inspired by the work of Liu *et al.* [18] which uses a kernel-based method to learn, jointly, a large margin one-class classifier and a soft label assignment for inliers and outliers. Using the soft label assignment, we implement outlier detection with cross domain image matching in an iterative sample reweighting way.

## 4.3. Domain adaptive image matching

### 4.3.1. Siamese loss

We introduce our proposal for domain adaptation for image matching task once labeled data is not available in the target domain. Let $X_s$ denote the source domain image set. A pair of images $x_i, x_j \in X_s$ are used as input to part of our network, as shown in Figure 4.2. $x_i, x_j$ can be a matching pair or an unmatching pair. The objective is to automatically learn a feature representation, $f(\cdot)$, that effectively maps the input $x_i, x_j$ to a feature space, in which matching pairs are close to each other and unmatching pairs are far apart. We employ the contrastive loss as introduced in [21]:

$$L(x_i, x_j, y) = \frac{1}{2} y D^2 + \frac{1}{2}(1-y)\{\max(0, m-D)\}^2, \tag{4.1}$$

where $y \in \{0,1\}$ indicates unmatching pairs with $y=0$ and matching pairs with $y=1$, $D$ is the Euclidean distance between the two feature vectors $f(x_i)$ and $f(x_j)$, and $m$ is the margin parameter acting as threshold to separate matching and unmatching pairs.

### 4.3.2. Domain adaptation loss

It is known that in deep CNNs, the feature representations transition from generic to task-specific as one goes up from bottom layers to other layers [22]. Compared to the convolution layers *conv1* to *conv5*, the fully connected layers are more task-specific and need to be adapted before they can be transferred [9].

Accordingly, our approach attempts to minimize the MK-MMD loss to reduce the domain disparity between the source and target feature representations for fully connected

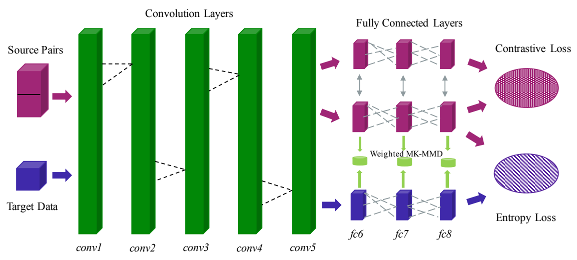


Fig. 4.2: The network for cross domain image matching and outlier detection. The contrastive loss makes the network to learn paired-image information from the source. The weighted MK-MMD loss trains the network to learn transferable features between the source and the inliers of the target. The entropy loss helps distinguish inliers and outliers in the target domain.

layers, $\mathscr{F} = \{fc6, fc7, fc8\}$. The multi-layer MK-MMD loss is given by,

$$\mathcal{M}(u_s, u_t) = \sum_{l \in \mathscr{F}} d_k^2(u_s^l, u_t^l), \tag{4.2}$$

where, $u_s^l = \{\boldsymbol{u}_i^{s,l}\}_{i=1}^{n_s}$ and $u_t^l = \{\boldsymbol{u}_i^{t,l}\}_{i=1}^{n_t}$ are the set of output representations for the source and target data at layer $l$, $\boldsymbol{u}_i^{*,l}$ is the output representation of inuput image $\boldsymbol{x}_i^{*,l}$ for the $l^{th}$ layer. The MK-MMD measure $d_k^2(\cdot)$ is the multi-kernel maximum mean discrepancy between the source and target representations [16]. For a nonlinear mapping $\phi(\cdot)$ associated with a reproducing kernel Hilbert space $\mathscr{H}_k$ and kernel $k(\cdot)$, where $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}, \boldsymbol{y}) \rangle$, the MK-MMD is defined as,

$$d_k^2(u_s^l, u_t^l) = ||\mathrm{E}[\phi(\boldsymbol{u}^{s,l})] - \mathrm{E}[\phi(\boldsymbol{u}^{t,l})]||_{\mathscr{H}_k}. \tag{4.3}$$

The characteristic kernel $k(\cdot)$, is determined as a convex combination of $\kappa$ PSD kernels, $\{k_m\}_{m=1}^{\kappa}$, $K := \{k : k = \sum_{m=1}^{\kappa} \beta_m k_m, \sum_{m=1}^{\kappa} \beta_m = 1, \beta_m \geq 0, \forall m\}$. In particular, we follow [23] and set the kernel weights as $\beta_m = 1/\kappa$ .

## 4.4. PROPOSED METHOD: OUTLIER-AWARE DOMAIN ADAPTIVE MATCHING

The task is to match images with the same content but from different domains where the outliers are present in the target domain. We assume that in the source domain there are sufficient labeled image pairs and in the target domain low-density outliers are present. As in conventional domain adaptation setting labeled data is not available in the target domain. We propose a deep triplet network which is comprised of three instances of the same feed-forward network with shared parameters, as shown in Figure 4.2.

### 4.4.1. IMPORTANCE WEIGHTED DOMAIN ADAPTATION

In our implementation, the MK-MMD loss in subsection 4.3.2 is calculated over every batch of data points during the back-propagation. Let n (even) be the number of source data points $u_s := \{\boldsymbol{u}_i^s\}_{i=1}^n$ and the number of target data points $u_t := \{\boldsymbol{u}_i^t\}_{i=1}^n$ in the batch. Then, the MK-MMD can be defined over a set of 4 data points $\boldsymbol{z}_i = [\boldsymbol{u}_{2i-1}^s, \boldsymbol{u}_{2i}^s, \boldsymbol{u}_{2i-1}^t, \boldsymbol{u}_{2i}^t]$, $\forall i \in \{1, 2, ..., n/2\}$. Thus, the MK-MMD is given by,

$$d_k^2(u_s, u_t) = \sum_{m=1}^{\kappa} \beta_m \frac{1}{n/2} \sum_{i=1}^{n/2} h_m(\boldsymbol{z}_i), \tag{4.4}$$

where, $\kappa$ is the number of kernels and $\beta_m = 1/\kappa$ is the weight for each kernel. And we can expand $h_m(\cdot)$ as,

$$h_m(\boldsymbol{z}_i) = k_m(\boldsymbol{u}_{2i-1}^s, \boldsymbol{u}_{2i}^s) + k_m(\boldsymbol{u}_{2i-1}^t, \boldsymbol{u}_{2i}^t) - k_m(\boldsymbol{u}_{2i-1}^s, \boldsymbol{u}_{2i}^t) - k_m(\boldsymbol{u}_{2i}^s, \boldsymbol{u}_{2i-1}^t), \tag{4.5}$$

4

in which, the kernel is $k_m(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{||\boldsymbol{x}-\boldsymbol{y}||_2^2}{\sigma_m})$.

With equations 4.4 and 4.5, we can interpret that in the minimum calculation unit ($h_m(z_i)$), two target domain images contribute to MK-MMD loss calculation. When there are outliers in the target domain, we only want the inliers to contribute to the calculation, but not the outliers. Therefore, we could assign the target samples with weights $w_i$ as 1 for inliers, and 0 for outliers. Because we have no ground truth labels, we can only treat the weights as the probability of the target samples to be inliers. Hence, we can introduce the weighted MK-MMD as,

$$d_{w_k}^2(u_s, u_t) = \sum_{m=1}^{\kappa} \beta_m \frac{1}{n/2} \sum_{i=1}^{n/2} w_{2i-1} w_{2i} h_m(\boldsymbol{z}_i), \tag{4.6}$$

where, $w_{2i-1}$ and $w_{2i}$ are the weights of the target data points $\boldsymbol{u}_{2i-1}^t$ and $\boldsymbol{u}_{2i}^t$ in $h_m(z_i)$ respectively, and $w_{2i-1}, w_{2i} \in [0, 1]$. We will explain how to obtain the weight for each target domain sample in next subsection.

### 4.4.2. Outlier detection

Since the inlier-outlier label is not available, we implement an entropy loss to iteratively reassign target domain sample probability of being an inlier, which provides the weights for the weighted MK-MMD.

We use the similarity measure $\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle$ to learn discriminative inlier-outlier information for the target domain data. We define three classes of reference data $u_r$ for similarity measure, the source domain class $\boldsymbol{u}^1$, the pseudo inlier class $\boldsymbol{u}^2$ and the pseudo outlier class $\boldsymbol{u}^3$. An ideal target output $\boldsymbol{u}_i^t$ needs to be similar to many of the outputs from one of the classes, $\{\boldsymbol{u}_k^c\}_{k=1}^K$. We assume $K$ data points for every class $c$, where $c \in \{1, 2, 3\}$ and $\boldsymbol{u}_k^c$ is the $k^{th}$ output from class $c$. Then the probability measure for each target sample can be outlined as,

$$p_{ic} = \frac{\sum_{k=1}^K \exp(\boldsymbol{u}_i^{t\top} \boldsymbol{u}_k^c)}{\sum_{c=1}^C \sum_{k=1}^K \exp(\boldsymbol{u}_i^{t\top} \boldsymbol{u}_k^c)}, \tag{4.7}$$

where, $p_{ic}$ is the probability that a target domain sample $x_i^t$ is assigned to category $c$. When the sample output is similar to one category only, the probability vector $\boldsymbol{p}_i = [p_{i1}, ..., p_{ic}]^\top$ tends to be a one-hot vector. A one-hot vector can be viewed as a low entropy realization of

$\boldsymbol{p}_i$. Thus, we introduce a loss to capture the entropy of the probability vectors. The entropy loss can be given by,

$$S(u_r, u_t) = -\frac{1}{n_t}\sum_{i=1}^{n_t}\sum_{c=1}^{C} p_{ic} log(p_{ic}). \tag{4.8}$$

In subsection 4.4.1, we discussed the weighted MK-MMD loss with weights $w_{2i-1}$ and $w_{2i}$. With the sample probabilities of target domain data calculated from equation 4.7, the weights are calculated as,

$$w_i = \begin{cases} \frac{p_{i1}+p_{i2}}{p_{i1}+p_{i2}+p_{i3}} & \text{if } x_i^t \text{ is classified as source} \\ \frac{p_{i2}}{p_{i1}+p_{i2}+p_{i3}} & \text{if } x_i^t \text{ is classified as others} \end{cases}. \tag{4.9}$$

If a target domain sample is classified as "source", then it has a high probability of being an inlier, and therefore should contribute more to reducing the domain disparity. So we calculate the weight of such a target domain sample with the sum of $p_{i1}$ and $p_{i2}$.

**Algorithm** We iteratively update the target domain data weights after each epoch during training, which works together with domain adaptation for guiding and correcting the detection of outliers and inliers.

The proposed algorithm for outlier detection is showed in the following. The proposed

**Algorithm 1**

```
Input: source domain and target domain training data
Output: target domain training data probabilities
1: Initialization $i = 0$, calculate the average Euclidean distance of each target domain
   training sample between all the source domain training samples, sort the distances in
   ascending order and initialize target domain training samples' weights according to
   the sorted distances, $x_i \in$ first half: $w_i = 0.7$ (pseudo inlier class), $x_i \in$ second half:
   $w_i = 0.3$ (pseudo outlier class). Inlier class consists of source domain training data,
   which has the same number of samples with pseudo inlier and pseudo outlier classes.
2: Repeat:
3: $i = i + 1$
4: make new mini batches
5: minimize the overall loss function objective (4.10)
6: update the samples' weights by equation 4.7 and 4.9
7: update the sets of pseudo inlier class and pseudo outlier class
8: Until target samples' probabilities are unchanged or training time ends
```

method is built upon the intuitive assumption that outliers originate from low-density distribution. Thus, we can assume that the ratio of outliers to all the target domain data is no more than 50%.

## 4.4.3. Overall objective

We propose a model for cross domain image matching and outlier detection, which incorporates learning image matching information from source domain (4.1), weighted domain adaptation between the source and the target (4.6) and outlier detection (4.8) in a deep CNN. The overall objective is given by:

$$min_u J = L(u_s) + \gamma M_w(u_s, u_t) + \eta S(u_r, u_t), \tag{4.10}$$

where, $u := \{u_s \bigcup u_t\}$ and $(\gamma, \eta)$ control the importance of domain adaptation (4.6) and entropy loss (4.8) respectively.

## 4.5. EXPERIMENTS

### 4.5.1. DATASETS

There are no publicly available datasets for our task. Therefore, we propose two datasets for evaluation. Sample images from the three datasets are shown in Figure 4.3.

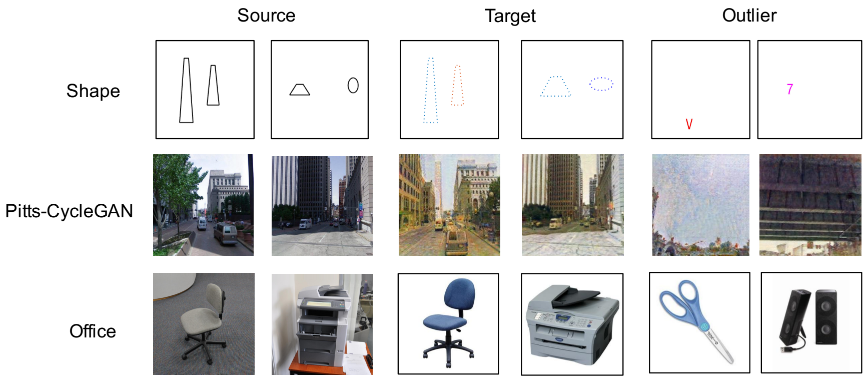


Fig. 4.3: Examples from *Shape*, *Pitts-CycleGAN* and *Office* sets.

**Shape** is one of the synthetic datasets we generate. It contains 60k source domain images, 30k target domain images (including 2800 outliers). The outlier images are made up of single alphabets or digits. The source domain and inlier images are combinations of two geometric shapes, drawn with black solid lines and colored dot lines, respectively. We define two images are a matching pair if the combination of shapes is the same.

**Pitts-CycleGAN** is the other synthetic dataset, which contains 204k Pittsburgh Google Street View images from Pittsburgh dataset [24] as the source domain, and 157k target domain images (including 52k outliers) generated by applying CycleGAN [25] to the Pittsburgh images. So the target domain images are in a painting style. The outliers are sky images or city views not containing any useful landmark information.

**Office** [1] consists of 3 domains, *Amazon, Dslr, Webcam*. We choose *Dslr* as source domain and *Amazon* as target domain. We make pairs with images from the same category. The outliers come from two randomly chosen categories ('speaker', 'scissors') out of the 31 categories.

### 4.5.2. IMPLEMENTATION DETAILS

For our triplet network, the three sub-networks share the same architecture and weights. Pre-trained AlexNet [26] is used for the sub-networks. We finetune the weights of *conv4 - conv5, fc6, fc7, fc8*. For the weighted MK-MMD, we use a Gaussian kernel with a bandwidth $\sigma$ given by the median of the pairwise distances in the training data. To incorporate the multi-kernel, we vary the bandwidth $\sigma_m \in [2^{-8}\sigma, 2^{8}\sigma]$ with multiplicative factor of 2 [9]. For performance evaluation, we sort the Euclidean distance between the query and all the gallery features (L2-normalized) to obtain the ranking result. Moreover, we employ the standard metric mean average precision (MAP).

### 4.5.3. BASELINE METHODS

There are no available baselines to directly compare with our method, thus, we separate our experiments to research on domain adaptive image matching 4.5.4 and effectiveness of outlier detection 4.5.5.

In the experiment on domain adaptive image matching, we assume no outliers exist in the target domain. Our method is to jointly learn the contrastive loss $L(u_s)$ and MK-MMD loss $M(u_s, u_t)$. It is trained with pairs from the source domain and images from the target domain, we call it ***SiameseDA***.

For evaluating the effectiveness of outlier detection, the target domain contains outliers. Our method is called ***DA+OutlierDetection***, which learns on the objective 4.10.

The baselines for each experiment are shown in Table 4.1.

| Baseline | Experiment |
|---|---|
| | **Domain adaptive image matching** |
| ***SIFT + Fisher Vector*** [11, 27] | trained on the source domain data |
| ***Siamese*** network [5] | trained on the source domain image pairs |
| | **Effectiveness of outlier detection** |
| ***SiameseDA*** (upper bound) | trained without outliers |
| ***SiameseDAOut*** (lower bound) | ***SiameseDA*** trained with outliers |

Tab. 4.1: Baseline methods for our experiments.

### 4.5.4. DOMAIN ADAPTIVE IMAGE MATCHING

In this section, we assume the target domain does not contain outliers. We explore if applying domain adaptation improves the performance of cross domain image matching. In this case, the learning objective is

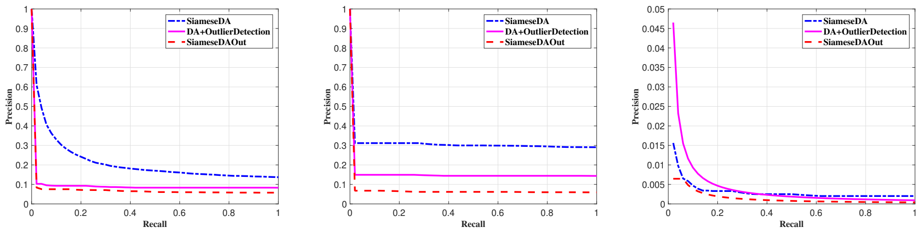
$$min_u J = L(u_s) + \gamma M(u_s, u_t), \tag{4.11}$$

where, the MK-MMD loss term $M(u_s, u_t)$ is the unweighted version as explained in subsection 4.3.2.

The MAP results are given in Table 4.2. Our method consistently outperforms the baselines across all the datasets. With applying MK-MMD loss for domain adaptation, the

| Method | Shape | | | Office | | | Pitts-CycleGAN | | |
|---|---|---|---|---|---|---|---|---|---|
| | $T \to S$ | $S \to S$ | $T \to T$ | $T \to S$ | $S \to S$ | $T \to T$ | $T \to S$ | $S \to S$ | $T \to T$ |
| SIFT + Fisher Vector | $2.5 \pm 0.4$ | $3.6 \pm 0.3$ | $3.4 \pm 0.3$ | $3.5 \pm 0.2$ | $12.0 \pm 0.5$ | $3.5 \pm 0.1$ | 0.04 | $0.8 \pm 0.05$ | $0.3 \pm 0.03$ |
| Siamese | $8.3 \pm 0.1$ | $95.0 \pm 0.2$ | $31.7 \pm 0.6$ | $10.7 \pm 0.5$ | $99.2 \pm 0.2$ | $77.2 \pm 0.3$ | $0.2 \pm 0.01$ | $81.3 \pm 0.3$ | $60.6 \pm 0.5$ |
| **SiameseDA** | **$26.4 \pm 0.2$** | $53.1 \pm 0.1$ | $46.2 \pm 0.1$ | **$29.1 \pm 0.1$** | $99.7 \pm 0.1$ | $77.5 \pm 0.2$ | **$0.4 \pm 0.01$** | $80.4 \pm 0.1$ | $59.5 \pm 0.1$ |

Tab. 4.2: MAP performance for cross domain image matching and in-domain image matching experiments on three datasets. $T$ means target domain, $S$ means source domain. $T \to S$ implies matching target domain images to source domain images, similar for $S \to S$, $T \to T$. Our method **SiameseDA** outperforms the baselines across all the datasets.



(a) Shape (b) Office (c) Pitts-CycleGAN

Fig. 4.4: Precision-Recall results of our method **DA+OutlierDetection**, SiameseDA and SiameseDAOut for the experiment of cross domain image matching with outlier detection on the three datasets. Our method gains over the lower bound method.

performance of matching $S \to S$ decreases comparing to that of Siamese method. This is within our expectation since the network may need to learn less from the source domain to be domain adaptive. Moreover, it is worth to notice that our method also improves the in-domain image matching ($T \to T$) of the target domain.

### 4.5.5. EFFECTIVENESS OF OUTLIER DETECTION

Here we assume the target domain contains outliers, which is to show if the presence of outliers reduces the accuracy of cross domain image matching, and our method could improve it.

The performance of our method (*DA+OutlierDetection*), upper bound (*SiameseDA*) and lower bound (*SiameseDAOut*) are given in Table 4.3. In terms of testing, we only take the classified inliers in the query set in calculation. From Table 4.3 we can see, our method outperforms the lower bound for all the three datasets, but is not better than the upper bound (except for Pitts-CycleGAN) as expected. It shows that the presence of outliers reduces the accuracy of cross domain image matching, and our method helps improve the performance in this case.

In Figure 4.4, we also show the retrieval performance in terms of the trade-off between precision and recall at different thresholds on our three datasets. The interpolated average

| Method ($T \rightarrow S$) | Shape | Office | Pitts-CycleGAN |
|---|---|---|---|
| SiameseDA | $26.4 \pm 0.2$ | $29.1 \pm 0.1$ | $0.4 \pm 0.01$ |
| **DA+OutlierDetection** | $\mathbf{11.9 \pm 0.1}$ | $\mathbf{15.9 \pm 0.2}$ | $\mathbf{1.1 \pm 0.03}$ |
| SiameseDAOut | $5.4 \pm 0.1$ | $6.8 \pm 0.1$ | $0.2 \pm 0.01$ |

Tab. 4.3: MAP performance for cross domain image matching with outlier detection on our three datasets. The proportion of outliers is 10%. Our method **DA+OutlierDetection** outperforms the lower bound, but does not surpass the upper bound.

precision is used for the precision-recall curves. We can see that our method gains over the lower bound method.

**Impact of outlier proportion** We also report the $F_1$-score to measure the performance of outlier detection of our method. Figure 4.5 shows the $F_1$-score of our method as a function of the portion of outlier samples for the three datasets. As can be seen, with the increase in the number of outliers, our method operates consistently robust.

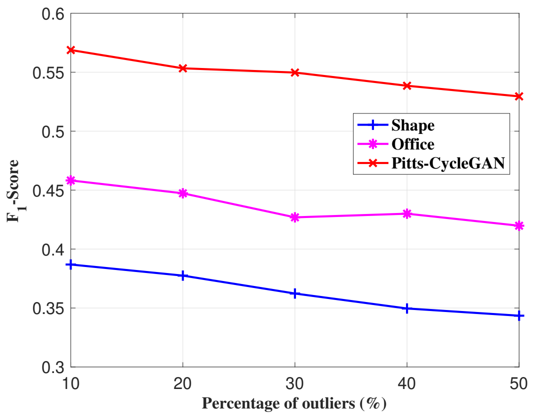


Fig. 4.5: $F_1$-scores for outlier detection on three datasets with different outlier proportion in the target domain. Our method is consistently robust.

It is important to notice the limitation of our method, which classifies some inlier samples as outliers during training. This is mainly caused by the way of initializing the probabilities of the target domain training data.

## 4.6. CONCLUSION

We have proposed a network that is trained for cross domain image matching with outlier detection in an end-to-end manner. The two main parts of our approach are (i) domain adaptive image matching subnetwork with contrastive loss and weighted MK-MMD loss, (ii) outlier detection with entropy loss by updating the probability of target domain data during training. The results on several datasets demonstrate that the proposed method is capable of detecting outlier samples and achieving cross domain image matching at the same time. But our method still needs improvement to overcome the problem of wrongly classifying inliers as outliers.

# REFERENCES

[1] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. “Adapting Visual Category Models to New Domains”. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV*. ECCV’10. 2010, pp. 213–226.

[2] B. Fernando, T. Tommasi, and T. Tuytelaars. “Location recognition over large time lags”. In: *Computer Vision and Image Understanding* 139 (2015), pp. 21–28.

[3] X. Ji, W. Wang, M. Zhang, and Y. Yang. “Cross-Domain Image Retrieval with Attention Modeling”. In: *2017 ACM Multimedia Conference* (2017).

[4] Y. Tian, C. Chen, and M. Shah. “Cross-View Image Matching for Geo-localization in Urban Environments”. In: *In CVPR* (2017).

[5] S. Chopra, R. Hadsell, and Y. LeCun. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern*. 2005, pp. 539–546.

[6] M. Long, Y. Cao, J. Wang, and M. I. Jordan. “Learning Transferable Features with Deep Adaptation Networks”. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*. 2015, pp. 97–105.

[7] K. Saito, Y. Ushiku, and T. Harada. “Asymmetric Tri-training for Unsupervised Domain Adaptation”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. 2017, pp. 2988–2997.

[8] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. “Adversarial Discriminative Domain Adaptation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 2962–2971.

[9] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. “Deep Hashing Network for Unsupervised Domain Adaptation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5385–5394.

[10] X. Zhang, F. X. Yu, S. Chang, and S. Wang. “Deep Transfer Network: Unsupervised Domain Adaptation”. In: *arXiv: 1503.00591* (2015).

[11] D. G. Lowe. “Object Recognition from Local Scale-Invariant Features”. In: *Proceedings of the International Conference on Computer Vision*. ICCV ’99. 1999, pp. 1150–1157.

[12] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays. “Learning Deep Representations for Ground-to-Aerial Geolocalization”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[13] B. Kong, J. Supancic, D. Ramanan, and C. C. Fowlkes. “Cross-Domain Image Matching with Deep Feature Maps”. In: *International Journal of Computer Vision* (2018).

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. “Domain-adversarial Training of Neural Networks”. In: *J. Mach. Learn. Res.* 17 (2016), pp. 2096–2030.

[15] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. “A Kernel Method for the Two-sample-problem”. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. 2006, pp. 513–520.

[16] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. “Optimal kernel choice for large-scale two-sample tests”. In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1205–1213.

[17] R. Chalapathy, A. K. Menon, and S. Chawla. “Anomaly Detection using One-Class Neural Networks”. In: *arXiv:1802.06360* (2018).

[18] W. Liu, G. Hua, and J. R. Smith. “Unsupervised One-Class Learning for Automatic Outlier Removal”. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3826–3833.

[19] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. “Adversarially Learned One-Class Classifier for Novelty Detection”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 3379–3388.

[20] J. Zhang, Z. Ding, W. Li, and P. Ogunbona. “Importance Weighted Adversarial Nets for Partial Domain Adaptation”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[21] R. Hadsell, S. Chopra, and Y. LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR ’06. 2006, pp. 1735–1742.

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. “How Transferable Are Features in Deep Neural Networks?” In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. 2014, pp. 3320–3328.

[23] M. Long, H. Zhu, J. Wang, and M. I. Jordan. “Unsupervised Domain Adaptation with Residual Transfer Networks”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 136–144.

[24] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. “Visual Place Recognition with Repetitive Structures”. In: *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 883–890.

[25] J. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2242–2251.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. 2012, pp. 1097–1105.

[27] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. “Image Classification with the Fisher Vector: Theory and Practice”. In: *Int. J. Comput. Vision* 105.3 (2013), pp. 222–245.

# 5

# WeightAlign: Normalizing Activations by Weight Alignment

*Batch normalization (BN) allows training very deep networks by normalizing activations by mini-batch sample statistics which renders BN unstable for small batch sizes. Current small-batch solutions such as Instance Norm, Layer Norm, and Group Norm use channel statistics which can be computed even for a single sample. Such methods are less stable than BN as they critically depend on the statistics of a single input sample. To address this problem, we propose a normalization of activation without sample statistics. We present WeightAlign: a method that normalizes the weights by the mean and scaled standard derivation computed within a filter, which normalizes activations without computing any sample statistics. Our proposed method is independent of batch size and stable over a wide range of batch sizes. Because weight statistics are orthogonal to sample statistics, we can directly combine WeightAlign with any method for activation normalization. We experimentally demonstrate these benefits for classification on CIFAR-10, CIFAR-100, ImageNet, for semantic segmentation on PASCAL VOC 2012 and for domain adaptation on Office-31.*

This chapter is published as:
Xiangwei Shi$^\star$, Yunqiang Li$^\star$, **Xin Liu**$^\star$, Jan van Gemert. *WeightAlign: Normalizing Activations by Weight Alignment*. International Conference on Pattern Recognition (ICPR), 2020. arXiv: `https://arxiv.org/abs/2010.07160`. $^\star$ The authors contributed equally to this work.

## 5.1. Introduction

Batch Normalization [1] is widely used in deep learning. Examples include image classification [2–4], object detection [5–7], semantic segmentation [8–10], generative models [11–13], *etc*. It is fair to say that for optimizing deep networks, BatchNorm is truly the norm.

BatchNorm stabilizes network optimization by normalizing the activations during training and exploits mini-batch sample statistics. The performance of the normalization thus depends critically on the quality of these sample statistics. Having accurate sample statistics, however, is not possible in all applications. An example is domain adaptation, where the statistics of the training domain samples do match the target domain statistics, and alternatives to BatchNorm are used. [14, 15]. High resolution images are another example, as used in object detection [5, 7, 16], segmentation [8, 9] and video recognition [17, 18], where only a few or just one sample per mini-batch fits in memory. For these cases, it is difficult to accurately estimate activation normalization statistics from the training samples.

To overcome the problem of unreliable sample statistics, various normalization techniques have been proposed which make use of other statistics derived from samples, such as layer [19], instance [20] or group [21]. These methods are applied independently per sample and achieve good performance, but gather statistics just on a single sample and are thus less reliable than BatchNorm.

In this paper, we propose WeightAlign: normalizing activations without using sample statistics. Instead of sample statistics, we re-parameterize the weights within a filter to arrive at correctly normalized activations. See Fig. fig. 5.1 for a visual overview. Our method is based on weight statistics and is thus orthogonal to sample statistics. This allows us to exploit two orthogonal sources to normalize activations: the traditional one based on sample statistics in combination with our proposed new one based on weight statistics. We have the following contributions.

- WeightAlign: A new method to normalize filter weights.
- Activation normalization without computing sample statistics.
- Performance independent of batch size, and stable performance over a wide range of batch sizes.
- State-of-the-art performance on 5 datasets in image classification, object detection, segmentation and domain adaptation.

## 5.2. Related Work

**Network Normalization by Sample Statistics.** The idea of whitening the input to improve training speed [22] can be extended beyond the input layer to intermediate representations inside the network. Local Response Normalization [4], used in AlexNet, normalizes the activations in a small neighborhood for each pixel. Batch Normalization (BN) [1] performs normalization using sample statistics computed over mini-batch, which is helpful for training very deep networks. BN unfortunately suffers from performance degradation when the statistical estimates become unstable for small batch-size based tasks.

To alleviate the small batches issue in BN, Batch Renormalization [23] introduces two extra parameters to correct the statistics during training. However, Batch Renormalization is

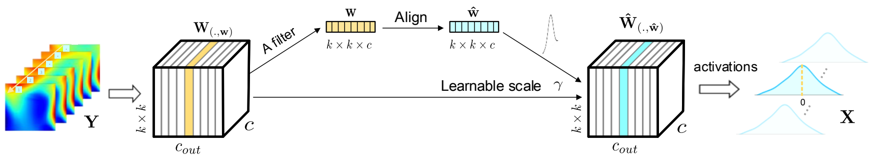


Fig. 5.1: Overview of WeightAlign (WA): Aligning filter weights allows normalizing channel activations. The weight $\mathbf{W}$ of an arbitrary convolutional layer is composed with $C_{out}$ filters with size $C \times k \times k$. Within a filter (yellow region in the left side), we first flatten it to a vector and then apply our proposed WA in Eq. (eq. (5.10)) to normalize the filter weights with a introduced learnable parameter $\gamma$. Normalizing weights of each filter in forward pass can realize the normalization of activations within each channel. Details in Section 5.3.

still dependent on mini-batch statistics, and degrades performance for smaller mini-batches. EvalNorm [24] corrects the batch statistics during inference procedure. But it fails to fully alleviate the small batch issue. SyncBN [25] handles the small batch problem by computing the mean and variance across multiple GPUs which is not possible without having multiple GPUs available.

Because small mini-batch sample statistics are unreliable, several methods [19–21] perform feature normalization based on channel sample statistics. Instance Normalization [20] performs normalization similar to BN but only for a single sample. Layer Normalization [19] uses activation normalization along the channel dimension for each sample. Filter Response Normalization [26] proposes a novel combination of a normalization that operates on each activation channel of each batch element independently. Group Normalization [21] divides channels into groups and computes the mean and variance within each group for normalization. Local Context Normalization [27] normalizes every feature based on the filters in its group and a window around it. These methods can alleviate the small batch problem to some extent, yet have to compute statistics for just a single sample both during training and inference which introduces instabilities [28, 29]. Instead, in this paper we cast the activation normalization over *feature space* into weights manipulation over *parameter space*. Network parameters are independent of any sample statistics and makes our method particularly well suited as an orthogonal information source which can compensate for unstable sample normalization methods.

**Importance of weights.** Proper network weight parameter initialization [4, 30, 31] is essential for avoiding vanishing and exploding gradients [32]. Randomly initializing weight values from a normal distribution [4] is not ideal because of the stacking of non-linear activation functions. Properly scaling the initialization for stacked sigmoid or tanh activation functions leads to Xavier initialization [33], but it is not valid for ReLU activations. He *et al.* [3] extend Xavier initialization [33] and derive a sound initialization for ReLU activations. For hypernets, [34] developed principled techniques for weight initialization to solve the exploding activations even with the use of BN. In [35] a data-dependent initialization is proposed to mimic the effect of BN to normalize the variance per layer to one in the first forward pass. To address the initialization problem of residual nets, Zhang *et al.* [36]

propose fixup initialization. We draw inspiration from these weight initialization methods to derive our weight alignment for activation normalization.

Weight decay [37] adds a regularization term to encourage small weights. Max-norm regularization is used in [38] to avoid extremely large weights. Instead of introducing an extra term to the loss, Weight Norm [39] re-parameterizes weights by dividing its norm to accelerate convergence. Such a re-parameterization, however, may greatly magnify the magnitudes of input signals. In contrast, we scale each filter's weights as inspired by He *et al.* [3]'s weight initialization approach, to generate activations with zero mean and maintained variance. This allows us to normalize activations without using any sample statistics.

## 5.3. Proposed Method

Batch Normalization (BN) [1] normalizes the features in a single channel to a zero mean and unit variance distribution, as in Fig. fig. 5.2, to stabilize the optimization of deep networks. The normalized features $\hat{\mathbf{x}}$ are computed channel-wise at training time using the sample mean $\mu_\beta$ and standard derivation $\sigma_\beta$ over the input features $\mathbf{x}$ as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_\beta}{\sigma_\beta}, \tag{5.1}$$

where $\mu_\beta$ and $\sigma_\beta$ are functions of *sample statistics* of $\mathbf{x}$. For each activation $\hat{\mathbf{x}}$, a pair of trainable parameters $\gamma$, $\beta$ are introduced to scale and shift the normalized value that

$$\mathbf{r} = \gamma \hat{\mathbf{x}} + \beta. \tag{5.2}$$

BN uses the running averages of sample statistics within each mini-batch to reflect the statistics over the full training set. A small batch leads to inaccurate estimation of the sample statistics, thus using small batches degrades the accuracy severely.

In the following, we express the sample statistics in terms of filter weights to eliminate the effect of small batch, and re-parameterize the weights within each filter to realize the normalization of activations within each channel.

### 5.3.1. Expressing activation statistics via weights

For a convolution layer, let $\mathbf{x}$ be a single channel output activation, $\mathbf{Y}$ be the input activations, and $\mathbf{w}$, $b$ be the corresponding filter and bias scalar. Specifically, a filter $\mathbf{w}$ consists of $c$ channels with a spatial size of $k \times k$. To normalize activations by using weights, we show how the sample statistics $\mu_\beta$ and $\sigma_\beta$ in Eq. (eq. (5.1)) are expressed in terms of filter weights.

An individual response value $x(t)$ in $\mathbf{x}$ is a sum of the product between the values in $\mathbf{w}$ and the values in a co-located $k^2c$-by-1 vector $\mathbf{y}(t)$ in $\mathbf{Y}$, which is equivalently written as the dot product of filter $\mathbf{w}$ and vector $\mathbf{y}(t)$,

$$x(t) = \mathbf{w} \cdot \mathbf{y}(t) + b. \tag{5.3}$$

By computing each individual response $x(t)$ at any location $t$, we obtain the full single channel output activation $\mathbf{x}$.

We use random variables $x$, $Y$ and $w$ to present each of the elements in $\mathbf{x}$, $\mathbf{Y}$ and $\mathbf{w}$ respectively. Following [3, 33, 34], we make the following assumptions about the network: (**1**) The $w$ , $Y$ and $b$ are all independent of each other. (**2**) The bias term $b$ equals 0.

*Expressing $\mu_\beta$ of Eq. (eq. (5.1)) in terms of weights:* The mean of activation $\mathbf{x}$ can be equivalently represented via filter weights as:

$$\mu_\beta = \mathbb{E}[x] = n\mathbb{E}[wY] = n\mathbb{E}[w]\mathbb{E}[Y], \tag{5.4}$$

where $\mathbb{E}$ is the expected value and $n = k^2 c$ denotes the number of weight values in a filter.

*Expressing $\sigma_\beta$ of Eq. (eq. (5.1)) in terms of weights:* The variance of activation $\mathbf{x}$ is:

$$\begin{aligned}\sigma_\beta^2 = \mathrm{Var}[x] =& n\mathrm{Var}[wY] = n(\mathbb{E}[w^2Y^2] - \mathbb{E}^2[wY]) \\ =& n(\mathbb{E}[w^2]\mathbb{E}[Y^2] - \mathbb{E}^2[w]\mathbb{E}^2[Y])\end{aligned} \tag{5.5}$$

With the Eq. (eq. (5.4)) and Eq. (eq. (5.5)), we will manipulate the weights to normalize activations in the following sections.

## 5.3.2. WEIGHTALIGN

We normalize activations by encouraging the mean $\mu_\beta$ of the activations in Eq. (eq. (5.4)) to be zero, and maintaining the variance of the activations to be the same in all layers. Inspired by [3], we manipulate the weights to satisfy these requirements.

The mean of the activations $\mu_\beta$ in Eq. (eq. (5.4)) is forced to be zero when the weight $w$ in a filter has zero mean:

$$\mathbb{E}[w] = 0. \tag{5.6}$$

We can exploit $\mathbb{E}[w] = 0$ in Eq. (eq. (5.5)) to further simplify the variance of the activations $\sigma_\beta^2$ in terms of the variance of the weights as,

$$\sigma_\beta^2 = \mathrm{Var}[x] = n\mathbb{E}[w^2]\mathbb{E}[Y^2] = n\mathrm{Var}[w]\mathbb{E}[Y^2]. \tag{5.7}$$

Thus, the activation variance $\sigma_\beta^2$ depends on the variance of the weights $w$ and on the current layer input $Y$. To simplify further, we follow [3] and assume a ReLU activations function. We use $Z$ to define the outputs of the previous layer before the activation function, where $Y = \mathrm{ReLU}(Z)$.

If the weight of the previous layer has a symmetric distribution around zero then $Z$ has a symmetric distribution around zero.[1] Because the ReLU sets all negative values to 0, the variance is halved [3], and we have that $\mathbb{E}[Y^2] = \frac{1}{2}\mathrm{Var}[Z]$. Substituted into Eq. (eq. (5.7)), we have

$$\sigma_\beta^2 = \mathrm{Var[x]} = \frac{1}{2}\mathrm{nVar[w]Var[Z]}. \tag{5.8}$$

This shows the relationship of variance between the previous layer's output before the non-linearity $Z$ and the current layer's single channel activation $x$.

Our goal is to manipulate the weights to keep the activation variance of all layers normalized. Thus we relate the variance of this layer $\sigma_\beta^2$ with the variance in the previous layer

[1] See proof in the supplementary material.

Var[Z], which can be achieved in terms of the weights by setting $\frac{1}{2}n\mathrm{Var}[w]$ in Eq. (eq. (5.8)) to a proper scalar (*e.g.* 1):

$$\frac{1}{2}n\mathrm{Var}[w] = 1. \tag{5.9}$$

With Eq. (eq. (5.6)) and Eq. (eq. (5.9)), we can re-parameterize a single filter weights to have zero mean and a standard deviation $\sqrt{2/n}$, which achieves the activation normalization like BN in Eq. (eq. (5.1)). Thus, here we propose WeightAlign as,

$$\hat{w} = \gamma \, \frac{w - \mathbb{E}[w]}{\sqrt{n/2 \cdot \mathrm{Var}[w]}}, \tag{5.10}$$

where $\gamma$ is a learnable scalar parameter

similar to the scale parameter in BN [1] like Eq. (eq. (5.2)). Since WA manipulates the weights, it does not have the $\beta$ term as in the form of BN, which is applied on activations directly. The detail explanation about this is given in supplementary material. From Eq. (eq. (5.10)), we can tell that the proposed WeightAlign(WA) method does not rely on sample statistics computed over a mini-batch, which makes it independent of batch size.

### 5.3.3. Initialization of weights

We initialize each filter weights **w** to be a zero-mean symmetric Gaussian distribution whose standard deviation (std) is $\sqrt{2/n}$ where $n = k^2c$. For the first layer, we should have $\sqrt{1/n}$ since the ReLU activation is not applied on the input. But the factor 1/2 can be neglected if it just exists on one layer and for simplicity, we use the same deviation $\sqrt{2/n}$ for initialization.

### 5.3.4. Empirical analysis and examples

To show the distributions of channel activations using different methods, we build an 8-layer deep network containing 7 convolutional layers and one classification layer with ReLU as nonlinear activation function. A mini-batch of size 128 independent data is sampled from $\mathcal{N}(0,1)$ is used as input. Specifically, the standard Kaiming initialization [3] without bias term is used to initialize the weights. Fig. fig. 5.2 shows the activation distributions of 8 different channels taken from two layers, the 3rd intermediate convolutional layer and the last classifier layer before the softmax loss. Each channel of the last classifier layer represents one class. To exclude the training effect, we first use the initialization model to conduct comparison between sample statistics based normalization and our WA normalization. For the visualization of trained model, please see Section 5.4. For the baseline model, it can be seen in Fig. fig. 5.2(a) that the activation distributions in the intermediate layer start drifting, leading to a constant output for the classifier layer (*i.e.* the 'Blue' indexed class/channel). From Fig. fig. 5.2(b) of Batch Norm (BN) [1], we note that reducing the Internal Covariate Shift (ICS) in intermediate layer can alleviate the distribution drifting for the final classifier layer. The output of the classifier layer is no longer a constant function. Group Norm(GN) [21] can also reduce ICS to some extent, as shown in Fig. fig. 5.2(c). As shown in Fig. fig. 5.2(d), WA can realize similar functionality of BN. Our proposed WeighAlign (WA) re-parameterizes the weights within each filter in Eq. (eq. (5.10)) to normalize channel activation. Since weight statistics is orthogonal to sample statistics, WA can be used in conjunction with BN, GN, LN and IN, as shown in Fig. fig. 5.2(e). Please refer to supplementary material for visualization of other normalization methods.

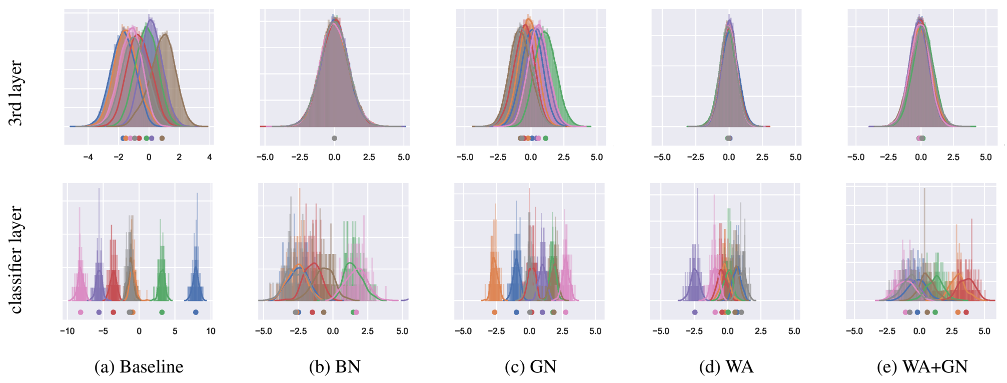


Fig. 5.2: Our WA method can alleviate internal covariate shift of activation (see Section 5.3.4 for details). Each color represents the activation distribution of different channels for different layers. For baseline model, the 'Blue' indexed channel will dominate all other channels, leading to a constant classification result. Normalizing channel activation in intermediate layer can avoid the constant output for the final classifier layer, which can be realized by WA.

## 5.4. EXPERIMENTS

We show extensive experiments on three tasks and five datasets: image classification on CIFAR-10, CIFAR-100 [40] and ImageNet; domain adaptation on Office-31; and semantic segmentation on PASCAL VOC 2012 [41] where we evaluate VGG [31] and residual networks as ResNet [30].

### 5.4.1. DATASETS

**CIFAR-10 & CIFAR-100.** CIFAR-10 and CIFAR-100 [40] consist of 60,000 32×32 color images with 10 and 100 classes, respectively. For both datasets, 10,000 images are selected as the test set and the remaining 50,000 images are used for training. We perform image classification experiments and ablation study on these datasets.

**ImageNet.** ImageNet [42] is a classification dataset with 1,000 classes. The size of the training dataset is around 1.28 million, and 50,000 validation images are used for evaluation.

**Office-31.** Office-31 [43] is a dataset for domain adaptation with 4,652 images with 31 categories. The images are collected from three distinct domains: Amazon (A), DSLR (D), and Webcam (W). The largest domain, Amazon, has 2,817 labeled images. The 31 classes consists of objects commonly encountered in office settings, such as file cabinets, keyboards and laptops.

**PASCAL VOC 2012.** PASCAL VOC 2012 [41] contains a semantic segmentation set and includes 20 foreground classes and a single background class. The original segmentation dataset contains 1,464 images for training, and 1,499 images for evaluation. We use the augmented training dataset including 10,582 images [44].

### 5.4.2. Implementation details

We use Stochastic Gradient Descent(SGD) in all experiments with a momentum of 0.9 and a weight decay of $5 \times 10^{-4}$. The plain VGG model is initialized with Kaiming initialization [3] and ResNets are initialized with Fixup [36]. We do not apply WA in the final classifier layer. For classification, we train the models with data augmentation, random horizontal flipping and random crop, as in [45]. Further implementation details are given in each subsection.

### 5.4.3. Experiments on CIFAR-10 & CIFAR-100

**Comparing normalization methods.** We compare our proposed WeightAlign (WA) against various activation normalization methods, including Batch Normalization (BN) [1], Group Normalization (GN) [21], Layer Normalization (LN) [19], Instance Normalization (IN) [20]. To demonstrate that WA is orthogonal to all these approaches, we also show the combination of WA with these activation normalization methods.

We conduct experiments on CIFAR-100 as shown in Tab. table 5.1, where all models are trained with a batch size of 64. WeightAlign outperforms every normalization method except BN over large batch size. By combining WeightAlign with sample statistics normalization methods, we can see that weightAlign adds additional information and improves accuracy. Especially, the GN+WA model achieves comparable performance of BN.

To further show it's flexibility, we fine-tune a pre-trained BN model on CIFAR-100 with WA. It achieves 22.38% error rate, which is the same as BN+WA model trained from scratch. This shows the possibility of finetuning WA on a pre-trained models with a sample-based normalization method.

On CIFAR-10 we do the same evaluation for ResNet50. BN and BN+WA are trained only with batch size of 64, and other methods are trained with batch size of 1 and 64. For batch size of 64 and 1, the initial learning rates are 0.01 and 0.001, respectively. The baseline method [36] is trained without any normalization methods. Results are given in Tab. table 5.2. Because CIFAR-10 is less complex than CIFAR-100, the performance gaps between different methods are thus less obvious. Nevertheless, our method is comparable to other normalization methods and it allows a batch size of 1, which is not possible for BN. By combining WA with the sample-based normalization methods, we again observe that weight statistics normalization helps in addition to sample statistics normalization.

We also compare WA with Weight Normalization (WN) [39] on CIFAR-100 with VGG16 and ResNet50 networks. For VGG16, the error rates of WN, WA, GN are 6.8%, 4.8%and 7.4% when the batch size is 1, and the ones of WN, WA, BN, GN are 8.6%, 5.4%, 5.8% and 11.0% when the batch size is 64. For ResNet50, WN cannot be trained. We can see WA and WN are workable regardless of the batch size for plain networks and WA outperforms WN in this model. For residual networks, WA can work alone without any activation normalization layer, while WN cannot. WN is only normalized by its norm to accelerate training. Without normalization layer in residual networks, WN cannot restrict the variance of the activation and the residual structure merges the activation from two branches, which leads to exploding gradients problem. Instead, WA normalizes the weights by zero-mean and a scaled standard deviation. This scale is determined by the size of convolutional filter and restricts the variance of weights and further maintains the variance of activations.

Tab. 5.1: Error rate of ResNet50 on CIFAR-100 for classification. WA outperforms every normalization method except BN overlarge batch size. When combined with statistics normalization methods, WA improves accuracy.

| CIFAR-100 | | | |
|---|---|---|---|
| Method | Error | Method | error |
| Baseline [36] | 28.43 | WA | 24.92 |
| BN [1] | 23.02 | BN + WA | 22.39 |
| IN [20] | 25.58 | IN + WA | 24.63 |
| LN [19] | 26.78 | LN + WA | 24.01 |
| GN [21] | 25.46 | GN + WA | 23.64 |

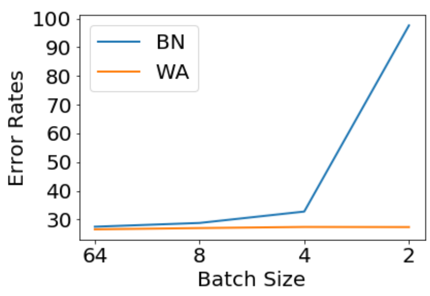
Tab. 5.2: Error rate of ResNet50 on CIFAR-10 for classification. Weight statistics normalization (WA) helps in addition to sample statistics normalization.

| CIFAR-10 | | | | | | | |
|---|---|---|---|---|---|---|---|
| Batch size 64 | | | | Batch size 1 | | | |
| Method | Error | Method | Error | Method | Error | Method | Error |
| Baseline [36] | 6.46 | WA | 6.21 | Baseline | 7.27 | WA | 6.61 |
| BN [23] | 4.30 | BN+WA | 4.29 | BN | - | BN+WA | - |
| IN [20] | 6.49 | IN+WA | 6.42 | IN | 6.91 | IN+WA | 6.50 |
| LN [19] | 5.02 | LN+WA | 5.12 | LN | 6.82 | LN+WA | 5.76 |
| GN [21] | 4.96 | GN+WA | 4.60 | GN | 5.79 | GN+WA | 5.51 |

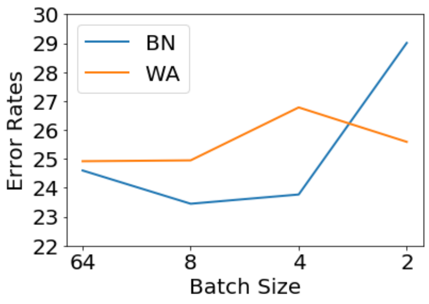
**Small batch sizes.** In Fig. fig. 5.3, we compare BN and WA with different batch sizes on VGG16 and ResNet18. We train with batch sizes of 64, 8, 4, 2 images per GPU. In all cases, the means and variances in the batch normalization layers are computed within each GPU and not synchronized. The x-axis shows different batch sizes and y-axis presents the validation error rates. From Fig. fig. 5.3 (a), we observe that the performance of WA is comparable to that of BN on the plain network with large batches. The error rates of BN increase rapidly when reducing the batch size, especially with batch size smaller than 4. In contrast, our method focuses on the normalization of weight distributions which is independent of batch size. The error rates of our method are stable over different batch sizes. We can observe similar results in Fig. fig. 5.3 (b) that our method performs stable over different batch size on residual network. It is interesting to see that the performance gap between WA and BN with ResNet18 is smaller than that with VGG16, which is due to the residual structure.

**Depth of residual networks.** We evaluate depth of 18, 34, 50, 101, 152 for residual networks with batch sizes of 1, 64 per GPU. Tab. table 5.3 shows the validation error rates on CIFAR-10. We observe that WA is able to train very deep networks for both small and large batch sizes.

**Visualization of weights and activations in training.** To verify the motivation, in this



(a) VGG16

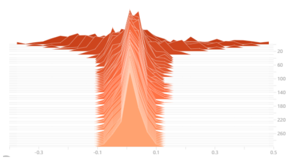


(b) ResNet18

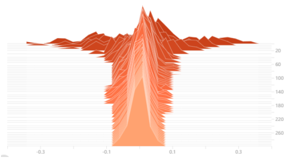
Fig. 5.3: Classification error *vs.* batch sizes. The models are VGG16 and ResNet18 trained on CIFAR-100. The error rates of BN increase rapidly when small batch sizes are used. Our proposed method does not rely on the sample statistics and is independent of batch size dimension. The error rates of WA are stable over a wide range of batch sizes.

Tab. 5.3: Applying WA to varying depth measured by error rate. WA is able to train very deep networks for both small and large batch sizes.

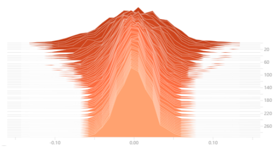
| | CIFAR-10 | |
|---|---|---|
| Model (+WA) | Batchsize 1 Error | Batchsize 64 Error |
| ResNet18 | 5.65 | 6.23 |
| ResNet34 | 5.84 | 5.78 |
| ResNet50 | 6.61 | 6.42 |
| ResNet101 | 5.74 | 6.41 |
| ResNet152 | 6.09 | 6.52 |



(a) Baseline



(b) BN



(c) WA

Fig. 5.4: Channel weights distribution of trained ResNet50 versus epoch: (a) channel weights of baseline model. (b) channel weights of BN model. (c) channel weights of WA model. The x-axis represents the value of weights. The y-axis represents the epoch. From top to bottom, the epoch varies from 1 to 300. WA makes the distributions of channel weights symmetric around 0. With a symmetric distribution around zero for the weight, the activation can also have a symmetric distribution around zero.

subsection, we visualize the distribution of weights and activation by training a ResNet50 model on CIFAR-100. A minibatch of 128 images from CIFAR-100 is input to the trained network to obtain features and weights.

**How are the filter weights distributed?** In Fig. fig. 5.4, we visualize how the channel weight distribution changes with the epochs. Fig. fig. 5.4 (a) illustrates the baseline ResNet50 model without using any normalization. When using BN, as shown in Fig. fig. 5.4 (b), the weight distribution tends to be symmetric around zero. From Fig. fig. 5.4 (c), we note that the proposed WA method can realize the same functionality of BN making the weights symmetric and smooth during training. Channel and layer weights distribution during training of other normalization methods can be found in the supplementary material. With a symmetric distribution around zero for the weight, the activation can also have a symmetric distribution around zero.

**How are the channel activations distributed?** In Fig. fig. 5.2 we have found that before training, the proposed WA method can effectively normalize channel activations by aligning the weights within each filter. In Fig. fig. 5.5 we further explore how the channel activations distributed on a trained ResNet50 network. We visualize the activation distributions for 4 different channels taken from two layers, a middle layer (the 2rd conv layer of the 8th residual block) and the last classifier layer. Since the weight decay is used, the scale and shift parameters $\gamma$, $\beta$ are converged to small values. From Fig. fig. 5.5 (d), we have similar observations as in Fig. fig. 5.2 that our method can normalize activation of different channels to zero mean and same variance. Fig. fig. 5.5 (e) also illustrates WA can be used in conjunction with GN to further normalize the activations.

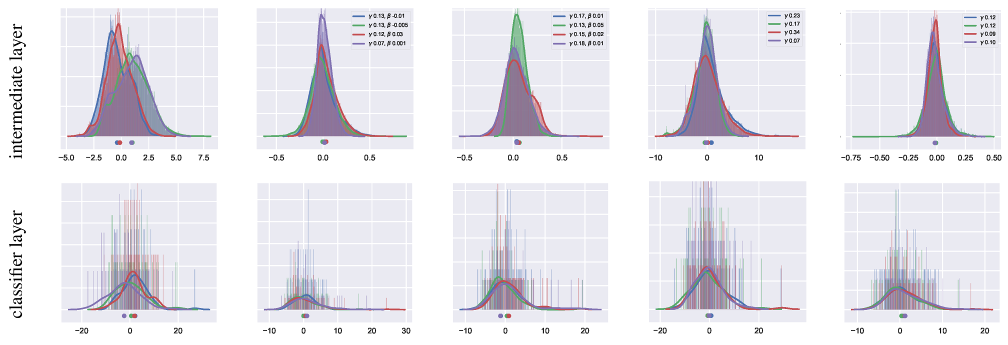
**Functionality of different components.** We align the filter weights by subtracting the mean and dividing by a properly scaled derivation. For our method, there are two components: $\mathbb{E}[w] = 0$ in Eq. (eq. (5.6)) $\mathrm{Var}[w] = \frac{2}{n}$ in Eq. (eq. (5.9)). We conduct comparative results for our model with different components for VGG and ResNet18 models on CIFAR-100. In Tab. table 5.4 we start with the baseline model [36] and augment incrementally with the two components. We observe that both components contribute substantially to the performance of the whole model.

### 5.4.4. COMPARISON WITH STATE-OF-THE-ART

**Image classification on ImageNet.** We experiment with a regular batch size of 64 images per GPU on plain and residual networks. The total training epoch is 100. The initial learning rate is 0.1 and divided by 10 at the 30th, 60th and 90th epoch. Tab. table 5.5 shows the top-1 and top-5 error rates of image classification on ImageNet.

For plain networks, WA performs relatively well as BN, only 0.2% difference on top-1 error rate. Combining WA with BN outperforms BN on VGG16 by 2.51% top-1 error rate and 1.38% top-5 error rate. For residual networks, WA performs slightly worse than BN, but combined BN+WA surpasses BN by 0.85% on top-1 error rate and 0.59% on top-5 error rate. This shows that WA and WA+BN can work well on large and complex datasets.

**Domain adaptation on Office-31.** Tab. table 5.6 presents overall scores of different normalization methods and their combination with WA on Office-31 dataset. Here we use ResNet50 models pretraied on CIFAR-100, which are obtained from previous experiments.



(a) Baseline (b) BN (c) GN (d) WA (e) WA+GN

Fig. 5.5: Channel activation distributions of trained models on CIFAR-100. Different color represents a different channel. $\gamma$ and $\beta$ are the parameters in each normalization. WA can align the channel activations by normalizing channel weights.

Tab. 5.4: Comparative results for VGG16 and ResNet18 models with different components. Both components contribute substantially to the performance.

| Components in WA | | VGG16 | ResNet18 |
|---|---|---|---|
| $\mathbb{E}[w] = 0$ | $\text{Var}[w] = \frac{2}{n}$ | Error | Error |
| × | × | - | 28.23 |
| ✓ | × | 35.74 | 27.97 |
| × | ✓ | - | 28.03 |
| ✓ | ✓ | 27.85 | 24.92 |

Tab. 5.5: Top-1 and top-5 error rates of image classification on ImageNet. WA and WA+BN can work well on large and complex datasets.

| model | Top-1 (%) Error | Top-5 (%) Error |
|---|---|---|
| VGG16* (Baseline) | 31.30 | 11.19 |
| VGG16 (BN)* | 29.58 | 10.16 |
| VGG16 (WA) | 29.78 | 10.23 |
| VGG16 (BN+WA) | **27.07** | **8.78** |
| ResNet50 (Baseline)† [36] | 27.60 | - |
| ResNet50 (BN)* | 24.89 | 7.71 |
| ResNet50 (WA) | 26.62 | 8.91 |
| ResNet50 (BN+WA) | **24.04** | **7.12** |

The * denotes that we directly test the PyTorch pretrained model.
The † denotes the numbers from the reference.

To perform domain adaptation experiments, we finetune the pretrained model on a source

Tab. 5.6: Classification accuracies (%) on Office-31 dataset (ResNet50). WA can be applied to other normalization methods and improves domain adaptation performance.

| Method | A → W | W → A | A → D | D → A | W → D | D → W | Avg |
|---|---|---|---|---|---|---|---|
| WA | 34.62 | 42.89 | 52.05 | **44.09** | 87.67 | 80.00 | 56.89 |
| BN [1] | 46.92 | 48.91 | 57.53 | 37.11 | 90.41 | 70.00 | 58.48 |
| BN + WA | 49.23 | 36.38 | **58.90** | 40.00 | **91.78** | 76.92 | 58.87 |
| GN [21] | 46.92 | 42.89 | 50.68 | 39.76 | 82.19 | 75.38 | 56.30 |
| GN + WA | 46.15 | **54.46** | 50.68 | 37.83 | 87.67 | 78.46 | 59.21 |
| IN [20] | 36.92 | 33.98 | 41.10 | 37.11 | 84.93 | 76.92 | 51.83 |
| IN + WA | 39.23 | 41.45 | 38.36 | 38.07 | 90.41 | 73.85 | 53.56 |
| LN [19] | 37.69 | 29.89 | 45.20 | 33.97 | 79.45 | 69.23 | 49.24 |
| LN + WA | **50.77** | 42.89 | 56.16 | 41.69 | **91.78** | **80.77** | **60.68** |

domain, and validate it on two target domains with 0.001 initial learning rate, e.g. finetune on A, and validate on W and D. Model with WA consistently outperforms model with GN, IN and LN. Tab. table 5.6 also shows that our method WA can be applied to other normalization methods successfully and improves domain adaptation performance. In Office-31, the three domains A, W and D have different sample distribution. Thus the normalization methods that rely on sample statistics perform worse than our method WA, which only applies on weights and is independent of sample statistics. When combined with WA, other activation normalization methods can benefit from the sample-independent merit of WA, which boosts their domain adaptation performance.

**Semantic segmentation on PASCAL VOC 2012.** To investigate the effect on small batch size, we conduct experiments on semantic segmentation with PASCAL VOC 2012 dataset. We select the DeepLabv3 [46] as baseline model in which the ResNet50 pre-trained on ImageNet is used as backbone. Given the pretrained ResNet50 models with BN and BN+WA on ImageNet, we finetune them on PASCAL VOC 2012 for 50 epochs with batch size 4, 0.007 learning rate with polynomial decay and multi-grid (1,1,1), and evaluate the performance on the validation set with output stride 16. The evaluation mIoU of BN method is 73.80%, and that of BN+WA is 74.87%. Combined with WA, the original BN performance is improved by 1.03%. This further shows that WA improves other normalization methods.

## 5.5. CONCLUSION

We propose WeightAlign; a method that re-parameterizes the weights by the mean and scaled standard derivation computed within a filter. We experimentally demonstrate WeightAlign on five different datasets. WeightAlign does not rely on the sample statistics and performs on par with Batch Normalization regardless of batch size. WeightAlign can also be combined with other activation normalization methods and consistently improves on Batch Normalization, Group Normalization, Layer Normalization, and Instance Normalization on various tasks, such as image classification, domain adaptation, and semantic segmentation.

## 5.6. Appendix

### 5.6.1. Proof of symmetric

Given two independent random variable $X$ and $Y$, the distribution of product random variable $Z$, where $Z = XY$, can be found as follows,

$$f_Z(z) = \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y(\frac{z}{t})\, dt. \tag{5.11}$$

If the distribution of $X$ is continuous at 0, then we have,

$$\begin{aligned} P(Z \le z) &= P(XY \le z) \\ &= P(Y \le \frac{z}{X} | X > 0) P(X > 0) + P(Y \ge \frac{z}{X} | X < 0) P(X < 0) \\ &= \int_0^{\infty} P(Y \le \frac{z}{t}) f_X(t)\, dt + \int_{-\infty}^{0} P(Y \ge \frac{z}{t}) f_X(t)\, dt. \end{aligned} \tag{5.12}$$

We take the derivation of both sides w.r.t. z and we get,

$$\begin{aligned} f_Z(z) &= \int_0^{\infty} \frac{1}{t} f_Y(\frac{z}{t}) f_X(t)\, dt + \int_{-\infty}^{0} \frac{-1}{t} f_Y(\frac{z}{t}) f_X(t)\, dt \\ &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y(\frac{z}{t})\, dt. \end{aligned} \tag{5.13}$$

Suppose the distribution of random variable $Y$ is symmetric around zero, where $f_Y(y) = f_Y(-y)$. Then we can have,

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y(\frac{z}{t})\, dt \\ &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y(\frac{-z}{t})\, dt \\ &= f_Z(-z). \end{aligned} \tag{5.14}$$

Therefore, the distribution of product random variable $Z$ will be symmetric around zero, if the distribution of one of independent random variable $X$ and $Y$ is symmetric around zero.

### 5.6.2. Explanation of WA equation form

In section 5.3.2, we give out the expression for WA as Eq. (eq. (5.10)). Compared with BN, WA does not have the $\beta$ term since it manipulates the weight directly instead of the activations. If we add a $\beta$ term to Eq. (eq. (5.10)), it will result in,

$$\hat{w} = \gamma \frac{w - \mathbb{E}[w]}{\sqrt{n/2 \cdot \mathrm{Var}[w]}} + \beta. \tag{5.15}$$

When it multiplies with the input activations $\mathbf{x}$, we will get an extra $\beta\mathbf{x}$, which is similar to the element in residual blocks. This will introduce the drift of activation variance again, which goes against our original intention.

### 5.6.3. ADDITIONAL EXPERIMENTS AND VISUALIZATION

#### SCALE FACTOR IN WEIGHTALIGN

We here conduct an ablation study experiments to validate the effect of our scale factor $\sqrt{n/2}$ in Eq. (eq. (5.10)). Specially, we compare our scale factor with a 0.2x our scale factor, 2x our scale factor and 4x our scale factor. The experiments are shown in Fig. fig. 5.6. We find that a similar scale to $\sqrt{n/2}$ will lead to similar performance. But our scale $\sqrt{n/2}$ has the best performance. Any other scale factors would cause the failure of training. Thus a proper scaled derivation plays an important role to make the training stable.

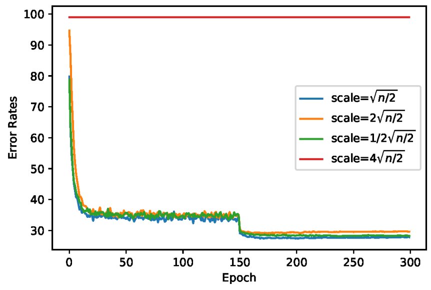


Fig. 5.6: Ablation study of scale factor. Validation error rates with different scale factors on CIFAR-100 (ResNet18). A slightly different scale factor can cause a failure of training like the red line. A proper scale factor is important to stabilize the optimization. Our scale achieves the best performance among other trainable scale factors.

#### EMPIRICAL ANALYSIS IN SECTION 5.3.4

We further show activation distributions for different normalization methods over 8 different channels taken from four different layers: the 1st, 3rd, 7th intermediate convolutional layers and the last classification layer. Fig. fig. 5.7 shows the comparison between baseline and other normalization methods including WA. Fig. fig. 5.8 shows the cases when IN, LN and GN are used in conjunction with our WA.

### 5.6.4. VISUALIZATIONS OF WEIGHTS AND ACTIVATIONS IN SECTION 5.4.3

We visualize the distributions of weights and activations as epoch increases during training. Fig. fig. 5.9 and Fig. fig. 5.10 show weights distributions of a single channel in a convolutional layer. The weights distributions of WA and BN are smooth and symmetric around zero, while the ones of other normalization methods are rough or asymmetric. Adding WA with LN, IN and GN smooths and symmetrizes the weights distributions of channels. Fig. fig. 5.11 and Fig. fig. 5.12 show activation distributions of a single channel in a convolutional layer. Similarly, the activation distributions of WA and BN are smooth and symmetric around zero, while the ones of other normalization methods are rough or asymmetric. Adding WA with LN, IN and GN smooths and symmetrizes the activation distributions of channels.

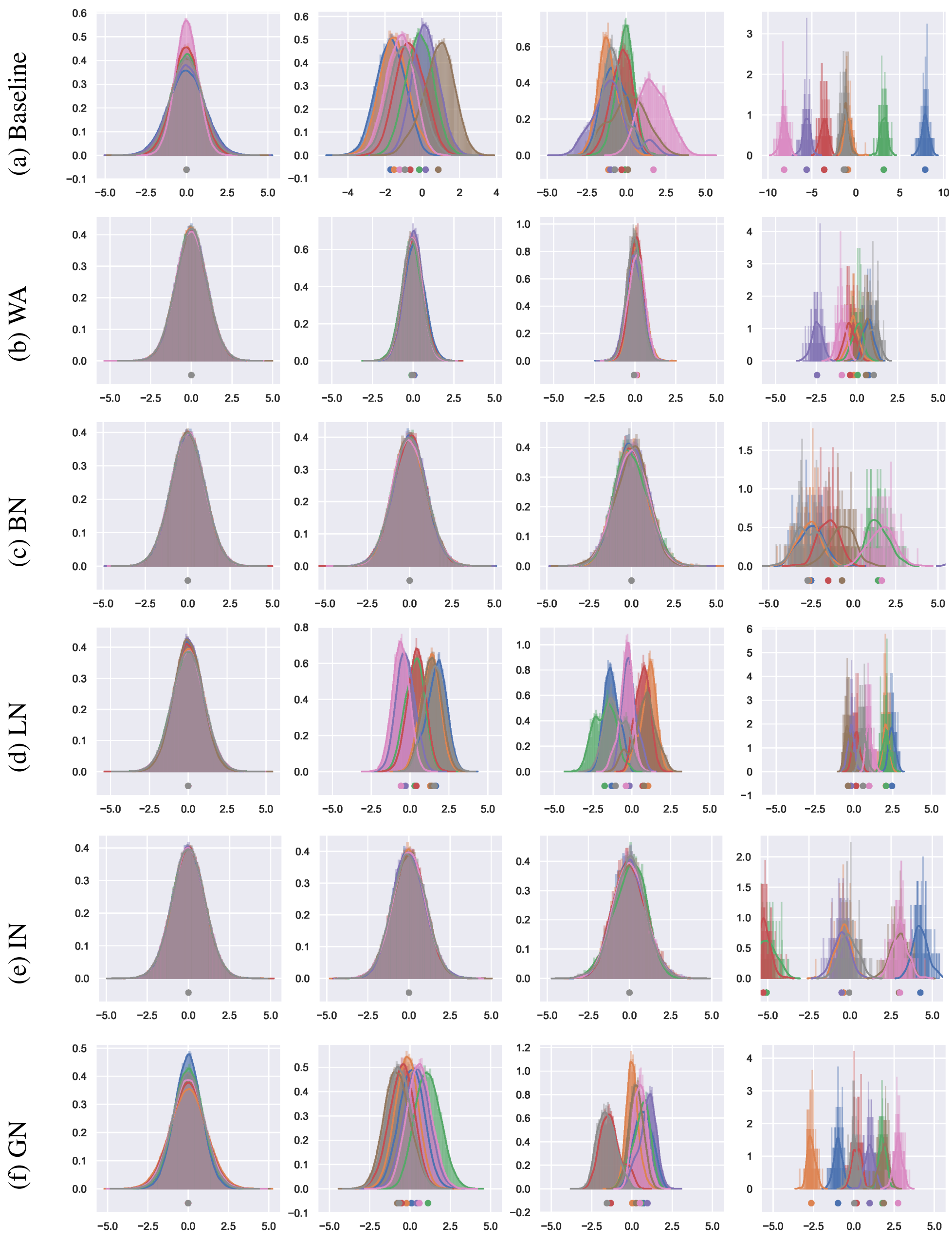


Fig. 5.7: Each color represents the activation distribution of different channels for different layers. The first three columns denote 1st, 3rd, 7th convolutional layers and the last one presents the last classification layer before softmax. All normalization methods can reduce internal covariate shift to some extent comparing with baseline.

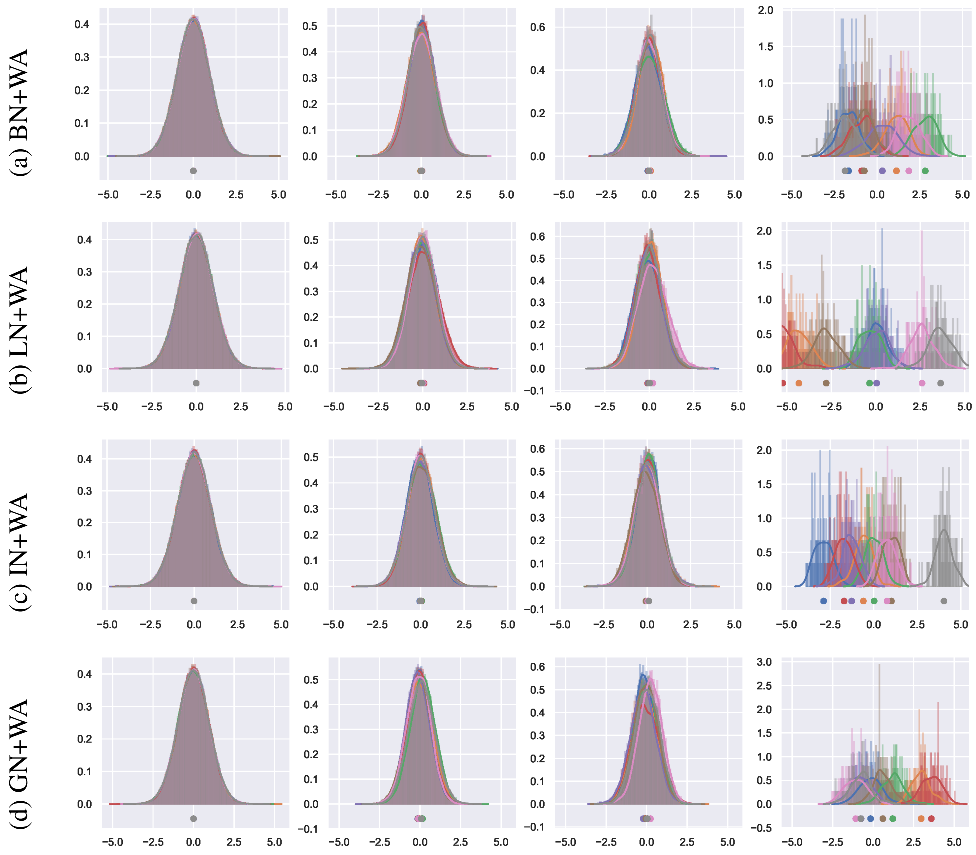
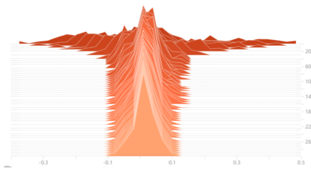
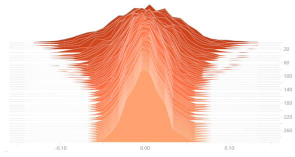
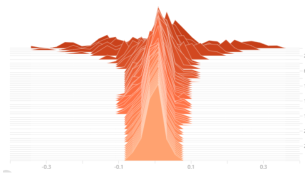
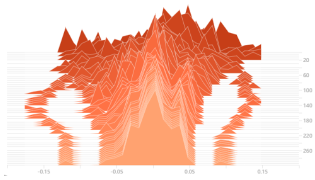


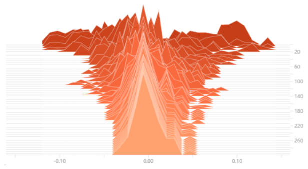
Fig. 5.8: The proposed WA can be used in conjunction with BN, LN, IN and GN. The first three columns denote 1st, 3rd, 7th convolutional layers and the last one presents the last classification layer before softmax. Note that the activation we plot here is before passing through specific normalization layer.


(a) Baseline


(b) WA


(c) BN


(d) LN


(e) IN

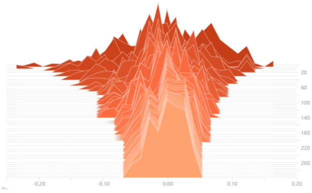
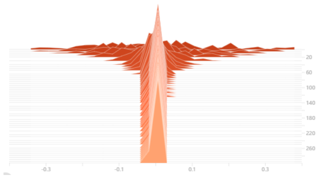
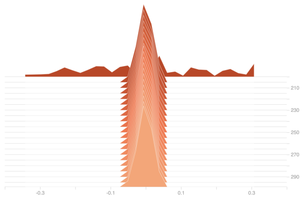
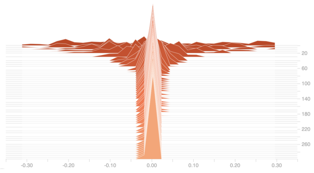

(f) GN

Fig. 5.9: Weights distributions of a single channel in a convolutional layer for different normalization methods. WA and BN have smooth and symmetric weight distributions.


(a) WA + BN


(b) WA + LN


(c) WA + IN

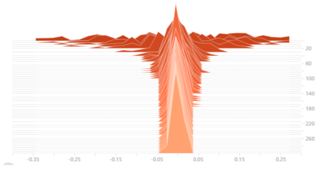
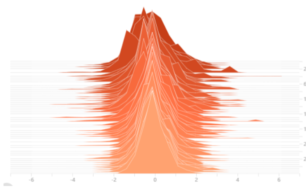
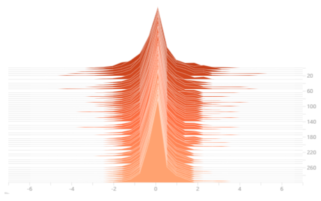
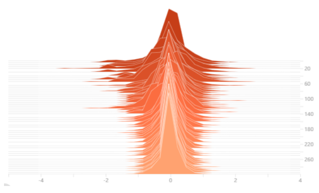
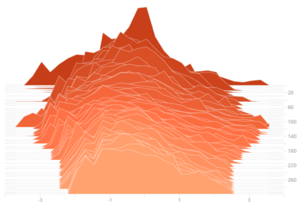

(d) WA + GN

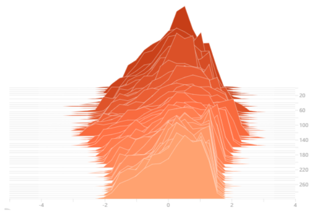
Fig. 5.10: Weights distributions of a single channel in a convolutional layer for different normalization methods in conjunction with WA method. Adding WA smooths and symmetrizes the weight distributions.


(a) Baseline


(b) WA


(c) BN


(d) LN


(e) IN

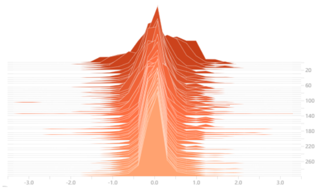
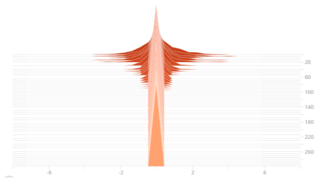
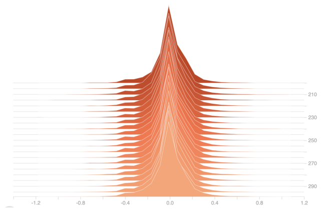
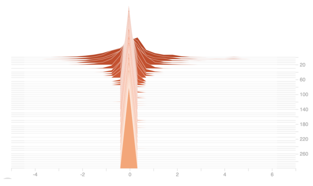

(f) GN

Fig. 5.11: Activation distributions of a single channel in a convolutional layer for different normalization methods. WA and BN have smooth and symmetric weight distributions.


(a) WA + BN


(b) WA + LN


(c) WA + IN

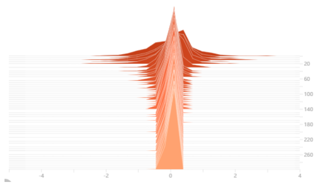

(d) WA + GN

Fig. 5.12: Activation distributions of a single channel in a convolutional layer for different normalization methods in conjunction with WA method. Adding WA smooths and symmetrizes the activation distributions.

# REFERENCES

[1] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[2] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. “PCANet: A simple deep learning baseline for image classification?” In: *IEEE transactions on image processing* 24.12 (2015), pp. 5017–5032.

[3] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[7] S. Ren, K. He, R. Girshick, and J. Sun. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[8] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[9] H. Noh, S. Hong, and B. Han. “Learning deconvolution network for semantic segmentation”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1520–1528.

[10] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[11] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein gan”. In: *arXiv preprint arXiv:1701.07875* (2017).

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[13] A. Radford, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[14] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han. “Domain-Specific Batch Normalization for Unsupervised Domain Adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7354–7362.

[15] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. “Revisiting batch normalization for practical domain adaptation”. In: *arXiv preprint arXiv:1603.04779* (2016).

[16] R. Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[17] J. Carreira and A. Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[19] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[20] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

[21] Y. Wu and K. He. “Group normalization”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[22] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[23] S. Ioffe. “Batch renormalization: Towards reducing minibatch dependence in batch-normalized models”. In: *Advances in neural information processing systems*. 2017, pp. 1945–1953.

[24] S. Singh and A. Shrivastava. “EvalNorm: Estimating Batch Normalization Statistics for Evaluation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3633–3641.

[25] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. “Megdet: A large mini-batch object detector”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6181–6189.

[26] S. Singh and S. Krishnan. “Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11237–11246.

[27] A. Ortiz, C. Robinson, D. Morris, O. Fuentes, C. Kiekintveld, M. M. Hassan, and N. Jojic. “Local Context Normalization: Revisiting Local Normalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11276–11285.

[28] W. Shao, T. Meng, J. Li, R. Zhang, Y. Li, X. Wang, and P. Luo. “Ssn: Learning sparse switchable normalization via sparsestmax”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 443–451.

[29] J. Yan, R. Wan, X. Zhang, W. Zhang, Y. Wei, and J. Sun. “Towards Stabilizing Batch Statistics in Backward Propagation of Batch Normalization”. In: *ICLR* (2020).

[30] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[31] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[32] R. Pascanu, T. Mikolov, and Y. Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. 2013, pp. 1310–1318.

[33] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[34] O. Chang, L. Flokas, and H. Lipson. “Principled Weight Initialization for Hypernetworks”. In: *International Conference on Learning Representations*. 2020.

[35] D. Mishkin and J. Matas. “All you need is a good init”. In: *arXiv preprint arXiv:1511.06422* (2015).

[36] H. Zhang, Y. N. Dauphin, and T. Ma. “Fixup initialization: Residual learning without normalization”. In: *arXiv preprint arXiv:1901.09321* (2019).

[37] A. Krogh and J. A. Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[38] N. Srebro and A. Shraibman. “Rank, trace-norm and max-norm”. In: *International Conference on Computational Learning Theory*. Springer. 2005, pp. 545–560.

[39] T. Salimans and D. P. Kingma. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 901–909.

[40] A. Krizhevsky, G. Hinton, *et al. Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. “The PASCAL visual object classes challenge 2007 (VOC2007) results”. In: (2007).

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[43] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. “Adapting Visual Category Models to New Domains”. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV*. ECCV’10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 213–226. ISBN: 364215560X.

[44] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. “Semantic contours from inverse detectors”. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 991–998.

[45] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. “Random erasing data augmentation”. In: *arXiv preprint arXiv:1708.04896* (2017).

[46] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

# 6 DISCUSSION

## 6.1. CONCLUSION

This thesis analyzes how to improve the efficiency of deep learning models on both video and image data with respect to four tasks: video object detection, video action recognition, cross domain image matching with outlier present, and deep activation normalization independent of sample statistics.

In the context of efficiency on video data, the proposed approaches leverage the redundancy in video frames. For video object detection, this thesis predicts future object locations from single static frames, while using the ground truth locations of all frames in a video. The method is based on the observations that the video object motion is smooth and continuous, which makes it possible to predict future locations of objects from only a static frame, over multiple subsequent future frames. With the predicted future locations, the model does not need to process every frame of a video to perform the object detection, which makes the proposed video object detection method efficient. In terms of video action recognition, the proposed method in the thesis is based on the assumption that similar frames affect the parameter updates similarly and therefore should lead to similar gradients. One can approximate the gradients of multiple frames with a single gradient of an aggregated frame-activation. This also makes use of the redundancy characteristic to video frames, and thus achieves efficiency.

In the context of efficiency on image data, the methods proposed in this thesis for, aim towards data-efficiency. The task of cross-domain image matching with outliers reaches data efficiency by not requiring data labels in the target domain, which largely reduces the annotation effort. While, the proposed activation normalization is independent of sample statistics. This method matches the normalization performance of BatchNorm without relying on data statistics, and thus is data-efficient.

With the works in this thesis, I aim to inspire the research field towards addressing the challenging problem of efficiency in deep learning and focusing on deep models that are both accurate and have reduced computations. The next sections discuss shortly such possible future directions, and are followed by the final reflection on this thesis.

## 6.2. FUTURE WORK

### 6.2.1. WHAT ROLE DOES BACKGROUND PLAY IN VIDEO UNDERSTANDING TASKS?

In this work, I demonstrate state-of-the-art results of the proposed video object detection method in Chapter 2 and video action recognition method in Chapter 3, while also highlighting some limitations. One of the limitations in the efficient video object detection work is that multiple motion patterns can be associated with the exact same appearance, for example, a ball can roll to the left or right on a table. In this case, the proposed video object detection method may fail to predict future locations from a single static frame. In practice, when an object moves the relative position with respect to the background will also change. In other words, the background will change differently if the object moves from left-to-right than if the object moves from top-to-down, while the object has exactly the same appearance. Moreover, objects appear in typical contexts: a car drives on a road, a chair is typically next to a table. If one can make use of the relative context changes, and learn the object motion relative to the context, it is possible to overcome this limitation. When video object detection is done without anticipating future object locations, are the background-changes important? Essentially, video object detection means detecting the target or foreground object(s), not the background. Removing or masking the background can improve the detection accuracy and can make the model transferable to different datasets. Some existing works [1–6] have explored the relationship between foreground and background in object detection, which seems to indicate that background information may not be informative if motion does not play a role. In terms of video action recognition, it is also unclear if the foreground contains the action information or the background, or even both are essential in action recognition. More research should focus on answering what is the role the background, and in specifically if background changes over time are important for video understanding tasks.

### 6.2.2. WHY DOES PREDICTING FUTURE LOCATIONS HELP IMPROVE DETECTION FOR THE STATIC FRAME?

One interesting phenomenon I observed in my experimental results on the proposed video object detection method in Chapter 2 is: the object detection accuracy for the static frames is improved by learning to predict future object locations. We hypothesize in the article that this is due to the extra supervision the static frame receive. However, it is unclear why motion supervision should improve static-object detection. A related questions is: does this hold only for moving objects, or does the detection accuracy improve also for static objects? This observations can be extended to a more general one: does anticipating a future state (*e.g.* motion, appearance, position, *etc.*) helps improve the accuracy of the current state predictions, across various tasks. And if this is the case what is the reason behind it?

### 6.2.3. ACTION RECOGNITION FOR INHOMOGENEOUS VIDEOS

In the context of video action recognition, this thesis shows that one can accumulate the frame activations along the temporal dimension, by clustering, when the frames are similar semantically. This seems to perform well in practice on videos that contain actions that are homogeneous – actions that are continuous over sequences of frames, without interleaving. In practice, inhomogeneous videos exist. On these videos one may need a very large number of clusters, where multiple clusters would map to the same action. An interesting future research direction is to learn to detect the homogeneous segments, inspired from video segmentation [7–9], and adaptively process the video segments.

## 6.3. Final words

This thesis explores techniques to improve deep network efficiency on video and image data. The thesis demonstrates the importance of leveraging video frame redundancy, and reducing reliance on large annotated datasets. By incorporating these strategies into the proposed methodologies, I have successfully achieved notable gains in various tasks. With this thesis, I made one step further towards the development of more efficient and effective deep neural network models for video and image analysis.

# Bibliography

[1] C.-D. Xu, X.-R. Zhao, X. Jin, and X.-S. Wei. "Exploring categorical regularization for domain adaptive object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11724–11733.

[2] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto. "Unsupervised moving object detection via contextual information separation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 879–888.

[3] B. Li, Z. Sun, and Y. Guo. "Supervae: Superpixelwise variational autoencoder for salient object detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 8569–8576.

[4] Q. Qian, L. Chen, H. Li, and R. Jin. "Dr loss: Improving object detection by distributional ranking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12164–12172.

[5] Y. Pang, X. Zhao, T.-Z. Xiang, L. Zhang, and H. Lu. "Zoom in and out: A mixed-scale triplet network for camouflaged object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2160–2170.

[6] Y. Zheng, R. Huang, C. Han, X. Huang, and L. Cui. "Background learnable cascade for zero-shot object detection". In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[7] S.-H. Gao, Q. Han, Z.-Y. Li, P. Peng, L. Wang, and M.-M. Cheng. "Global2local: Efficient structure search for video action segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16805–16814.

[8] G. Lorre, J. Rabarisoa, A. Orcesi, S. Ainouz, and S. Canu. "Temporal contrastive pretraining for video action recognition". In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2020, pp. 662–670.

[9] Z. Li, Y. Abu Farha, and J. Gall. "Temporal action segmentation from timestamp supervision". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8365–8374.

[10] X. Shi, Y. Li, X. Liu, and J. van Gemert. "WeightAlign: Normalizing Activations by Weight Alignment". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 9788–9795.

[11] X. Liu, S. Khademi, and J. Van Gemert. "Cross domain image matching in presence of outliers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[12] X. Liu, S. L. Pintea, F. K. Nejadasl, O. Booij, and J. C. van Gemert. “No Frame Left Behind: Full Video Action Recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 14892–14901.

[13] X. Liu, F. K. Nejadasl, J. C. van Gemert, O. Booij, and S. L. Pintea. “Objects do not disappear: Video object detection by single-frame object location anticipation”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2023.

[14] O. Strafforello, X. Liu, K. Schutte, and J. van Gemert. “Video BagNet: short temporal receptive fields increase robustness in long-term action recognition”. In: *4th Visual Inductive Priors for Data-Efficient Deep Learning Workshop*. 2023.

# ACKNOWLEDGEMENTS

Reflecting on this wild ride, I find my Ph.D. journey to be both long and short. It has been a great experience to pursue my Ph.D. in the CV lab and the SPRCV group, surrounded by warm-hearted people who support, inspire, motivate, and encourage each other. I would like to sincerely thank all of you.

To my talented group of supervisors, **Jan**, **Silvia**, **Fatemeh**, and **Olaf**, I feel incredibly lucky to have had you guide me through my Ph.D. journey. **Jan**, I deeply appreciate the fundamental values of independent research that you instilled in me, which I will always keep in mind in my future career. You are a brilliant supervisor with countless ideas, yet you never forced me into a project but rather let me pursue what I am passionate about and supported me. **Silvia**, thank you for always being there to help me overcome obstacles throughout my research. You are an exceptionally patient supervisor. I still remember the days when you helped me improve the illustration of a figure in the paper, walked through the code to debug it together with me, and organized my random pop-up ideas. **Fatemeh**, I appreciate your warm and kind assistance during my difficult times. You always took the time to help me with my research, even after you left the company last year. **Olaf**, thank you for your ongoing support and supervision even after you left the company. I feel lucky to have had you as my supervisor, someone who could promptly identify errors in the mathematical part of my draft paper.

**Marcel**, my promotor, I was amazed by your logical expression and fast understanding of my research progress at the progress meetings. Thank you for your guidance in planning my research progress and for your advice from a high-level perspective that helped me make decisions.

**Ombretta**, **Attila**, **Robert-Jan**, my office mates before COVID times, it was wonderful that we all started our Ph.D. at almost the same time and worked on our Ph.D. in the same office. **Ombretta**, it was nice that we worked on the same topic and had a collaboration, which resulted in a publication in the end. **Attila**, thank you for lending me your bass guitar. I practiced it now and then and will continue learning it. **Robert-Jan**, it was fun to have you as the host of events.

**Xiangwei**, **Chengming**, **Hesam**, I am happy to have had you as my new office mates for the last one and a half years of my Ph.D. **Xiangwei**, it was nice to see you back. It was full of fun when we were in the office. I will miss those good times. **Chengming**, it's such a coincidence that we come from the same province in China. We should hang out more often, whether we are in the Netherlands or Sichuan. **Hesam**, I was surprised by the level of spicy food you tried with the three of us. It was a wonderful trip with you when we were in London for the conference.

**Yunqiang**, it was great to collaborate with you when I just started my Ph.D.. Thank you for the career advice. **Xucong**, you are very kind and helpful in listening to my worries when I was searching for a job. Thank you for your valuable suggestions and advice. **Ziqi**, it is great to know that things are going well in London. All the best. **Stephanie**, thank you for the conversation we had when I was checking the job market. **Tom**, I hope to hear your infectious laughter again when I visit the lab now and then. **Chirag**, it was wonderful to have those interesting conversations when we met in the office hall. I am amazed by your talent for starting a conversation easily and naturally. **Yeshwanth**, it was great to work together with you as a TA for the deep learning course. **Jose**, I enjoyed our chats when you were in the office.

**Arman**, it was fun and happy to see you come to our office and "bother" Xiangwei. **Yuko**, I feel you are such a kind person; I wish I had the chance to talk to you more. **Taylan**, it was nice to be office mates with you for a short time. **Jin**, **Amogh**, **Nicolaas**, **Marian**, it was fun to go to the ICCV conference together with you in South Korea in 2019. **Jim**, **Richkard**, thank you for organizing the retreat; it was fun. **Osman**, you are like an older brother in the CV lab; thanks for the interesting brainstorming in the CV lab meeting. **Casper**, it was fun to explore Guimaraes with you for the spring school in Portugal. **Sander**, **Alejandro**, **Aurora**, you joined the lab when I was about to leave. All the best on your Ph.D. journey. **Yancong**, your research on geometric structures is interesting. **Meng**, I hope your work and life in Mianyang are going well.

**Seyran**, thank you for your guidance for my master thesis and the suggestions you gave me when I started my Ph.D. in the lab. **David**, **Marco**, **Jesse**, **Hayley**, **Nergis**, thank you all for providing critical and valuable comments to junior researchers in the lab.

**Ruund**, thank you for being so helpful and kind all the time. You quickly solved my concern by replacing my old laptop recently. **Bart**, thank you for helping to report my broken laptop to IT and having it fixed.

I also would like to express my deepest appreciation to my parents! Your love, support, understanding, and encouragement throughout my life helped me reach where I am today. Without you, I would not have achieved this milestone. I dedicate this accomplishment to you.

Last but not least, **Yucheng**, you have been an essential source of support and encouragement throughout my Ph.D. journey. Your unwavering belief in my abilities and your willingness to listen to my frustrations and share worries and sorrows have been invaluable in my life. Thank you for your love, support, and understanding over the past years, making me a better person. I am fortunate to have you in my life, and I look forward to the future we will experience together.

# Curriculum Vitæ

**Xin Liu**

## Education

|  |  |
|---|---|
| 2019–2023 | Ph.D. in Computer Science<br>Delft University of Technology<br>Delft, Netherlands |
| 2016–2018 | Master in Embedded Systems<br>Delft University of Technology<br>Delft, Netherlands |
| 2011–2015 | Bachelor in Electrical and Automation Engineering<br>Harbin Institute of Technology<br>Harbin, China |

## Work Experience

|  |  |
|---|---|
| 2023- | Senior Research Scientist<br>Danone<br>Utrecht, Netherlands |
| 2018-2019 | Software Engineer<br>ING<br>Amsterdam, Netherlands |

# List of Publications

- X. Shi, Y. Li, X. Liu, and J. van Gemert. “WeightAlign: Normalizing Activations by Weight Alignment”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 9788–9795
- X. Liu, S. Khademi, and J. Van Gemert. “Cross domain image matching in presence of outliers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0
- X. Liu, S. L. Pintea, F. K. Nejadasl, O. Booij, and J. C. van Gemert. “No Frame Left Behind: Full Video Action Recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 14892–14901
- X. Liu, F. K. Nejadasl, J. C. van Gemert, O. Booij, and S. L. Pintea. “Objects do not disappear: Video object detection by single-frame object location anticipation”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2023
- O. Strafforello, X. Liu, K. Schutte, and J. van Gemert. “Video BagNet: short temporal receptive fields increase robustness in long-term action recognition”. In: *4th Visual Inductive Priors for Data-Efficient Deep Learning Workshop*. 2023