Recognition and Modeling of Manipulation Actions

Nachwa Abou Bakr

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Nachwa ABOU BAKR

Thèse dirigée par **James CROWLEY**, Professeur, Université Grenoble Alpes et codirigée par **Rémi RONFARD**, Directeur de Recherche et Responsable Equipe, Université Grenoble Alpes

préparée au sein du **Laboratoire d'Informatique de Grenoble** et de l'**Inria** dans l'**École Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique**

Recognition and Modeling of Manipulation Actions

Reconnaissance et modélisation des action de manipulation

Thèse soutenue publiquement le **12 juin 2020**,
devant le jury composé de :

M. Bernt Schiele
Professeur au Max-Planck-Institut Informatik, Rapporteur

M. Christian Wolf
Associate Professeur à l'INSA-Lyon, Rapporteur

M. Daniel Roggen
Associate Professeur à l'Université of Sussex, Examinateur

Mme. Marie-Christine Rousset
Professeur à l'Université Grenoble Alpes, Présidente

M. James Crowley
Professeur à Grenoble Institut National Polytechnique, Directeur de thèse

M. Rémi Ronfard
Professeur à l'Université Grenoble Alpes, Co-Directeur de thèse

Mme. Agnès Giboreau
Directrice de Recherche à l'Institut de Paul Bocuse, Invitée

Acknowledgements

I would like to express my sincere gratitude to the award of this research grant funded by the *LabEx PERSYVAL-lab*.

However, the completion of this thesis would not have been possible without the expertise of Prof. *James Crowley* and *Remi Ronfard*, my thesis directors and for their wisdom and encouragement.

A great thanks to *Bernt Schiele* and *Christian Wolf* who accepted reviewing this manuscript and helped in improving it. Thanks as well to the examining committee for thoroughly evaluating my thesis.

I would like also to thank my colleagues in the team Pervasive at Inria: *Dominique Vaufreydaz* and in particular, the PhD students who we shared the journey in the same time: *Amr*, *José*, *Raphael*, and *Thomas*.

All my office mates with whom we shared very memorable, interesting discussions and coffee breaks: *Ivana Schenk*, *Juan*, *Maxime Belgodere*, and *Pierre*.

I extend my gratitude to my close friends for their support. It took a long trip with a lot of great memories. *Amaya*, *Dann Wynen*, *Konstantin Shmelkov*, *Nur Zakri*, *Sandra Nabil*, *Stéphane Lathuilière*, *Victoria Abrevaya*, *Vilma*, *Vlad*, *Youna*.

I am extremely grateful to my family for their generous patience and understanding through this thesis. *Bachoura*, *Ghias*, *Hamda* and *Rouida*.

At last but not least, a debt of gratitude I owe to *Etienne Balit* throughout this trip step by step, for the amazing discussions, his generous support and continuous care.

Contents

List of Figures

List of Tables

Abstract

Manipulation actions transform manipulated objects from some pre-existing state into a new state. This thesis addresses the problem of recognition, modelling of human manipulation activities. We study modelling manipulation actions as state transformations. We describe results on three problems: (1) the use of transfer learning for simultaneous visual recognition of objects and object states, (2) the recognition of manipulation actions from state transitions, and (3) the use of reversible actions as data augmentation technique for manipulation action recognition.

These results have been developed using food preparation activities as an experimental domain. We start by recognizing food classes such as tomatoes and lettuce and food states, such as sliced and diced, during meal preparation. We use multi-task learning to jointly learn the representations of food items and food states using transfer learning. We model actions as the transformation of object states. We use recognised object properties (state and type) to learn corresponding manipulation actions. We augment training data with examples from reversible actions while training. Experimental performance evaluation for this approach is provided using the *50 salads* and *EPIC-Kitchen* datasets.

Résumé

Les actions de manipulation transforment les objets manipulés d'un état préexistant en un nouvel état. Cette thèse aborde le problème de la reconnaissance, de la modélisation des activités de manipulations humaines. Nous décrivons nos résultats sur trois problèmes : (1) l'utilisation de l'apprentissage par transfert pour la reconnaissance visuelle simultanée d'objets et de leurs états, (2) la reconnaissance d'actions de manipulation à partir de transitions d'états, et (3) l'utilisation d'actions réversibles comme technique d'augmentation de données pour la reconnaissance d'actions de manipulation.

Ces résultats ont été développés en utilisant les activités culinaires comme domaine expérimental. Nous commençons par reconnaître les ingrédients (comme les tomates et la laitue) ainsi que leurs états (tranchés ou coupés en dés par exemple) pendant la préparation d'un repas. Nous utilisons l'apprentissage multitâche pour apprendre conjointement les représentations des ingrédients et de leurs états selon une approche par transfert d'apprentissage. Nous modélisons les actions en tant que transformations d'états d'objets. Nous utilisons les propriétés reconnues des objets (état et type) pour apprendre les actions de manipulation correspondantes. Nous augmentons les données de formation avec des exemples d'actions réversibles pendant la formation. L'évaluation expérimentale de cette approche est réalisée en se servant des jeux de données *50 salads* et *EPIC-Kitchen*.

Chapter 1

Introduction

In this thesis, we are interested in the task of automatic detection and recognition of manipulation actions. Most current work on action recognition views an action as a spatio-temporal pattern. Manipulation actions, however, are different, as they are generally undertaken to effect a change on the environment. The action is only considered to have been successfully performed if the environment has been changed in the desired manner. Thus, recognition requires a state based approach in which an action is recognized as a transformation from a pre-existing state to a resulting end-state.

The techniques developed in this thesis can be useful for applications such as life-logging developing memory prosthetic for cognitively impaired individuals. In particular, this can provide an important enabling technology for an intelligent collaborative assistant that could monitor complex activities such as food preparation and medical procedures to offer assistance and advice, and to warn of inappropriate or erroneous actions. Such a technology may also be useful for disassembly and assembly for repair, assembling DIY (Do-It-Yourself) kits for furniture and other household objects, and monitoring of cleaning in restaurants, hotels and health-care institutions for certification.

1.1. Action recognition from state transformations

Early approaches to action recognition have been strongly influenced by the nature of available benchmarks. These benchmarks concentrated on action domains where motion is an essential property of the action, such as sport and locomotion actions [47, 79]. Object-interactions are limited in these benchmarks. As a result, much of the current literature concentrated on techniques that recognize actions on the single level, such as motion patterns and appearance, without regard for its effect on the environment. These approaches include motion energy [9], space-time interest points [48], dense trajectories [86], and recently deep learning approaches that uses 3D convolutional networks [35]. But, is action recognition all about motion?.

Human manipulation actions change the situations of the entities in the scene. In many activities, such as cooking or cleaning, entities include not only solid objects, but also ingredients such as liquids, powders or finely cut foodstuffs. The important information for recognizing the activity involves changes in situation, which can be accomplished with one or a series of events.

Recognizing certain actions from motion only map produce incomplete descriptions and cannot discriminate whether certain actions are real or imitations. We believe that action recognition task need to be studied not only on the signal-level (using motion and appearance only) but also on the situational-level (studying objects and relations in the scene).

Going beyond motion, some works have investigated approaches to represent actions as transformations of the scene from preconditions into post-conditions [88]. Other approaches learn relations between objects in the scene and the environment [7]; relations such as co-existence of objects and object interactions. We take inspirations from these works to study object-state transformations for the problem of action recognition beyond motion.

Cooking provides an ideal domain for the study of human actions and activities. Our goal is to monitor and analyze the behavior of a person as they prepare a meal by observing the effects of their actions on foodstuffs. We address this task by first detecting and locating ingredients and tools, then recognizing actions that involve transfor-

mations of ingredients, such as "dicing tomatoes", and use these transformations to segment the video stream into visual events.

In this thesis, we ask the following questions. *What is the most effective problem formalization for detecting food objects and their states?*. *What type of information about objects is useful for action recognition task?*. *How to build an end-to-end model for action recognition using state-transformations?*.

We have explored the use of common network architectures to provide descriptions of entities and relations. To minimize requirements for training data we have preferred pre-trained architectures that support transfer learning as well as weakly supervised learning techniques. Then, we used these scene object state descriptors to detect the happening of an event as changes of the scene situations.

We have found that recognition networks pre-trained on image classification tasks are well adapted for describing activities as a series of state changes in the relevant entities. We have used class activation maps to determine the location of entities. We have extended available datasets with new images and new object classes that represent common entities and situations found in cooking. For evaluation, we used 50 salads dataset to report on classification and localization of food objects and their states. Changes in state and location indicate that actions have been performed. These can be encoded as a series of state changes expressed using predicates whose arguments are the relevant entities such as foodstuffs.

We discuss modeling a manipulation action as an event that transforms an object from a pre-existing state into a new state. Thus we can say that the action causes a change in the state of the corresponding object. We defined the state of entities in the scenes as a transition function from pre-state to post-state within trimmed video clips. We used these defined states in learning to estimate the state of a new entity in the scene (Figure 1.1). We also showed that this definition enables the possibility to reverse some actions for data augmentation while training.

We participated in the challenge of action recognition organized by the community of EPIC-Kitchen dataset [14]. Our participation allowed us to compare our method with other frame-based action recognition techniques such as 2SCNN [76] and TSN [87].

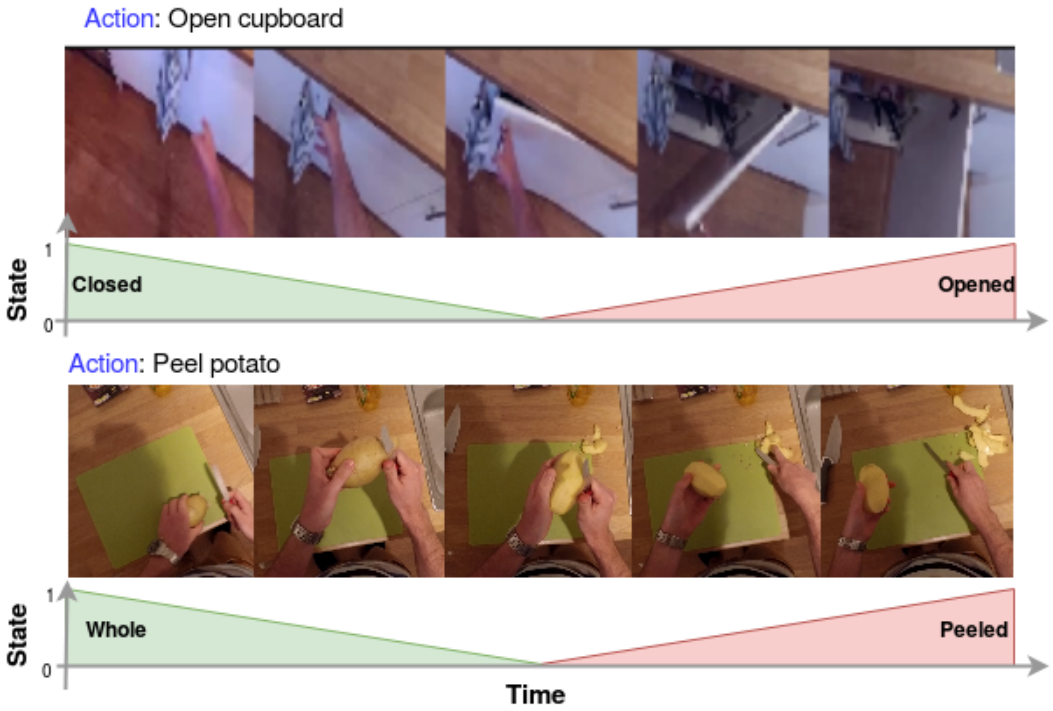


Figure 1.1: Changes in object states over time for action recognition. Two sample sequences from the EPIC kitchen dataset.

Our method for recognizing manipulation actions using state transformations was found to provide comparable performance to these techniques, outperforming in some cases and providing similar performance in others.

1.2. Thesis contributions

This work discusses an approach to recognition of human manipulation actions using the transformations of the state of objects in the environments. It shed lights on the different challenges that this process may face; starting by object recognition and their states to action recognition. Here we summarize the main contributions accomplished within the course of this thesis.

- To recognize food objects, we proposed to learn to label and localize food objects and their states in a weakly-supervised manner. We showed that learning jointly

food objects and states helps in sharing features between these two labels and achieved better performance than learning to classify each label independently. This work resulted in collecting a new image dataset and extending an existing food state dataset by annotating its food object labels.

- We performed an *oracle* study on the contribution of objects to the task of action recognition. This study helps in demonstrating how much certain object information contributes to the task of action recognition. We tested various types of information related to objects such as the order of which objects appear to the scene, the time when they appear, the spatial position, and their state. This study helps in assessing the upper bounds on performance for this object-related information. We also investigated different types of network architectures such as MLP, RNN, and CNN.

- We study the recognition of actions through learning states transformations of objects involved in the clip. For that, we extended the trained model on food objects to learn manipulation actions from a small set of frames. With this model, we have participated in the EPIC-kitchen challenge for action recognition.

- We also showed that modeling actions as state-transformation enables finding reversible actions. These reversible actions can serve as data augmentation while training. We showed results on training to learn actions with and without reversible actions on EPIC-Kitchen dataset.

1.3. Thesis outlines

This thesis is organized as follows: chapter 2 starts by defining the terms used in this manuscripts and a short background into machine learning and specifically deep learning. In the light of this background, this chapter then explains different object recognition tasks and a general framework to object recognition used in computer vision methods. After that, we review the current approaches to the problem of action recognition, and we organized approaches according to the circumstances and problems for which each method is appropriate, with particular attention to manipulation actions.

In chapter 3, we describe experiments with techniques for locating foods and recognizing food states in cooking videos. We discuss two problems of object recognition that we approach in this work: intra-class variation and weakly object localization. We propose a neural network architecture for detecting food objects and their states in a weakly-supervised manner. The model's backbone is a VGG network pre-trained on the ImageNet dataset. For training, we describe the production of a new image dataset that provides annotation for food types and food states. We compare results with two techniques for detecting food types and food states and show that training the model to jointly recognize food type and food state improves recognition results. We use this model to detect composite activation maps for food objects and evaluate the model on frames from 50 salads dataset. This chapter ends with an ablation study on the number of VGG layers in the backbone model.

In chapter 4, we start by a review of different approaches to manipulation action recognition. We report on an *Oracle* for recognition of manipulation actions from only object related information. In particular, this study examines the extent to which object information contributes to the recognition of manipulation actions. We show through this analysis that temporally ordered object lists along with object states achieved the best performance. This analysis motivated our work to investigate about manipulation actions as they transform objects from an initial state into a final state.

In chapter 5, we report on the use of object state transitions as a mean for recognizing manipulation actions. This method is inspired by the intuition that object states are visually more apparent than actions from a still frame and thus provide information that is complementary to spatio-temporal action recognition. We start by defining a state transition matrix that maps action labels into a pre-state and a post-state. From each sampled frame, we learn appearance models of objects and their states. Manipulation actions can then be recognized from the state transition matrix. This model has been evaluated on EPIC kitchen-action recognition challenge. We also demonstrated how the idea of state transformation can be used to extend another competing technique (TSN [87]). At the end of this chapter, we explain a novel method of data augmentation by reversing actions. We apply this concept to the adapted model of TSN (i.e TSN-State) during training phase and show how this can improve the model performance.

Chapter 6 that summarizes the principal results of this thesis, discusses limitations. We also examines directions and research questions that can be addressed in further studies such as Video Narration and Recipe following.

Chapter 2

Background: Recognizing a Phenomenon

In this chapter, we start by reviewing the definitions of the key terms used in this work and our analysis of manipulation actions such as phenomena, object, and action. We then continue with a background overview of machine learning, and in particular, two very rapidly growing domains object recognition and action recognition. At the end of this chapter, we review the main concepts in convolutional neural networks as they are the main block of our experiments in this thesis.

2.1. Recognizing and observing a phenomenon

What is a phenomenon? A *phenomenon* is anything that can be reliably and repeatedly perceived and described in an input signal. An *entity* is an internal representation for a phenomena, generally including a category label, that enables associations with other entities, memories or knowledge. Entities are described with properties; which are measurable characteristics that can be visual or functional ones.

What is an object? In the field of visual object recognition, the term object refers to physical objects, generally associated with visually apparent properties such as color, shape, or form. Objects with similar properties form a class or category; thus, objects are instances of these classes. An object class is a commonly shared concept

and properties of a certain set of objects. The words *class* and *category* are used interchangeably, referring to a visually consistent set of objects.

What is an Action? We define an *Action* as a deed performed by an agent to achieve a goal state of the world. An action may be formally defined as a deliberate phenomenon that changes the state of the environment. Action types include manipulation such as fixing a car and kicking a ball, or locomotion actions such as walking and running. By definition, both types inherit the effect of changing the world state. *Locomotion actions*, change the physical location of entities in the environment. Therefore, in many cases, features of local spatio-temporal can be used to represent the properties of these actions (e.g. velocity and acceleration).

In contrast to locomotion, **Manipulation actions** involve interactions with one or more objects. We generally associate this type of action with one or more physical objects, as in "opening a door" and "cutting a tomato with a knife". Grammatically, these are referred to as the subject and object of an expression. This can cause confusion as both the grammatical terms "subject" and "object" refer to "physical objects" in the vocabulary of object recognition. The use of the term "entity" helps to avoid such confusions.

Moreover, the success of this type of action is determined by reaching the object's desired state. Thus, in addition to local motion in the scene, manipulation actions share a set of global features that can be represented by the relations between the involved objects and environment. Examples of state changes include full/empty bottle, open/closed door, and attached/detached car wheel.

Recognition is the process of assigning a category label to a phenomenon. Given the input signal x, output signal w is the association of x with previous experience or knowledge. Recognition requires to retrieve knowledge from an encoded model. With machine learning, recognizers encode patterns from data samples extracted from a certain domain.

2.2. Machine learning

After more than 50 years, machine learning is currently making rapid progress, driven in part by the availability of planetary scale data and high performance parallel computing. A number of algorithms have recently been demonstrated to meet or surpass human performance in specific tasks [28, 75]. Progress in machine learning has recently made possible demonstration of systems with human level abilities in traditionally challenging domains such as natural language understanding and generation, computer vision, robotics, and computer graphics.

Convolutional Neural Networks (CNNs) [50] are particularly well suited to provide feature robust recognition for computer vision tasks such as object and action recognition.

In **Supervised learning**, the goal is to learn the mapping between a set of inputs and outputs. Example labelled data is provided as input and output pairs that are used to estimate a function that transforms the input into the desired output. An important goal is to learn a general mapping that can provide a correct output for novel unseen input. In training, we want to maximize generalization, so the supervised model defines the general underlying relationship between training examples. If the model is over-trained, this can cause *over-fitting* to the examples used and the model would be unable to adapt to new unseen inputs.

If the training data is not a good representative sample of the target domain, then the supervised learning may estimate a function that is biased. The model can only be imitating exactly what was shown, so it is crucial to train on data that accurately represents the entire input domain. Also, supervised learning usually requires many data before it learns. Providing labelled data that adequately covers the entire input domain is the most difficult and expensive challenge for supervised learning.

In contrast to **Unsupervised learning**, where only input data is provided in the training process. There are no labelled examples to aim for. However, it is still possible to find many interesting and complex patterns hidden within data without any labels.

Weakly-supervised learning is a mix between supervised and unsupervised approaches. This category of algorithms makes it possible to mix together a small amount of labelled data with a much larger unlabeled dataset, which reduces the burden of having enough labelled data. Therefore, it opens up many more problems to be solved with machine learning.

2.3. Object recognition

The ability to identify the objects present in a scene is a basic requirement for interacting with the environment. While this task may seem effortless for humans, this is a classic complex problem for computer vision.

2.3.1. Object recognition tasks

The key to understanding visual scenes is four closely related sub-problems. The first and easiest is **image classification** where the task consists of assigning input images with a probability of the presence of a particular visual object class (dog, car, cat, ...). More precisely, given a set of images, $I = I_1, ..., I_n$ and a set of label vectors $Y = y_1, ..., y_K$, $y_i \in \{0, 1\}^{n \times 1}$; where K is the number of classes, the task is to produce a set of output vectors $\hat{Y} = \hat{y_1}, ..., \hat{y_K}$, that matches Y as much as possible.

The second problem is **object detection**, which involves both classifying and locating regions of an image that best describe an object class. The object is usually localized by determining a bounding box around the image region that is occupied by the object.

The third problem is a more demanding one, **semantic segmentation**, which requires providing a precise pixel-wise boundary of each object type in the scene. This problem is called **instance segmentation**, in which the system needs to differentiate between different object instances of the same class in the scene.

The fourth problem is **image captioning**, which produces a literal description of the image in a form of phrases or sentences that characterize the objects in the scene and their properties.

2.3.2. Object recognition systems

Since the early 1960's, attempts to solve the object recognition problem have contributed to the creation of the field of Computer Vision. Visual recognition systems in general start by extracting low-level features (e.g. edges, corners and textures). While these features alone may not be enough to draw conclusions regarding the image content, they serve as an input to a more complex decision process.

Interestingly, this dataflow is similar to the way our brain processes data coming from the eyes. The brain contains several layers of neurons dedicated to different low-level processes. Those layers compose different regions such which are known to provide early image description using convolution with receptive fields over a range of scales and orientations [72, 62].

For computers, the decomposition of the recognition process plays an important role to simplify appearance variation problems of objects. The same object may appear very different under small changes in illumination or viewpoint. For an object recognition system to be usable, it needs to be invariant to disruption sources (i.e. illumination, noise, scale, intra-class variation, rotation, ...). A two-step decomposition enables an easier sharing of this burden; illumination, translation, scale, rotation, can be handled by the abstraction of the image through low-level features (edges, corners, ...) while intra-class variations and noise are dealt with by the decision process [67].

For several years, systems of detection and recognition in the computer vision field used several layers of hand-crafted features. These include two-dimensional filters used to detect simple structures such as corners and edges, or spatial pyramids to model scales [13]. Based on this information, a higher level of object detection is calculated. The problem of these approaches is that the filter types have to be chosen and optimized by hand. In contrast, CNNs allow to learn and optimize the filters automatically.

In 2009, ImageNet image dataset [15] was released: a dataset with over 15 million labelled images that belong to approximately $22K$ different categories. The dataset authors organize every year an object recognition challenge, ImageNet Large Scale Visual Recognition Challenge (ILSVRC), to classify a subset of $1K$ classes of the dataset.

In 2012, a deep convolutional neural net named AlexNet [45] achieved an error rate of 15.3%, compared to an error rate of 25% for the nearest competitor. This marked a paradigm shift toward use of deep neural networks for computer vision.

Since then, the field of computer vision has made rapid progress with the advances in both computational hardware devices as well as the easy access to digitized data. These advances resulted in many successful frameworks for the task of object detection. Some followed a traditional object detection pipeline, generating at first region proposals of objects and then classifying each proposal into different object categories [29, 65, 23]. Other frameworks regarded object detection as a regression or classification problem, adopting a unified framework to achieve the final output directly in form of categories and locations [64, 54].

2.4. Action recognition

Action recognition is the study of describing video clips with their semantic contents. The main property in this field is the existence of a new dimension to the input data: the temporal dimension. These frames are generally correlated spatially and temporally. Thus, visual recognition from videos can use both motion and appearance information. In this section, we start by defining different visual tasks of action recognition. Then, we examine a general action recognition framework and explore different approaches to action recognition.

2.4.1. Action recognition tasks

Similarly to object recognition, the domain of visual action recognition has been studied through different sub-tasks. **Action classification** considers already trimmed video clips, and the recognition task is to define what action class each clip belongs to from a closed set of action classes. In contrast, **Action detection** refers to the localization of temporal boundaries in the video clip that best surround a known action. This task sometimes refers to the spatial localization of the action in the scene as well as the temporal one [94]. A more challenging problem is **video description**,

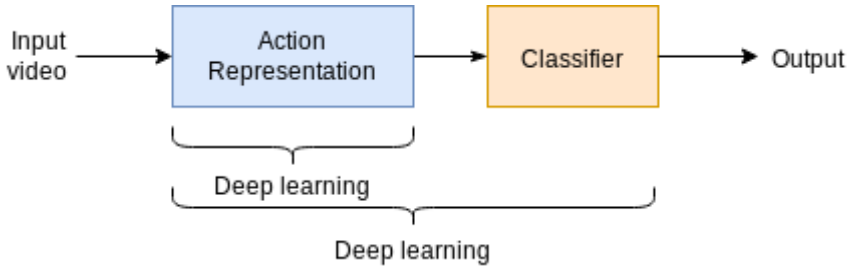


Figure 2.1: General framework of Action Recognition System. If deep learning is used it can be at the level of action representation only or at the level of the whole pipeline in end-to-end settings.

an ambitious problem that takes as input a sequence of frames and produces a caption that describes the visual content in a sequence of words or phrases.

2.4.2. Action recognition systems

The first important question in this field is how to represent actions? Actions appearing in videos inherit all difficulties of a 2D image (camera view, appearance, translation, intra-variation, etc.). In addition to these difficulties, actions differ in their temporal dimension (e.g. motion speed, velocity, etc.), making action representation a challenging problem. Kong et al. defined a successful action representation method to be efficient to compute, effective to characterize actions, and can maximize the discrepancy between actions, to minimize the classification error [43].

Many literature reviewers prefer to classify methods for action representation by separating methods into two eras; one before the evolution of deep learning and one using deep learning techniques [43, 97]. While deep learning methods currently outperform traditional methods, the general framework of action recognition has not changed a lot. Some methods replaced hand-crafted features of action representation into deep features, and others include the feature classification step with the representation learning process (see Figure 2.1).

Traditional methods for action recognition described an action either with global features or with local ones. Global representation of actions extracts global features from the whole scene such as silhouette-based features, then describes smaller patches

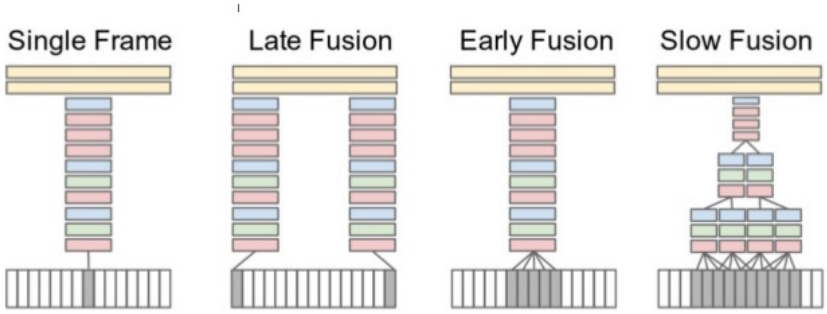


Figure 2.2: Fusion of temporal information into spatial. Illustration originally appeared in [40].

inside. These are limited to camera motion and require preprocessing such as background subtraction, foreground extraction. Conversely, local representation starts by detecting interest points and describe surrounding areas. Hand-crafted local representations have been shown to be effective for action representation and delivered state-of-the-art performance before deep learning era. Examples of these methods are spatial-temporal interest points described by spatio-temporal Gabor energy filters [12], 3D-SIFT [73], HOG3D [42], HOGofDepth [6], Local Trinary Patterns [98], and Dense trajectories [85].

Deep learning methods for action representation can be split according to the moment when the features of the temporal dimension are fused with the spatial features (see Figure 2.2). Some methods represent actions by describing spatio-temporal patches of the videos right from the beginning of the representation. Conversely, others use a sequence of 2D representation of single frames and aggregate results of all frames only at the end. Between these two methods lies a spectrum of methods that fuse information from frames at different stages in the representation process.

The success of certain neural architectures for object representation motivated the idea of extending existing image recognition architectures to video-based applications. In practice, 2D filters are inflated into 3D filters to include the temporal dimension of the problem while learning action representation [36, 37]. The main expected issue is the huge number of training parameters that results from adding a new dimension to the convolutional kernels.

However, not every image recognition method can be generalized to video applications. For example, image representation of bag-of-words which study a histogram of visual words has proven its power for object classification. This idea has been used for videos classification by extracting features independently from each frame then aggregating the results over all the video segment. Those models by design are not able to learn the action's temporal direction (e.g. opening and closing actions) as they entirely ignore temporal structure.

To keep track of the temporal structure of the video without exploding the number of model parameters, some models use existing image-based models and aggregate the results of the prediction of single frames [87]. Other researchers proposed to follow spatial information with a more sophisticated structure that captures temporal correlations in the video clip. In practice, this can be accomplished by adding 1D temporal kernels on top of 2D spatial kernels which are known as 2.5D ConvNets [91], or following spatial information with a Recurrent Neural module [17, 100]. This way of fusing the information still uses the knowledge learnt from the spatial domain while gradually incorporate the temporal domain as features flow deeper in the network (Figure 2.2).

Researchers have also explored the use of two-stream networks [87, 76, 38, 10, 78] in which one stream is used to analyze image appearance from RGB images and the other represents motion from motion images (e.g. Optical Flow, RGB difference). Both streams are then aggregated to generate a representation of the video clip. These approaches provide spatio-temporal analysis while avoiding the substantial increase in training parameters.

2.5. Convolutional neural networks

Convolution Neural Networks (CNNs), or ConvNets, have been investigated since the late 1980s and 1990s. Using neural networks to recognize handwritten zip codes or document recognition are well-known successful case studies of this concept being used in the early days [50].

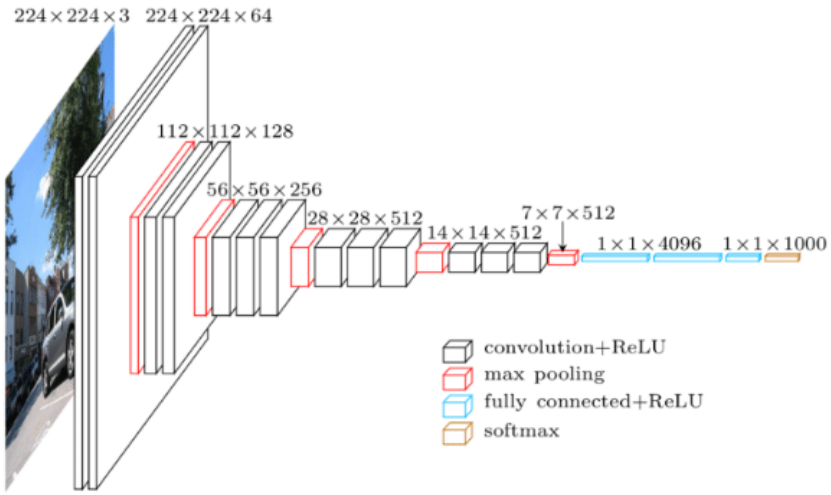


Figure 2.3: The architecture of VGG16 model.

One typical CNN architecture is VGG [77], developed and trained by the Visual Geometry Group in Oxford (Figure 2.3). This CNN architecture consists of a sequence of interleaved layers of convolution and pooling which constructs an initial feature hierarchy. These features are then used in the decision function modelled by one or more fully connected layers (FC).

In the following, we explain the underlying layers of this CNN architecture briefly. In particular, a convolutional layer, a pooling layer, and a fully connected layer. Then, we explain concepts that are widely used in computer vision as well as this work: fine-tuning and transfer learning.

2.5.1. Convolutional layer

The basic idea of CNN was inspired by a concept in biology called the receptive field. Analysis of visual cortex of cats and monkeys showed that these receptive fields are sensitive to certain types of stimulus such as edges and bars [32, 21]. For example, a receptive field in the retinal ganglion cell is arranged into a central disk, and a concentric ring where each responds oppositely to the light. Visual information is then passed from one cortical area to another. Each of them is more specialized than the last one. This operation can be formulated as convolution [57]. Differential edge

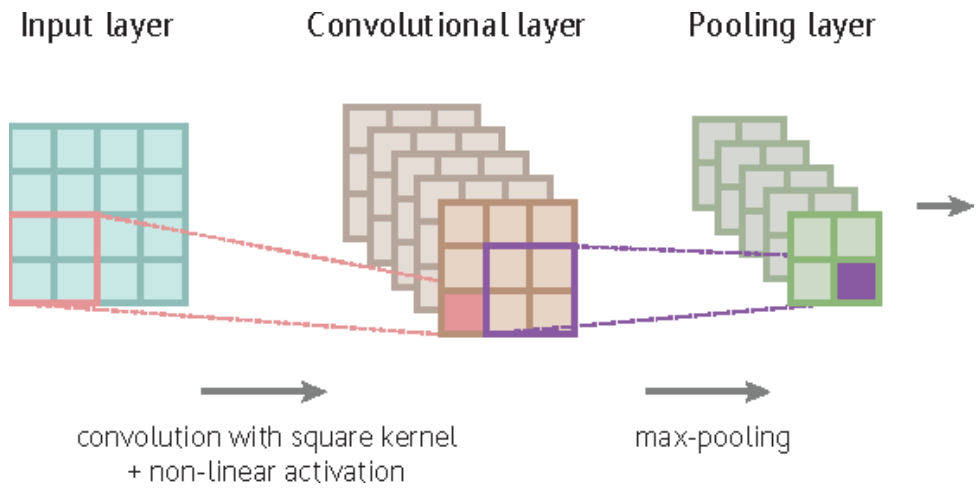


Figure 2.4: Maintaining relative spatial locations of the features throughout the CNN

detectors have been used in computer vision systems such as Canny and Sobel filters. These filters are convolved with the image to produce the image response to that filter. A similar operation is used in CNNs with the difference that the receptive field weights are learnt during a training process.

In practice, every neuron of a convolutional layer is connected with a small portion of adjacent neurons from the previous layer. The output matrix h of the convolution operation on a 2D image I with size of $x * y$ and a convolution filter g of size $u * v$ is defined as:

$$h[x, y] = I[x, y] * g[u, v] = \sum_{u,v} g[u, v]I[x - u, y - v]$$

In ANN, the matrix h is called a *feature map*, which is the image response to the convolved filter. Usually, each neuron is followed by an non-linear activation function f and thus the matrix $f(h)$ is called an *activation map*.

The convolutional feature map has the property of maintaining the relative spatial location throughout the network; which means the bottom-left cell in a feature map correspond to the bottom-left region in the original image (see Figure 2.4). Therefore, activation maps of fully convolutional networks (FCN) can be used to have a rough idea of what stimulated the network to take a particular decision.

2.5.2. Pooling layer

One of the ConvNets distinctive concepts is pooling. The idea of the pooling step, or more precisely the spatial pooling, is to reduce the resolution of the feature map, and eliminating noisy and redundant convolutions, and computational overhead yet retaining most of the important information.

The layer input takes a fixed-size patch p of adjacent neurons and applies a pooling operation on p. The pooling operation calculates a specified aggregation function on every patch of the feature maps (see Figure 2.4). This aggregation function can be the Max, Min, Sum or Average and correspondingly the layer is called Max Pooling layer, or Average Pooling Layer. The results of the pooling layer are down-sampled or pooled feature maps. Pooling layers add no trainable parameters to the training.

Another type of pooling layers are **Global Pooling** (GP) layers where a pooling filter is applied over the complete feature map. For example, a feature map with dimensions $w \times h \times d$ is reduced in size to have dimensions $1 \times 1 \times d$. GP layers reduce each $w \times h$ feature map to a single number by simply aggregating all $w \times h$ values. It is an extreme pooling over the whole feature map that can be used to represent the whole feature map with a single number. As in standard pooling layers, the GP operations can be Average or Maximize and therefore called *Global Average Pooling (GAP)* or *Global Max Pooling* respectively. We will see a beneficial application to this type of pooling throughout this thesis.

2.5.3. Fully connected layer

In a fully-connected layer, each neural unit is connected to all of the units in the previous layer. Thus, each neuron has a number of trainable parameters equal to the number of neurons in the previous layer.

A multi-layer network has shown its approximation capabilities of almost any function [31]. However, this comes at the cost of the number of learnable parameters in such a network. Thus, in many classification problems, fully-connected layers are one of the latest layers in the network architecture coming right before the output layer

where each neuron can be used to represent a class. This layer is used to decode the last feature vector in the CNN network into specific classes.

2.5.4. Transfer learning

Developers face two main problems in building a recognition system using supervised learning: the computational cost of training a high number of parameters in a neural network, and insufficient labeled training data for many tasks. However, popular neural networks have been trained on large-scale datasets such as ImageNet [45] and made publicly available for many different tasks such as object detection and classification.

Transfer learning techniques leverage the use of these trained neural network models on large-scale datasets to adopt them on new tasks. For example, a CNN model trained on the task of image classification has proven to be useful for resolving other problems such as object detection and localization [60]. Indeed, training a deep neural network architecture on certain classes shows that bottleneck features can be used as high-level descriptors of data. These descriptors can be then used to classify different classes than the ones the model was originally trained for. In practice, these pre-trained models are used as fixed features extractors. These fixed features can be classified with simple classification methods to predict new classes.

In computer vision, trained deep neural network learns hierarchical features ranging from local low-level features to domain-specific high-level features. Low-level features are generally common features for image-related tasks. By tweaking high-level features, we can transfer the learnt features to a new task or a new domain. **Fine-tuning** is a transfer learning technique that uses a pre-trained network model as a starting point and allows the network to continue learning to adapt to new tasks. Practically, this is achieved by training a pre-trained model with a slow learning rate on the new task.

2.6. Discussion

This chapter provides a general background for many of the techniques used in this thesis: terminology, machine learning techniques for object and action recognition, and convolutional neural networks.

We defined the terms such as *Phenomena*, and *Object* in the context of computer vision. We also explained different characteristics of *Actions*, including manipulation action which involve object interactions as opposite to locomotion actions. We also defined the *Recognition* process as labeling an observed phenomena as a known category, and discussed how recognition enables description.

Different approaches to machine learning have different advantages and limitations. Supervised learning algorithms demand expensive labeled examples. Weakly-supervised learning techniques leverage labeled data to learn more complex tasks than the ones defined by the given data labels. Machine learning techniques are applied to computer vision problems such as object and action recognition. For each problem, we defined the different sub-problems studied in the literature and brief history of the evolution of the domain.

ConvNets are now widely used for solving many common computer vision problems. At the end of this chapter, we briefly explain the main layers compose a convolutional neural network and different techniques used to leverage the re-use of pre-trained neural networks for new tasks.

In the next chapter, we are reporting on our first contribution regarding the classification and detection of different objects and their states using weakly supervised method and the challenges this can bring.

Chapter 3

Detecting Food Class and Food State

This chapter reports on experiments with techniques for the classification and localization of food classes and food states. This is a challenging problem in computer vision given the considerable variation in the appearance of food under changes to its state.

This chapter starts by discussing the challenges we faced in the task of detecting food class and food state (section 3.1), namely, intra-class variations and the need of training data pairs for supervised detection. Then, we introduce a method of weakly supervised localization of food classes in section 3.2 and discuss two methods for combining different feature maps of food classes for localization. In section 3.3, we review available datasets for studying this task and report on the results of the experiments using our own collected dataset. In section 3.4, we performed an ablation study which shows that jointly learning of food classes and food states results in improved detection and localization of composite food classes.

3.1. Introduction and related works

In this chapter, we approach two issues in typical object recognition systems: intra-class variations and the lack of labeled images for the task of food state detection. We begin with a review of previous work on these problems.

3.1.1. Intra-class variations

Resolving intra-class variations of objects can be challenging. Solid objects can exhibit significant changes in appearance which can complicate recognition even for humans. For example, one would rather describe a food with its appearance attributes such as its colour, shape, state than its class (e.g. in Figure 3.1 three examples of diced, orange, food entity which can be either carrot, pumpkin, or sweet potatoes).

Multiple approaches to distinguish different objects with their attributes have been described in the computer vision and machine learning literature. A direct approach would be to adapt **Mutli-class learning** to this problem by treating each attribute as a new class. For example, sliced tomato and a whole tomato would be considered as two different classes. While this approach has the benefit of simplicity, the number of category sets grows exponentially as the number of different attributes increases.

In contrast, **Multi-label learning** associates each object with a set of possible labels. For example, an object can be a "tomato" and "sliced" at the same time. Multi-label learning supposes a degree of correlations or dependency between labels. According to Zhang and Zhou [101], Multi-label learning can be categorized into multiple families based on the order of correlation between labels. A first-order strategy would decompose the multi-label learning problem into several independent binary classification problems (one per label). A model is learned for each attribute label independently. Although it might achieve a good performance per label, this strategy can not model the correlations between different labels. A high-order strategy would use high-order relations among labels such that each label is considered to influence the co-existence of most other labels.

Figure 3.1: Diced foodstuff: pumpkin, sweet potato, and carrot (from left to right). It can be easier to describe these food entities with their state than their classes.

Multi-task learning (MTL) supposes that the learning of one task relies on or constrains the learning of other tasks. Task, here, refers to a machine learning task, for example, learning each attribute alone as one task. While processing each task independently is prone to ignore such correlations, learning multiple tasks at the same time may exploit commonalities and differences across these tasks. Considering the problem as multiple tasks encourages the learning process to mine feature useful for both tasks as well as task-specific features [11, 89].

Simultaneously learning more than one task means sharing some parameters for the learning of both tasks. Parameter sharing can be either soft or hard parameter sharing. In soft parameter sharing, each task has its own parameters regularized to encourage them to be similar. In hard parameter sharing, a set of parameters are dedicated to being used by all tasks equally; generally, this is applied by sharing hidden layers between all tasks.

MTL learning algorithms are efficient as mentioned in [71] for several reasons: MTL implicitly increases the training data samples since all samples are used to train all tasks (implicit data augmentation). Noisy data can result in the use of irrelevant features. MTL can help by focusing attention on features that are used by other tasks as additional evidence. MLT also helps by choosing representations that serve in solving all tasks simultaneously and ignoring representations do not matter.

3.1.2. Weakly supervised localization

Data annotation is an essential problem in object recognition. While multiple publicly available large-scale datasets are available, building a recognition system for a new object requires going through the annotation process for every new application. This problem becomes more critical when annotation requires labeling object category, position, and other attributes. However, in some cases annotating images with only image-level labels can be easier. Weakly-supervised localization makes effective use of these image-level labels to learn to locate objects in the scene.

In fully convolutional networks (FCN), this localization is provided by the structure of ConvNets. As discussed in Convolutional neural networks in section 2.5, ConvNets preserve the coarse-grained spatial location of the network activations. These ac-

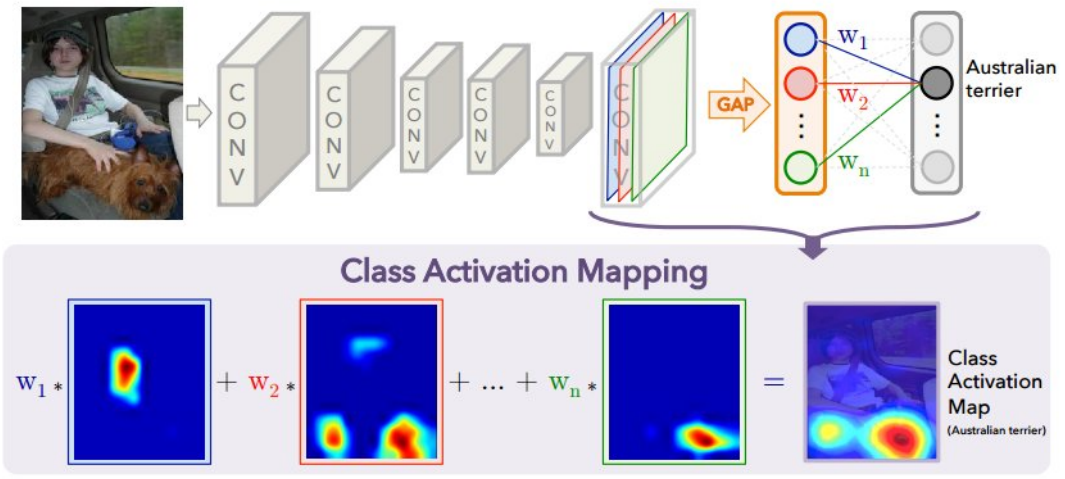


Figure 3.2: Class Activation Maps. Illustration originally appeared in [102]

tivations can be traced back to locate the region of the input image that triggered the network output. Recent works have proposed to use these network activations for modeling network attention to different images classes [61, 74, 102]. Among these works, Oquab et al. [61] has explained that the use of pre-trained ConvNet on classification tasks maintains spatial information. The authors also showed, in their paper *is object localization for free?* [61], that some object localizition can be achieved by evaluating the output of ConvNets on multiple overlapping patches of feature maps.

An end-to-end method has been proposed by Zhou et al. [102] to learn to draw implicit attention of the network to class-specific features while training. In practice, this is achieved by adding a Global Average Pooling layer (GAP) immediately after the last convolutional layer and using the pooled vector as input to the fully connected layer whose task is to decode deep features into object classes. Therefore, each class is sparsely encoded using the feature maps of the last convolution layer. These are referred to as Class Activation Maps (CAM) and shown in Figure 3.2. This setting gives the network a limited ability to encode the network attention to class-specific regions. Thus, CAMs can be used to localize objects in the image while trained in a weakly-supervised manner (i.e. using only image-level labels).

3.2. Learning food concepts

The goal of this chapter is to learn representations for two common food concepts: food class and food state. We use the term "food class" to refer to specific foods such as tomato and cucumber. We use the term "food state" to refer to the shape and the physical appearance of a food class as it undergoes preparation. For example, sliced, diced and peeled are food states.

Classical object detection methods treat an object (e.g. car, door, cat, ...) as one visual class. For food objects, changes in food state can also entail a dramatic change in visual appearances, as well as changes in 3D shape (Figure 3.3). Treating food objects as a multi-class classification problem involve considering each combination of different food classes and food states as one different visual class. This can rapidly increase the number of different categories of objects and therefore make learning to distinguish these classes challenging. Moreover, food objects in different states may have inter- and intra-variability (Figure 3.4) which in its turn adds complexity to the recognition task. Thus, we are looking for a method that can learn to distinguish food objects that can scale up to the increasing number of food classes.

Besides this, due to the rich vocabulary of food objects and activities, the availability of densely annotated cooking video datasets is limited. To the best of our knowledge, at the time of this study, no image state dataset was available. Recently, Jelodar and Sun [34] published an image dataset of food states. For video cooking datasets, Table 3.1 summarizes the size and content of several accessible cooking video datasets, showing

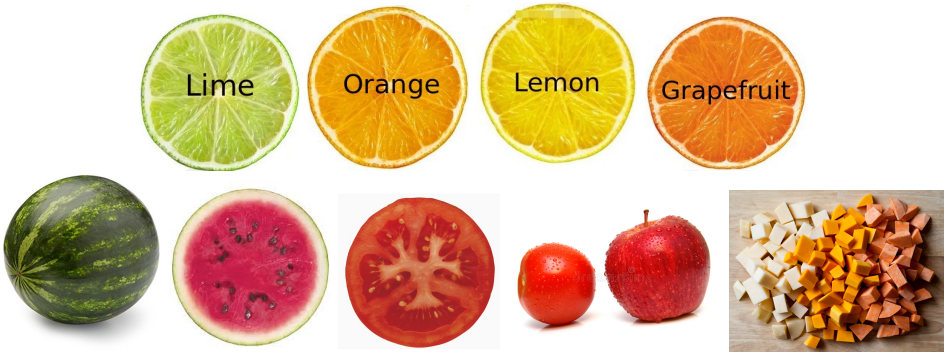


Figure 3.3: Inter and intra variability of Food.

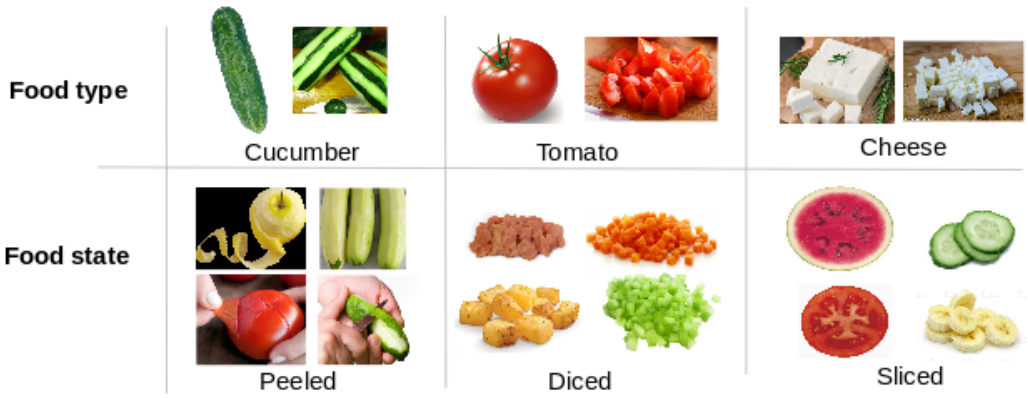


Figure 3.4: Food concepts

Dataset	Recipes	Actions	Object classes	Object states
MPII Cooking v2 [69]	36 activities	Yes	N/A	N/A
50 Salads [81]	1 recipe	Yes	N/A	N/A
Breakfast [46]	non-scripted	Yes	N/A	N/A
KUSK [27]	20 recipes	Yes	23	N/A
YouCook2 [103]	89 recipes	Yes	33	N/A
EPIC-Kitchens [14]	non-scripted	Yes	352	N/A
EGTEA+ [52]	7 recipes	Yes	53	N/A

Table 3.1: Available cooking video datasets for action recognition.

that while action annotations are widely available, food classes are rarely annotated, and food states are never annotated in cooking video datasets.

These observations motivate our decision to collect a dataset from Google images for both food classes and food states. Then learn to classify these concepts with multi-task learning techniques and to localize them in a weakly supervised manner as detailed in the following.

3.2.1. Concepts activation maps

We treat the problem of learning to represent food objects and states as a multi-task problem where learning each attribute is a task. We use the hard parameter sharing technique where the backbone structure is shared to learn common feature representations of all tasks while treat the high-level layers as task-specific layers to learn discriminative patterns for that task (Figure 3.5). Concept Activation Maps are the resulting network activation of this task-specific layer.

To localize objects in a weakly supervised manner, we use Class Activation Maps (CAMs) [102] as an indicator of the image region occupied by a class member. Concepts activation maps are sparse represent for each concept extracted from the shared class activation maps. Similarly to CAMs, we use GAP pooling layers to locate concepts as well. Since GAP has no additional training parameters, replacing fully connected layer by the GAP layer forces the network to tweak its deep feature to learn class-specific features at the last convolutional layer. The number of activation maps of this last convolutional layer equals to the number of object classes. Since GAP pools a 2D image feature map into one scalar, the resulted vector is used directly to compute the loss.

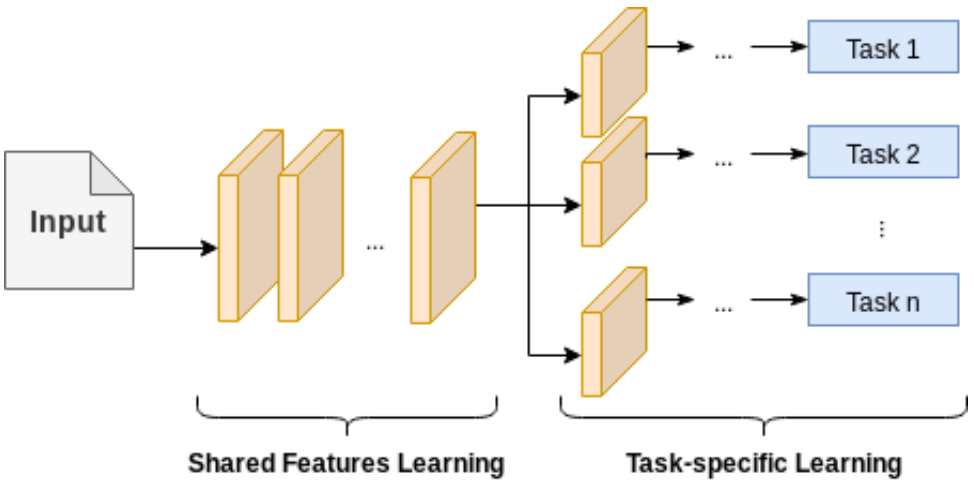


Figure 3.5: Illustration of hard parameter sharing for multi-task learning.

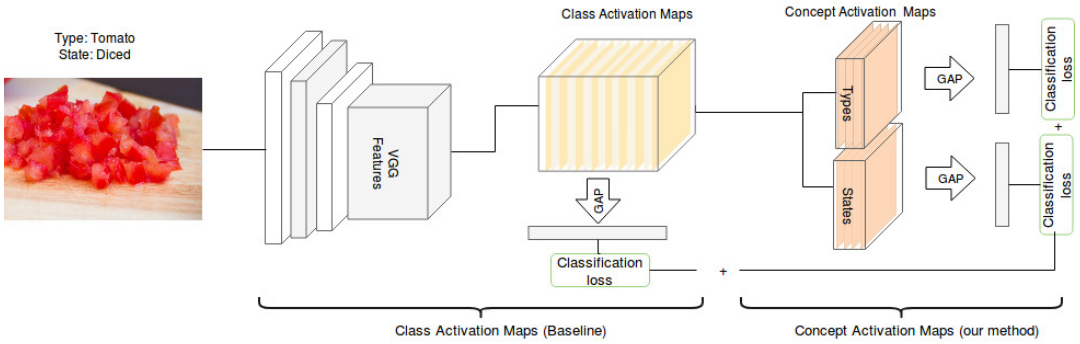


Figure 3.6: During training, we learn food concept maps for food classes and food states from labeled examples.

In practice, concept activation maps are implemented as a depth-wise convolutional layer on top of the CAMs layer. The number of filters of this layer is equal to the number of values a concept can take. The goal of this layer is to decouple localization of food class and food state from the combined examples. For training, we compute a separate cross-entropy loss for each concept. The training architecture is illustrated in Figure 3.6.

3.2.2. Concepts composition

Describing food transformations requires combining recognition of food class and food state. We refer to these as "composite classes". For example, "diced tomato" or "sliced cucumber" are obtained by composing food class concepts (tomato, cucumber) with food state concepts (sliced, diced). After training and for the objective to locate composite classes, we explain here how we combine these concepts together without the need to learn the composite classes during training.

We consider a composite class to be composed of more than one concept. An image region is considered to belong to a composite class if it belongs to all of its concepts. We use two ways to composing concepts: **Product Concept Composition** where each pixel in the corresponding concept maps is element-wisely multiplied. This is defined as:

$$P_{cc}(x, y) = \prod_{c=1}^{n} A_c(x, y)$$

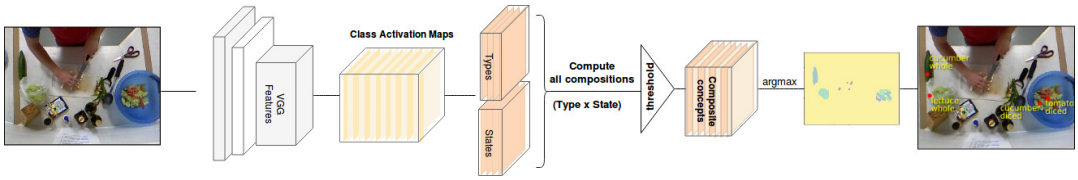


Figure 3.7: During testing, we compute the activation maps from unlabeled examples by composing the learnt concept maps.

where $A_c(x, y)$ is the network activation value for the c^{th} concept activation map at the pixel (x, y), and n is the number of concepts. Second, **Average Concept Composition** where each composite concept map is the element-wise average prediction over the number of concepts. In practice, we define the average composition of concepts as:

$$P_{cc}(x, y) = \frac{1}{n} \sum_{c=1}^{n} A_c(x, y)$$

The number of resulting composite concept maps $= C_1 \times C_2 \times \cdots \times C_n$ where C_n is the number of different classes of the n^{th} concept.

3.2.3. Food localization

We compute the location of a food class in a specific state from its corresponding composite map (P_{cc}). Figure 3.7 summarizes the post-processing for food localization on test images. Firstly, input test images are rescaled and passed to the network. Each of the output predicted concept maps is resized to the size of the original test image. Secondly, composite concept maps are computed for all different combination of concepts. Thirdly, composite concept maps are normalized and filtered as follows:

$$P_{cc}(x, y) = \begin{cases} P_{cc}(x, y), & \text{if } P_{cc}(x, y) >= Threshold \\ background, & \text{otherwise} \end{cases}$$

The threshold is set to 80% of overall activation maps. Therefore, the predicted label of pixel (x, y) is

$$P_l(x, y) = \underset{cc}{\text{argmax}}(P_{cc}(x, y))$$

We compute the surface of connected components from $P_l(x, y)$ image for objects localization. We also compute the pixel-wise detection accuracy.

		cheese diced	*cheese whole*	*cucumber diced*	*cucumber peeled*	*cucumber sliced*	*cucumber whole*	*lettuce diced*	*lettuce whole*	*tomato diced*	*tomato peeled*	*tomato sliced*	*tomato whole*	*Total*
A)	Train	40	38	43	40	59	33	31	31	35	27	59	32	468
	Validate	10	5	5	5	9	4	4	6	4	5	7	4	68
B)	Test	143	11	91	65	112	238	154	249	185	0	53	91	1392

Table 3.2: Collected dataset details. A) Extracted training, validating sample images from Google Images. B) Annotated test samples from selected key-frames of 50 Salads dataset.

3.3. Experimental setup

This section starts with some details about the collection process of image dataset for training, in addition to the annotation of test frames from 50 salads dataset. We then report on our experimental results for jointly learning food class and food state during cooking activities. We compare results of learning food classes and food states as a multi-task problem to learning these composite classes as a multi-class problem. We use as a baseline the original implementation of CAMs [102] as a multi-class learning technique. We experiment on key-frames from 50 salads dataset.

3.3.1. Dataset collection

Training dataset We collected a training set of images from Google Images, using all possible composite concepts as query keywords. Those keywords are considered to be the image-level labels. We manually filtered irrelevant images to get a total of 468 images for training (on average, 39 images per composite class). Details of the collected dataset is listed in Table 3.2(A).

Testing dataset. Since our goal is to recognize actions from videos, our testing images are extracted from a cooking videos dataset, in particular, 50 salads dataset [81]. For our experiment, 50 salads dataset is a suitable dataset as the ingredients appear at different places and different states during the videos. The dataset has a small number

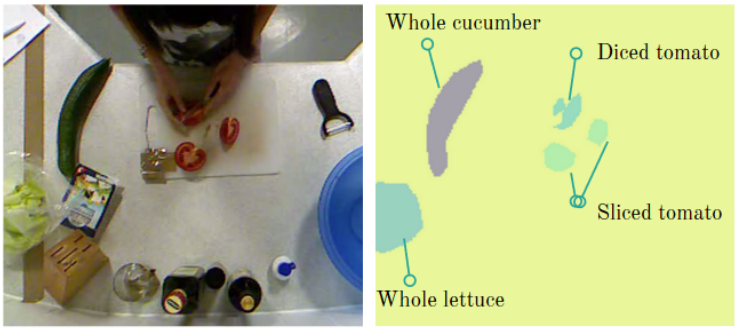


Figure 3.8: Example of labeled frames from 50 salads dataset.

of ingredient classes which facilitates the evaluation, and the recipes are scripted (the set of ingredients are fixed during image sequences).

We annotated 251 key-frames for the following actions: *cut tomato*, *cut cucumber*, *cut cheese*, *peel cucumber*, *cut lettuce*. Key-frames have been chosen to be the mid frame of the *_post* part of annotated actions; we choose these actions as they are the moment where ingredients have been transformed into a new state. This state is expected to remain fixed until the next action. An example of annotated frames is show in 3.8. This annotation process results in an average of 116 samples for testing per composite class (Table 3.2 (B)). Each ingredient is segmented with a polygon using the *LabelMe* tool [1]. Ground-truth annotations are available [2].

3.3.2. Implementation

We used the CAM implementation [102] as the baseline for localizing foods for all the different composite classes. Activation maps of CAM are directly evaluated since composite classes are separately present in each activation map. We used similar network configurations for training the baseline and training our method.

[1] `https://github.com/wkentaro/labelme`

[2] Annotation of food objects in keyframes from 50 salads dataset: `https://hal.archives-ouvertes.fr/hal-01815512/file/annotation_json.zip`.

	Top 1			Top 3		
Composite class	Baseline	Product	Average	Baseline	Product	Average
Cheese_diced	18.75	**80.00**	77.78	29.41	80.00	**87.50**
Cheese_whole	**100**	**100**	**100**	**100**	**100**	**100**
Cucumber_diced	**61.54**	33.33	42.11	**81.40**	35.71	44.44
Cucumber_peeled	0.00	**40.00**	**40.00**	0.00	**50.00**	**50.00**
Cucumber_sliced	71.43	93.55	**93.94**	77.14	94.74	**95.00**
Cucumber_whole	**67.35**	64.38	65.56	**74.58**	65.79	68.37
Lettuce_diced	58.54	**91.30**	86.21	78.03	**92.59**	88.89
Lettuce_whole	42.55	**65.22**	51.52	67.61	**70.37**	64.29
Tomato_diced	**80.34**	79.78	74.26	**86.45**	82.86	79.51
Tomato_sliced	80.00	80.65	**83.33**	88.10	86.11	**89.19**
Tomato_whole	5.56	**89.80**	87.76	5.71	**92.06**	90.32
Mean	53.28	**74.36**	72.95	62.58	77.29	**77.96**

Table 3.3: Food localization results on key-frames from 50 salads dataset. Best results are in bold.

3.3.3. Results

In this section, we report the results of localizing food ingredients using the baseline method [102] and our proposed method. We evaluated both methods using the midpoint hit criteria as proposed by [69]. The midpoint is computed as the centre of gravity of the prediction values of the composed concept map. As in [69], a positive hit is considered if the midpoint falls inside the ground-truth mask; if the fired detection does not belong to the correct ground-truth label, it is counted as False Positive.

Table 3.3 reports on the performance of each class in terms of precision. The results show a significant improvement on localization precision (74%), whereas the baseline achieves (53%) on classifying composite classes. In this experiment, both concept composition methods (product and average) achieve similar performance. We also computed pixel-wise accuracy of the resulting activation maps, both for our method (63% without background, 94% with background) and the baseline (23% without background, 32% with background), again showing a significant improvement.

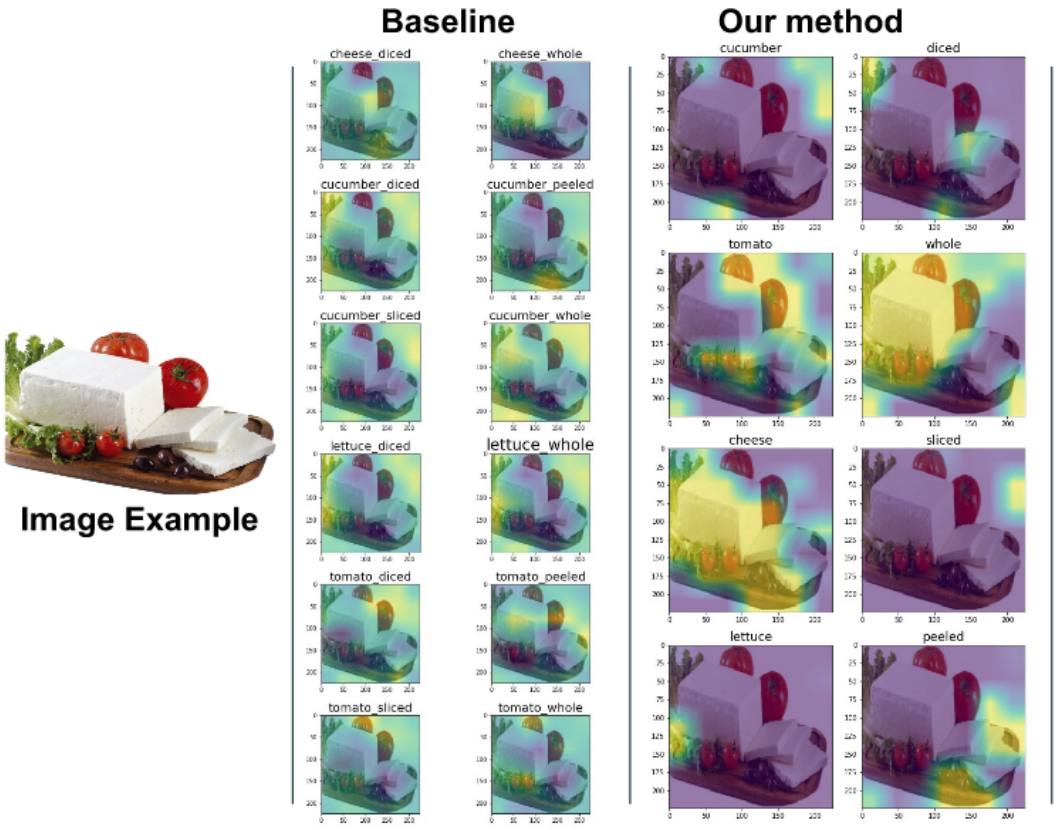
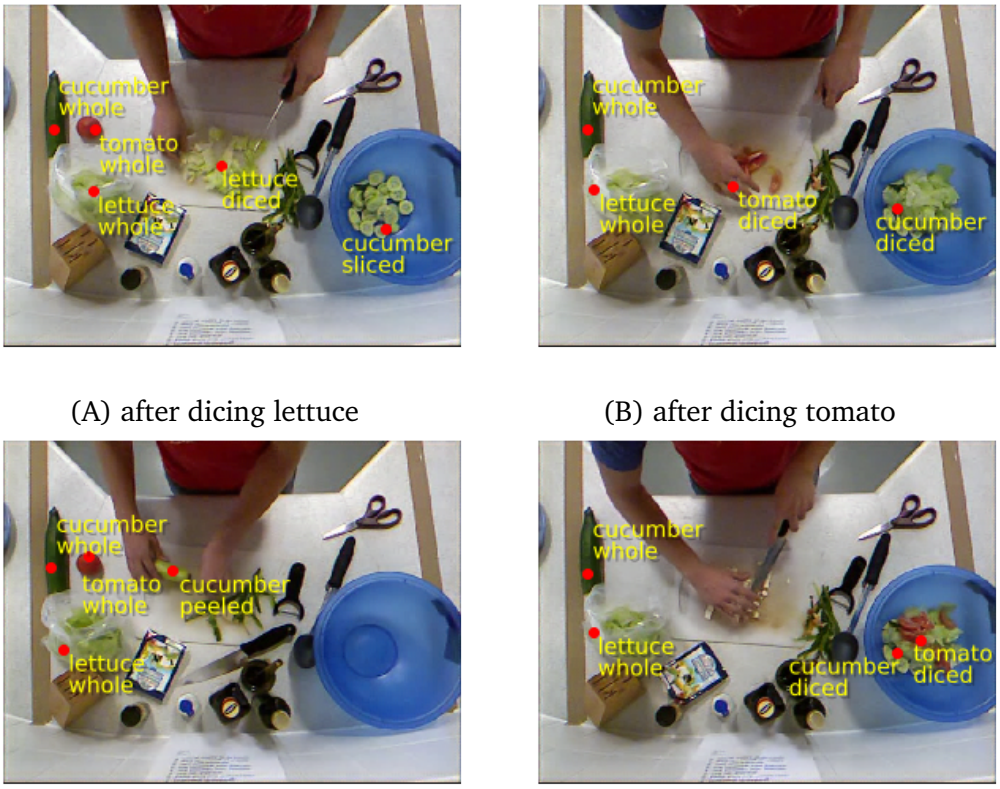


Figure 3.9: Qualitative results of food concept localization on a test image. Left: Image example. Middle: Class Activation Maps of every composed class. Right: Concept Activation Maps of each concept. Best viewed in colors.

For qualitative results, Figure 3.9 shows an example of localization of food concepts on a test image. We can see that using a multi-class learning method has difficulties in localizing different food composed classes, while our model is able to locate different food concepts in the image. Another more complex test images are shown in Figure 3.10. These images are taken from 50 salads after an action has been performed where we can see that the model is able to locate different food objects.



(A) after dicing lettuce

(B) after dicing tomato

(C) after peeling cucumber

(D) after putting tomato into bowl

Figure 3.10: Qualitative results of weakly supervised localization of food on frames from 50 salads dataset. Best viewed in colors.

3.4. Additional experiments

In this section, we report on additional experiments that use the same model to verify our previous results. First, we evaluate the model on a more extensive image set. Then, we perform a hyper-parameter optimization study on the number of deep layers in VGG used as backend in our model. Finally, we compare our model that learns jointly object classes and states using the multi-task learning technique to learning two separate classifiers for each concept.

Dataset	Food classes	Food states	# images
RPAL-Cooking States V2.0 [34]	N/A	11	7.6k
RPAL-V2.0[34] + Ours[1]	19	11	8.1k

Table 3.4: Food object (classes, states) image datasets.

3.4.1. Extending the experiment

The objective of this experiment is to perform a more comprehensive evaluation using a relatively larger dataset on our model to detect food objects used earlier in this chapter. For this purpose, we used RPAL-Cooking states V2.0 dataset [34]. This dataset contains 11 food states for 18 food classes. Unfortunately, at the time of this experiment, the dataset includes 11 annotated food states but does not include the annotation of food classes.

Annotation process We annotated the image dataset using an image tagger that we built. The annotation process was iterative; we annotated a few images then used these to train the model to predict the new images and manually correct false predictions. The code of this image tagger can be found online. By the end of this process, we combined our collected dataset and RPAL-Cooking states dataset. This resulted in 19 food classes and 11 food states (Table 3.4). We used this image collection to report on the following experiments.

Comparison with baseline Table 3.5 reports on the classification results on the combined dataset. We can conclude from the results table that the problem of classifying composite classes is a better fit when treated as a multi-task problem (i.e. jointly learn each concepts is outperforming learning to predict composite classes directly). This is the same conclusion as the one we derived from the previous experiment. However, these results can be a consequence of the larger number of trainable parameters in the task-specific layer. In the next section, we report on an experiment where we fix the number of learnable parameters for a fair comparison.

		Precision	Recall	F1-score	Accuracy
Food cam	Food	**72.44**	**73.04**	**0.7202**	**72.74**
	State	**80.11**	**80.40**	**0.7995**	**80.75**
Baseline	Food	66.64	67.02	0.6543	65.22
	State	75.52	75.80	0.7538	75.38

Table 3.5: Classification results of the combined dataset on the baseline and our model (Food CAM). Best results in bold.

3.4.2. Comparison between different problem formalization

Classifying food concepts (object types and states) can be formalized in three ways: As Multi-class classification, Per-concept classifier, Multi-task classification. In this section, we compare the effectiveness of these three problem formalizations for classifying objects and theirs states.

In multi-class classification problem, each pair of food type and food state is considered as one class. However, the number of classes is combinatory of types and states. With this formalization, the classifier will need to distinguish every composite class without benefiting from the potential common features among different concepts. The second formalization of this problem can be by building two separate classifiers, one for each concept. This formalization can be costly in terms of number of training parameters. The third formalization is using multi-task learning techniques. In this case, both concepts are learned simultaneously. First, common features are learned, then different tasks are learned in parallel. This setting can allow the network to exploit the potential common features exists in both concepts then learn concept-specific features for each task.

We conducted three experiments, one for each problem formalization to evaluate them. We used the extended combined dataset (i.e. our collected dataset and RPAL-Cooking States V2.0 dataset [34]) to train the classifiers for each of these experiments. For multi-task classification, we use the same model shown previously in Figure 3.6, where there is a shared layer for both concepts. For per-concept classifier, we use two separate models where no parameter is shared (Figure 3.11). Thus, the question we

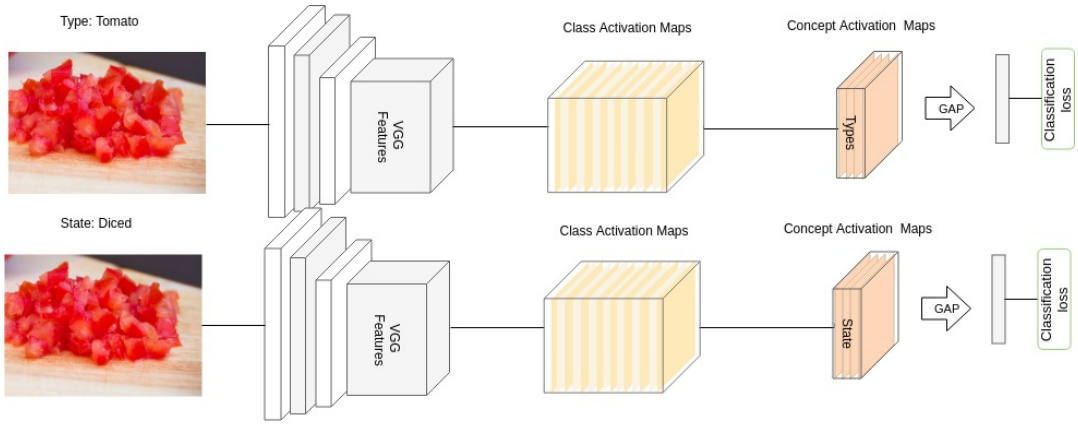


Figure 3.11: Learning each food concept in a separate classifier (no shared weights).

Classifier	Multi-class		Per-concept		Multi-task	
	Type	State	Type	State	Type	State
Accuracy	65.22	75.38	70.08	79.70	**72.74**	**80.75**
#params	-		569,673		569,673	

Table 3.6: Evaluation of different problem formalizations for classification of food concepts. Accuracy per concept and number of trainable parameters are compared.

are asking in this section is: Does the fact of having shared weights helps in learning different food concepts?. For the multi-class classification, we used the same baseline model [102] discussed in the previous section.

In Table 3.6, we report on the classification accuracy of both food concepts. The table shows that sharing information between these two food concepts can benefits of learning different attributes than learning a different detector per attribute in the case of food concepts. In addition that formalizing the problem as a multi-task classification problem is the most effective formalization for classifying interleaved concepts as in food objects.

		precision	recall	f1-score	accuracy
Food cam [VGG-19]	Food	72.44	73.04	0.7202	72.74
	State	80.11	80.40	0.7995	80.75
Food cam [VGG-18]	Food	72.39	73.11	0.7177	72.82
	State	81.18	80.89	0.8059	81.60
Food cam [VGG-17]	Food	73.57	73.25	0.7233	71.80
	State	**83.24**	82.66	**0.8283**	**83.50**
Food cam [VGG-16]	Food	**76.54**	**76.43**	**0.7588**	**75.93**
	State	83.18	**82.73**	0.8275	82.83
Food cam [VGG-15]	Food	70.94	71.97	0.7066	69.17
	State	82.00	82.38	0.8203	82.88

Table 3.7: Classification results of RPAL-Cooking states V2.0 dataset with different VGG features. Best results in bold.

3.4.3. Hyper-parameters optimization

The model, shown in Figure 3.6, uses a pre-trained backbone network which is the VGG model [77] pre-trained on ImageNet Dataset [45]. This VGG backbone has been initialized and then frozen during training. VGG features (e.g. the activations of the last convolutional layer) are used as input to the shared model. In Transfer Learning, the choice of the number of VGG layers to include in the model can affect the learnt features. This can be due to the fact that increasing the depth of the network, provides a larger receptive field encoding higher level features.

VGG network architecture has 19 convolutional layers. To find the good trade-off number of deep layers to include in the model, we conducted an experiment to study the effect of the number of VGG convolutional layers on classification performance. We report the results in Table 3.7. The results show that VGG-16 gives the best performance in our case and thus we used VGG-16 in all the previous experiments as a backbone.

3.5. Discussion

Recent progress in machine learning [64, 29, 54] has provided techniques that can be used to detect and locate objects. However, such techniques require a large number of annotated images. Unfortunately, none of the commonly available datasets for food provides images or annotations for different food and their states. It is costly to have this type of dataset as the combinatorial nature of the problem. To remedy this situation, we have created a new annotated dataset from Google Images, using food classes and food states as keywords for queries. We use this dataset to train a pre-trained model with the weakly-supervised learning technique of Zhou et al. [102]. We use the resulting activation maps to train a new layer which recognizes food states and food classes simultaneously. Then, this model has been evaluated on complex scenes from 50 salads dataset.

The problem of classifying objects and their states can be considered as a multi-class classification problem, a multi-label classification problem or even a multi-task classification problem. We reported on experiments to compare different solutions to these problems. These experiments showed that multi-task architecture is a better fit for the problem of classifying objects and their states. We end this chapter with a hyper-parameter optimization study that provide experimental justification to the choice of the number of VGG layers used in our backbone architecture.

Chapter 4

Recognizing Manipulation Actions: Related works and Oracle study

Almost by definition, manipulation actions involve interaction with objects over time, and thus require recognition techniques to detect and localize objects as discussed in the previous chapters. In addition, manipulation actions are generally performed for a purpose involving changing or preserving the state of objects. In this chapter, we review current approaches to recognizing manipulation actions and position our work with respect to the state of the art. We, then, describe an *Oracle* analysis that studies the link between the task of object recognition and manipulation action recognition task.

4.1. The recognition of manipulation actions

Actions can be described with different characteristics such as the motion, the context, and the interactions with objects. Sport actions for examples require spatial temporal description of motion to differentiate between actions such as walking and running. Other actions are highly dependent on the scene context such as swimming, cooking and driving as shown in [58, 30]. The third characteristics of actions are object interactions as in brushing teeth and opening a door. Describing an action with a certain characteristic depends on the problem domain. For example, the swimming

Front crawl Baby crawling

Figure 4.1: Action examples from UCF dataset [79]

action in Figure 4.1 can be described using scene context information (a person and a swimming pool) or object relations (a person in a swimming pool), but describing the same action as front crawling would require motion information. In the same figure, recognizing the image as baby crawling would require motion information.

Early approaches to action recognition have been strongly influenced by the nature of available action recognition datasets. The standard benchmarks for action recognition concentrate on domains where motion is an essential property of the action such as UCF101 [79] and sports-1M [39]. The object interactions are limited in these datasets. As a result, much of the current literature concerns techniques that recognize actions as motion patterns, without regard for effects on the environment. However, recognizing actions from only motion cannot discriminate whether actions such as brushing one's teeth or dicing vegetables are real or imitations.

While actions generally involve motion, manipulation actions also involve interaction with objects in the scene. Focusing on context information and motion only to model manipulation actions such as "cutting a tomato", or "kicking a ball", tend to over-fit and poorly generalizes to unseen contexts or situations [88]. Thus, approaches of manipulation action recognition need to consider objects explicitly in action representation.

To better understand the different recognition methods to manipulation action, we categorize them into two main approaches. The first approach considers relations between objects in the scene and the environment (e.g. co-existence of objects). The second approach represents actions as transformations from preconditions into post-conditions.

In the first approach, modeling object relations between sequence of frames is complementary, or sometimes alternative, to learning local spatio-temporal patterns of motion. Visual object relations can be seen as constructing a scene graph that expresses the co-existence of object sets and different object relationships in the scene. Among these relations, object-to-object relations describes the pairwise object interaction [7], and inter-relations between sets of objects [55, 104].

An alternative approach for recognizing manipulation actions relies on the changes in the environment and the context of the scene. Thus, the detection of these conditions can be enough to infer the occurrence of the action. For example, Wang et al. [88] have demonstrated a way to represent an action as a transformation between precondition and effect of the action. Considering that the action happens in the middle of the video, the first n frames represent the precondition, and the last n frames represent the post-condition. They use ConvNet features to embed precondition frames and post-condition frames. Even this work has not been evaluated on manipulation actions in particular, it shows promising results in discriminating similar actions that involve objects such as kicking a bag, and kicking a person.

In this thesis, we are interested in modeling actions as transformations of object states. To best of our knowledge, the only works that discusses this idea are [19, 5]. Fathi and Rehg [19] proposed to detect object states by first detecting changed areas from the beginning and ending frames of a trimmed action segment. In practice, changed areas in the scene are detected and represented with classical features for shape, color and texture. Then, for each action segment, a concatenated feature vector of two frames (one from the beginning, and one from the end) to train linear action classifier. The method considers changed areas in the scene to be an object in a specific state or new object.

Supervised learning of different object states is challenging due to the lack of datasets that provide object states. Alayrac et al. [5] have investigated the idea of automatic discovery of both object states and actions from videos. While this work is promising, it has been evaluated on a small number of action classes.

For the objective of this thesis, we are interested in modeling actions with explicit information about the objects. In the coming chapters, we investigate an end-to-end

method to learn objects explicitly with their states and study manipulation actions as a transformation of object states. We argue that object states are more apparent in individual frames than an action verb. However, to better study the impact of certain object information on the task of manipulation actions recognition, we performed an oracle study to recognize actions through object information. The rest of this chapter reports on this oracle study to provide a deeper understanding of what type of object information is required for the task of manipulation action recognition.

4.2. Oracle protocol

In this section, we start by defining an oracle study and define our objective from this study. Then, we set the experiment protocol and describe the used models as well as the experimental details of the training process. We, then, explain each of the experiments by describing its input data and report on the results. This section ends with a summary and conclusion of these experiments.

4.2.1. An Oracle study definition

The word *oracle* comes from the Latin verb *ōrāre*, which means "to speak". An oracle is a person or agency considered to provide wise and insightful counsel or prophetic predictions or precognition of the future, inspired by the gods[1]. With the same analogy, an oracle study is an experiment that is given controlled access to the system ground-truth information for the goal of getting insights about certain hypotheses.

Oracle studies have been used in the literature of computer vision to confirm specific hypotheses or discover new ones. For example, in [66], an oracle study is constructed to confirm the hypotheses that using the information of previous optical flows are useful for estimating the current flow. In that study, the ground-truth from previous optical flow was used to estimate the current flow. For the problem of action detection, Xu et al. [93] conducted an oracle study to demonstrate the the effectiveness of incorporating information from the future frames by giving access to future informa-

[1] wikipedia, retrieved in Nov, 2019

tion instead of predicting it. This type of studies may help in finding an upper-bound of feasible results and in suggesting new hypothesis.

4.2.2. Recognizing actions through objects

The question we are studying in this section is: *How does objects information contribute in recognition of manipulation actions?*. Our objective of this study is to better understand the correlation between the object detection and action recognition tasks. Specifically, how information from surrounding objects in the scene contributes to the task of manipulation action recognition.

One motivation to conduct this analysis comes from the fact that some verbs are associated with specific objects according to their affordances. For example, in EPIC-kitchen dataset [14], the action verb *sharpen* always occurred with the existence of the object noun *knife*. Another motivation to this study comes from the hypotheses about how do human infer about actions; Recent advances in experimental neuroscience establishes a link between object recognition and action understanding in human perception [59, 25].

We study the correlation with the following types of object information: (1) The importance of the order in which objects appearing in the scene, to investigate this we have conducted two oracle experiments. The first experiment has access to the set of objects that appeared in the whole video clip without ordering. The second experiment uses the order of which the set of objects enters the scene. (2) The temporal position when an object appears in the scene, for that we assign the list of objects in the scene for every frame in the video clip. (3) The spatial position where the object is located in the scene, for that we scored objects on their position in the scene. Objects closer to the scene centre receive higher scores than those at the borders. (4) The state of the object, we assign a state for objects in each frame of the video clip.

4.2.3. Experimental setup

We use a variety of neural network architectures to explore their ability in learning different types of input data. These architectures include dense neural networks,

convolutional neural networks, as well as recurrent neural networks. We used standard versions of these architectures. Here are the experimental details of these models.

Dataset For this study, we have used the EPIC-Kitchen dataset [14] as it is a large-scale dataset which would help in learning significant relations. The dataset contains $N_{verb} = 125$ verbs and $N_{noun} = 352$ nouns. The organizers provide ground-truth labels for the action verb and noun. The dataset is recorded by 32 participants in their native kitchen environments. Each participant narrates the recorded video with a simple sentence. The action ground-truth labels are extracted from participants narration sentences; a verb and a list of nouns in the sentence. In addition to the action ground-truth, the dataset provides object bounding boxes for some frames (precisely 1 frame per second). The oracle acts as a perfect object detector and has controlled access to these ground-truth information depending on what we are questioning. Since the test set is not available, we follow Baradel et al. [7] in splitting the training set of the dataset into training and validation depending on the participant IDs. The training set includes videos of participants with IDs from 1 to 25. The validation set includes videos of participants 26 to 31. We report results on the validation set.

MLP Multi-Layer Perceptrons are dense neural networks, where all neurons between two consecutive layers are connected. We study two different architectures of MLPs. One is built to encode dimension reduction of input data, and the other is supposed to learn bottleneck features of objects which are then mapped to verb classes. Here are the used architectures in this study. For both models, the output layer consists of $N_{verb} = 125$ neurons while input layer varies depending on the experiment and will be mentioned in every experiment:

- MLP (A): consists of 3 hidden layers with 300, 200, and 150 neurons respectively. Each layer is activated with ReLU.
- MLP (B): consists of 3 hidden layers with 100, 50, and 100 neurons respectively. Each layer is activated with ReLU.

Standard RNN Recurrent neural networks learn correspondences in temporal sequences. All RNNs have feedback loops in the recurrent layer. This lets them maintain information over time. However, it can be challenging to train standard RNNs to solve problems that require learning long-term temporal dependencies. We have tested two

configurations of standard RNNs; one is composed of one hidden layer and the other composed of three hidden layers.

LSTM Long-Short Term Memory network [16] is a specific type of RNN that uses special units in addition to standard RNN units. LSTM units include a 'memory cell' that can maintain information in memory for long periods. A set of gates is used to control when information enters the memory, when it is output, and when it is forgotten. This architecture enables an LSTM to learn longer-term dependencies between temporal sequences. We have experimented with multiple configurations of LSTMs, in particular, LSTM with one hidden layer, 2-hidden layers, and 3-hidden layers. In addition to Bidirectional LSTM (Bi-LSTM) with one and three hidden layers.

Conv1D 1D convolutional neural networks can be used on sequential data by applying the convolution operation over the temporal dimension. We used variations of Conv1D layers followed by ReLU activation function. A general schema of those models is shown in Figure 4.2.

- Conv1D (L=5): a sequence of 5 Conv1D layers each is activated by ReLU and followed by a max-pooling layer.
- Conv1D (L=7): a sequence of 7 Conv1D layers each is activated by ReLU and followed by a max-pooling layer. Skip links are added to this architecture to merge temporal features at different temporal scales. One skip connection is added from layer L2 to L4, and another from L4 to L6.

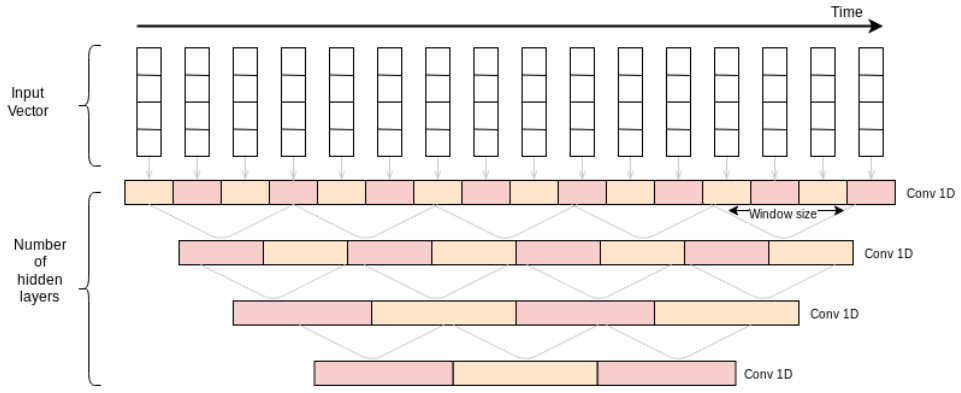


Figure 4.2: CNN 1D general architecture used in the experiments

Training setting For training, all experiments uses is the cross entropy function as a loss function to learn verb classes. Unless specified differently, the experimental settings of the oracle tests are as follows. The learning rate is selected to be the highest at which loss is still improving.

For MLP models, learning rate is $lr = 10^{-3}$. For the rest of the models, the learning rate is set at $lr = 10^{-4}$. The learning rate is divided by 2 when the loss function remains stable after 3 consecutive epochs. To control the learning rate during training, we use "ReduceOnPlateau" algorithm. We use the Adam optimizer algorithm [41] to update the network parameters. We run each experiment for 100 epochs and we report on results from the model that achieves the best loss (with minimum loss value) during training.

Evaluation Following Baradel et al. [7], we report the verb accuracy on the validation set. The reported results are averaged over three runs. We used Pytorch library to implement these models. All of the code of following the experiments is available online[2].

4.3. Oracle Experiments

At the time of performing these experiments, the action recognition challenge of EPIC dataset was in its early phase. The only available paper using EPIC kitchens for evaluation at that time was Baradel et al. [7] published in ECCV18. Baradel et al. report an accuracy of 40.89% for the verb recognition task. Assuming a uniform distribution, the probability of randomly choosing the correct verb (random chance) is 0.8% , while the probability of choosing the correct verb by choosing the most frequent class (largest class chance) is 21.55%.

In the following, we describe each oracle experiments with its objective, input data and results. For that, we will use N_{verb}, and N_{noun} to refer to the number of verb classes and noun classes respectively, and V_{seg} to refer to a trimmed video segment.

[2] https://github.com/Nachwa/oracle_analysis_epic

4.3.1. Objects set for verb recognition

In this first experiment, we ask the following question: *How well can we recognize verbs using only the set of objects that appeared in the video segment?*. To answer this question, we take as input the set of all objects that appear in an action segment and learn to output the verb that best describes this frame sequence. There is no ordering notion applied to this set. The input vector is not ordered, which makes a dense neural network architecture suitable for this task (i.e. MLP). We used the two architectures of MLPs explained earlier.

This set of objects is used as input to a multi-layer neural network to predict action verbs. We extract the set of objects in a trimmed action segment from provided ground-truth object annotations. Practically, the input is the vector obj_set of size $(N_{noun} \times 1)$ and is hot-encoded as follows:

$$obj\_set(obj_{ID}, v_{seg}) = \begin{cases} 1, & \text{if } obj_{ID} \text{ exists in } v_{seg} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where $v_{seg} \in V_{seg}$ video segments and obj_{ID} is the object ID. We report the results of this experiment on the validation set. Table 4.1 shows that around 28% of action verbs can be detected with only information about the used objects in the video segment.

	MLP		Random	Largest class	Method of
	A	B	chance	chance	Baradel et al. [7]
Verb Acc	28.46	28.07	0.08	21.55	40.89

Table 4.1: Oracle study: set of objects in a video clip using two MLPs architectures.

4.3.2. Objects ordered chronologically

In this experiment, we study the effect of sorting the object sets according to their appearance in the video segment. This ordering does not include information on when exactly the object appeared but only the order of appearance.

To ensure a fixed vector size, the size of input vector is $(N_{objseq} \times 1)$ where N_{objseq} is the length of the longest list of objects in all videos of the dataset which is equal to

$= 512$. If the object list is shorter than N_{objseq} the vector is padded with -1. We use the provided object annotations for EPIC-kitchen frames. The ground-truth labels of objects are given for 1 frame per second. Because we are only concerned with the order in which objects appear, we skip frames that have no objects. For example, given the following set of objects:

- at $t_i = 1$: $[obj_1, obj_3]$,
- at $t_i = 2$: no object appeared.
- at $t_i = 3$: $[obj_2]$.
- at $t_i = 4$: $[obj_2]$.

The corresponding input vector is as follows: $[obj_1, obj_3, obj_2, obj_2, -1, ..., -1]$. In this experiment, we compare several network architectures that are often used with chronological or sequential data such as a standard RNN, a LSTM, Bidirectional LSTM, and 1D-Convolutional Neural Network (CNN). We also report on MLP for completeness.

The results of training with these different architectures is shown in Table 4.2. As was expected, MLP architectures do not perform well in this task. Recurrent Neural networks perform better than MLPs at finding a correlation between the object appearance and the action verb being performed. We can see, also, that Conv1D model performs the best on the validation set, outperforming RNN methods.

	MLP		RNN	LSTM			BiLSTM		Conv1D
Exp.	A	B	L=1	L=1	L=2	L=3	L=1	L=3	L=5
obj. set	28.46	28.07	-	-	-	-	-	-	-
obj. chron	24.09	25.24	26.13	24.62	26.45	27.68	26.62	29.05	29.90

Table 4.2: Oracle study: The effect of ordering vs. no ordering the object list. *obj.set* refers to the experiment with a set of objects with no order as input. *obj.chron* refers experiment when input is the objects ordered chronologically.

4.3.3. Objects ordered temporally

This experiment studies the effect of adding information about when exactly the object appeared in the video segment. The order in the input vectors respects the temporal appearance of objects. Unlike the previous experiment, we assign to each frame all objects appeared in the scene. Even though a frame contains no object, it preserve its position in the input matrix.

In practice, the input is a matrix of ordered vectors of objects. The size of input matrix is $(N_{frame} \times N_{noun})$ where N_{noun} is the number of noun classes. This dimension correspond to the hot-encoded vector of objects in the scene $(1 \times N_{noun})$. For the N_{frame} dimension, we sample 1 frame per second for each video segment to ensure a fixed matrix size and use the maximum length of all videos $N_{frame} = 414$ frames which correspond to 06 minutes and 54 seconds. The list of hot-encoded vectors are, then, stacked together in temporal order. The matrix is the stacked list of hot-encoded vectors of each frame as follows:

$$obj\_mat(frame_i, obj_{ID}) = \begin{cases} 1, & \text{if } obj_{ID} \text{ exists in } frame_i \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where $frame_i$ is the frame index and obj_{ID} is the object ID.

The table 4.3 shows the results of this experiment. We can notice a significant improvement in verb recognition by adding information about when a certain object has appeared in the video segment. This conclusion is equally valid for all experimented models in the table 4.3.

Exp.	RNN		LSTM		BiLSTM		Conv-1D	
	L=1	L=3	L=1	L=3	L=1	L=3	L=5	L=7
Obj. chron	26.13	-	24.62	27.68	26.62	29.05	29.90	-
Obj. tempo	**34.65**	**34.37**	**35.81**	**34.01**	**32.29**	**34.77**	**32.44**	**35.00**

Table 4.3: Oracle study: The effect of ordering object lists chronologically (*Obj. chron*) and temporally (*Obj. tempo*).

4.3.4. Explicit information about time and video duration

From the previous experiment, we have found that temporal-ordering the objects in the scene is an important feature for recognizing action verbs. From these promising results, in this experiment, we perform a deeper investigation of the effect of adding more explicit temporal information. We practically add the following temporal information: (A) The moment in time when the object appears in the video segment, and (B) The duration of the video segment. We encode these features by adding the frame numbers explicitly in an ascending and descending order. The ascending order encodes the frame number, while the descending order encodes the video duration by counting the time left before the end.

In practice, we define 4 vectors (k_1, k_2, k_3, k_4) to encode information about time and duration. Each vector is of size $(N_{frame} \times 1)$. For a video segment v_i with a length of $N_{frame}(v_i)$ and $frame_i \in [0, N_{frame}(v_i)[$, we define:

- The frame number k_1 defined as:
  $k_1(frame_i) = frame_i$.
- The descending frame number k_2.
  $k_2(frame_i) = N_{frame}(v_i) - frame_i$.
- The ratio of total video length from the beginning until a certain frame.
  $k_3(frame_i) = frame_i / N_{frame}(v_i)$.
- The ratio of the video segment at a certain frame before it reaches its end.
  $k_4(frame_i) = (N_{frame}(v_i) - frame_i) / N_{frame}(v_i)$.

Each of these vectors (k_1, k_2, k_3, k_4) are padded with zeros to ensure fixed size. Then, we concatenate them to the head of the input matrix obj_mat in equation 4.2. The final input matrix size is $(N_{frame} \times (4 + N_{noun}))$. We use the same models as in the previous experiment. We refer to adding these time features as (vec_k) in Table 4.4. The results show that adding information about time and video duration improves the results in most architectures.

Experiment	RNN		LSTM		BiLSTM		Conv-1D	
	L=1	L=3	L=1	L=3	L=1	L=3	L=5	L=7
Obj. tempo	34.65	34.37	35.81	**34.01**	32.29	**34.77**	**32.44**	35.00
$+(vec_k)$	**36.74**	**36.51**	**36.19**	32.83	**35.98**	32.19	32.02	**36.98**

Table 4.4: Oracle study: comparing objects ordered temporally (*Obj. tempo*) with and without adding explicit information about time and duration ($+vec_k$).

4.3.5. The effect of the spatial position of objects in the scene

In the previous experiments, we studied the correlation of the recognition of an action verb to the existence of objects in the scene as well as to their temporal appearance. In this experiment, we study the effect of adding spatial information about the location of each object in the scene. Our intuition is that some objects may appear in all frames as part of a background while having no relation to the actual action being performed. Thus, not all objects participate equally in the action.

Our hypothesis is that manipulated objects are the objects that are the most relevant to the action. We study two ways of measure object relevance: The objects under manipulation are the objects (A) within the hand region, or (B) at the centre of our attention. The scoring, here, is to filter out objects that are not under manipulation. We score objects depending on the distance of their location to (A) or (B). Ideally to study both hypothesis, we need eyegaze and hands position ground-truth. As these data are not available, we use the image centre as the centre of attention. This is justified because EPIC-Kitchens is an egocentric dataset. To study (A), we use hand position computed using trained Mask-RCNN and provided by Baradel et al. [7].

For the scoring function, we choose the Gaussian function $G(\mu, \sigma)$, shown in figure 4.3. These scores are high when the object appears close to the centre. Low scores are assigned when the object is far from the centre. We set the standard deviation σ to be half the size of the input image. However, the mean μ of the function G is the image centre with image-centred scoring while in hand-centred scores the mean is the centre coordinate of detected hands. These scores compute how relevant an object is given its the center (obj_x, obj_y) of its location in the image.

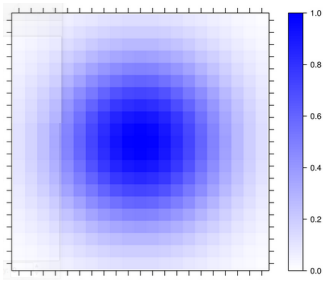


Figure 4.3: 2D Gaussian function used to score objects spatial location in the scene. Objects closer to center gets a score closer to 1.

The input matrix has the same shape of input matrix in the experiment 4.3.4 which is $(N_{frame} \times (4 + N_{noun}))$ which includes the 4 temporal feature vectors vec_k. However, the object hot-encoding in obj_mat (eq 4.2) is replaced with objects computed scores as follows:

$$obj\_scr(frame_i, obj_{ID}) = \begin{cases} G\_score(obj_x, obj_y), & \text{if } obj_{ID} \text{ in } frame_i \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $G\_score(obj_x, obj_y)$ is the Gaussian function either image-centred or hand-centred, $frame_i$ is the frame index and obj_{ID} is the object ID.

We study the effect of scoring the relevance of an object to the performed action according to its spatial location in the scene. We compare this hypothesis to no-scoring technique of object relevance in which all objects in the scene are treated as equally relevant to the action (as in Experiment 4.3.4). Results of this study, reported in

Socring	RNN		LSTM		BiLSTM		Conv-1D	
Method	L=1	L=3	L=1	L=3	L=1	L=3	L=5	L=7
w/o scoring	**36.74**	**36.51**	**36.19**	**32.83**	**35.98**	**32.19**	32.02	**36.98**
Image centre	34.92	35.13	33.35	31.64	34.63	31.53	**33.23**	36.77
Hands centre	34.25	33.88	34.12	29.58	33.86	30.05	33.12	32.33

Table 4.5: Oracle study: the effect of scoring objects on their spatial position. The table compares a no-scoring method (*w/o scoring*) to scoring objects relative to (*image centre*) and to (*hands centre*).

Table 4.5, shows a decrease in the recognition results when objects are scored by their location in the scene compared to not scoring them. These results are surprising and do not confirm our hypothesis. Note that we revisit this conclusion in an additional experiment in chapter 5 in section 5.3.

4.3.6. The effect of using object states for verb recognition

From all previously presented experiments, we observe that including meta-data about objects (object order, time of object appearance, and spatial scoring of objects) is not sufficient to equal the performance of the state-of-the-art method proposed by Baradel et al. [7]. In the following, we study the effect of adding information about the state of objects to the matrix of ordered objects. Our hypothesis is that the success of an action can be measured by achieving the desired goal.

In the case of manipulation actions, the desired goal can be a certain object state. To study this idea, we assume a perfect classifier of object states. Since ground-truth labels of object states are not available in EPIC-Kichens dataset, we assign a state to each frame in the video clip. For each verb, we manually define verb rules $R(v) : Pre(v) \rightarrow Post(v)$ by assuming that a verb changes the state of the frame from a pre-state to a post-state. $Pre(v)$ returns the state ID of the pre-state of the verb v using the rule $R(v)$. For example, the verb *open* changes the state in the frame from *opened* to *closed* as follows: $R(open) : closed \rightarrow opened$, and similarly, $R(cut) : whole \rightarrow diced$.

This process of manually defining verb rules resulted in 31 different states and 49 verb rules. It also left some verbs with out defined verb rules. We assign the pre-state to all frames before the middle frame of the video segment and the post-state to all frames after the middle frame. However, when no verb rule is defined we assign 1 which indicates the *no-state* state. The label encoding of this experiment can be written as

follows:

$$obj\_st(frame_i, obj_{ID}) = \begin{cases} 1, & \text{if } (R(verb_{ID}) \text{ is not defined}) \\ & \quad \text{and } (obj_{ID} \text{ in } frame_i) \\ Pre(verb_{ID}), & \text{if } (R(verb_{ID}) \text{ is defined}) \\ & \quad \text{and } (obj_{ID} \text{ in } frame_i) \\ & \quad \text{and } (frame_i < N_{frame}(v_i)/2) \\ Post(verb_{ID}), & \text{if } (R(verb_{ID}) \text{ is defined}) \\ & \quad \text{and } (obj_{ID} \text{ in } frame_i) \\ & \quad \text{and } (frame_i \geq N_{frame}(v_i)/2) \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

In practice, the input matrix has the same shape of input matrix in the experiment 4.3.4 which is $(N_{frame} \times (4 + N_{noun}))$. It includes the 4 temporal feature vectors vec_k and the object matrix ordered temporally. However, the object hot-encoding in obj_mat (eq 4.2) is replaced with obj_st defined in eq 4.4. In table 4.6, we report results about this experiment compared to results of experiment 4.3.4. The table shows that adding object states to the input matrix has a significant influence to the performance in most architectures. We can also notice that object states along with other information perform the best on Conv-1D architecture. However, we should note that the states are encoded from actions directly which makes the problem of action recognition significantly easier. In the coming chapter (chapter 5), we re-used this state-defined rules in another experiment; more details can be found in subsection 5.1.1.

	RNN		LSTM		BiLSTM		Conv-1D	
Method	L=1	L=3	L=1	L=3	L=1	L=3	L=5	L=7
w/o obj. state	36.74	36.51	36.19	32.83	35.98	32.19	32.02	36.98
with obj. state	**54.35**	**63.04**	**56.37**	**56.22**	**57.38**	**58.36**	**68.59**	**72.90**

Table 4.6: Oracle study: study the effect of adding meta-data about object states. The table compares the the object matrix with (*with obj. state*) and without object states (*w/o obj. state*).

4.4. Discussion

Manipulation actions can be described by the way they interact with objects. However, manipulation actions are also actions, and as such, inherit the properties of an action such as patterns of motion and relevance to context. In literature, different approaches focused on a specific combination of these properties. A review of the literature in section 4.1 revealed that previous investigations have not explicitly considered object state in the representation of actions. This encouraged us to investigate such an approach.

We began an investigation of the correlation between the presence of objects and manipulation actions in video sequences. We investigated the usefulness of including meta-data about objects, including the order of appearance of objects, the time of object appearance, the spatial locations of objects, and object attributes. We excluded all other type of information from videos such as appearance and motion. For that, we started by assuming a perfect object detector in a video sequence and learn to derive the action verb of the video. We compared different neural network architectures for this analysis, including CNNs, RNNs, and MLPs.

A summary of these experiments is shown in Table 4.7. These results can be reproduced using the following link [3]. These experiments demonstrate that the objects present in a manipulation action video can play an important role in recognition of the action. Providing information about object classes present in the scene and with no motion information can achieve comparable results to the state-of-the-art. Adding attributes about the objects, such as object states, seems to make the task of manipulation action recognition substantially more reliable. With this conclusion, we decide to continue studying the explicit inclusion of these meta-data to our model for the recognition of manipulation actions. In the next chapter, we explain how we use object and state information for the recognition of manipulation actions.

[3] https://github.com/Nachwa/oracle_analysis_epic

Object set	Chronological order	Temporal order	Time features vec_k	Spatial scoring	Object states	MLP	RNN	LSTM	BiLSTM	Conv-1D
✓	-	-	-	-	-	28.46	-	-	-	-
-	✓	-	-	-	-	25.24	27.04	27.68	29.05	**29.90**
-	-	✓	-	-	-	-	34.65	**35.81**	34.77	35.00
-	-	✓	✓	-	-	-	36.74	36.19	35.98	**36.98**
-	-	✓	✓	✓	-	-	35.13	34.12	34.63	**36.77**
-	-	✓	✓	-	✓	-	54.35	56.37	58.36	**72.90**
Method of Baradel et al. [7]						40.89				
Random chance						0.08				
Largest class						21.55				

Table 4.7: Oracle analysis on the relation between object-related information for manipulation action recognition using different network architecture. Best results are listed for each model.

Chapter 5

Recognizing Manipulation Actions from State-Transformations

A manipulation action transforms an object from a pre-existing state (pre-state) into a new state (post-state). Thus we can say that the action causes a change in the state of the corresponding object. In this chapter, we investigate the feasibility of recognizing object classes and object states from a small number of frames and use changes in object states to recognize actions.

In section 5.1, we start by a brief review that positions our approach in the broad spectrum of related works. In section 5.2, we explain state-changing actions and how do we label video segments with states and explain our model architecture for action recognition. We then move to report on our participation to EPIC-kitchen challenge on action recognition. After that, in section 5.3, we explain how we used foveated vision concepts on input images for a faster experimentation. Then, in section 5.4, we show how to apply the method of state-transformation on other frame-based methods for action recognition; We study the generalization of our model on top one baseline methods for action recognition. At the end of this chapter, in section 5.5, we show how we can use reversible actions for data augmentation during training.

5.1. Modeling manipulation action as state transformation

Most current approaches to action recognition interpret a frame sequence as a spatio-temporal signal. As we mentioned earlier, 3D Convolutional Neural Network is a direct adaptation of 2D CNN to the spatio-temporal case. However, it results in a substantial increase in the number of parameters that must be learnt, significantly increasing the computational cost and the requirements for training data. An alternative approach is to decompose recognition into a static recognition phase using a 2D kernel followed by with either a 1D temporal kernel [91] or a Recurrent Neural network [16] to learn temporal information. Researchers have also explored the use of two-stream networks in which one stream is dedicated to analyzing image appearance from RGB frames and the other analyzes motion from optical flow maps [87, 76, 38]. Such approaches provide spatio-temporal analysis while avoiding the considerable increase in training parameters.

However, these approaches do not take objects explicitly into account. In the literature, two alternatives to learning spatio-temporal patterns from the videos are investigated. Both learn higher-level patterns from the signal; One approach studies object correlations or interactions throughout the video [7, 55, 104] and the other focuses on the transformations in the scene that can be in the form of preconditions and post-conditions [19, 88, 5].

Our method is a mix of both these approaches, instead of learning object correlations or scene transformations, we propose to learn object transformations where objects are associated with different states, and changes in objects are shreds of evidence of the occurrence of a particular action. This is similar in concept to the first approach as it operates on semantic level of objects information while enriching them with explicit state properties. It is also similar to the second approach as it models actions as transformations while using transformations of objects and not the scene. We also argue that changes in objects are more apparent than action verbs from static frames. Thus, we propose to predict object states and object classes from a few frames and use these to learn changes in objects properties for the modeling of different action classes in an end-to-end manner.

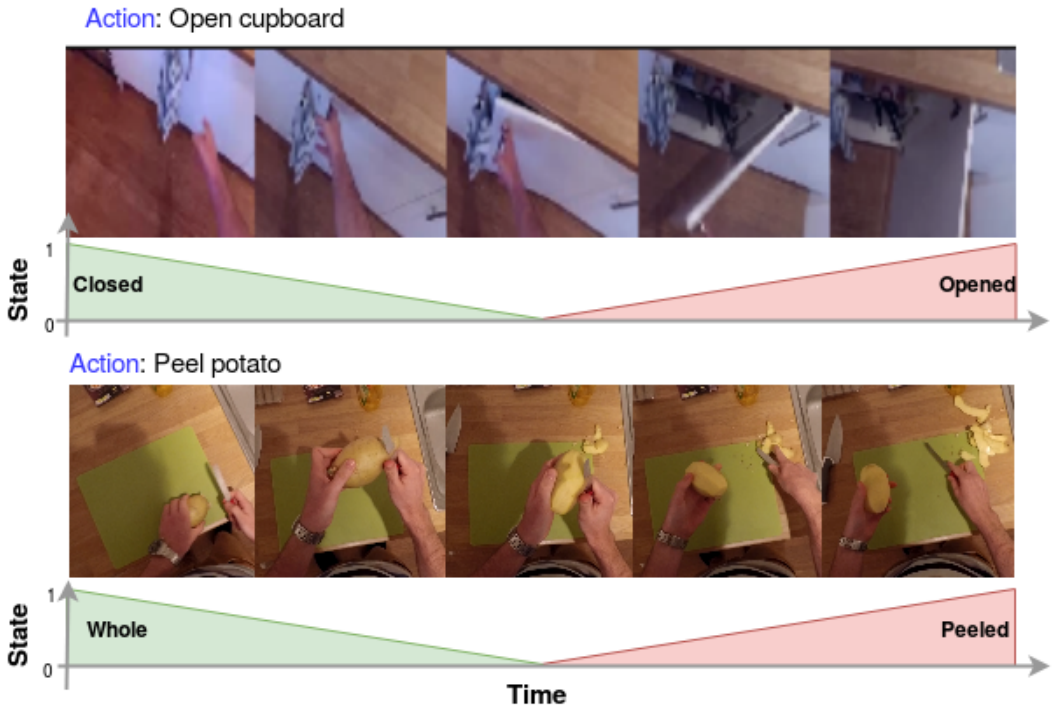


Figure 5.1: Changes in object states over time for action recognition. Two sample sequences from the EPIC kitchen dataset.

We use few frames to detect actions. This idea is inspired by the human ability to develop an understanding of a situation using a limited number of static observations. People associate observations with background knowledge in the form of previously seen episodes or past experience [20, 8, 59]. This ability makes it possible to interpret a complex scene from static images and make hypotheses about unseen actions that may have occurred and could explain changes to the scene. For example, we can understand which action is shown in Figure 5.1 with 5 frames or less from the video clip. Inferring the associated actions in frame sequences is a relatively effortless task for a human, while it remains challenging for machines [80]. We believe that such analysis may provide an effective method for inferring actions from a set of frames which are chronologically ordered and contains semantic relations between objects. Such inference would complement hypotheses from spatio-temporal action recognition.

5.1.1. State-changing actions

An action, as defined in the Cambridge dictionary[1], is the effect something has on another thing. Therefore, a manipulation action $a_i \in A$ is composed of: the subject that performs the action, the verb $v_i \in V$ which describes the effect of the action, and the object $n_i \in N$ on which the effect is applied to.

The action recognition problem can be formulated with one class for each possible combination of these attributes. For example, *cut tomato* and *cut cucumber* can be considered as two different classes as in [81]. Some recent datasets have considered the decomposition of an action into a verb and one or more objects $a = (v, (n_1, .., n_n))$ such as EPIC-kitchen dataset [14], GTEA+ dataset [52], and recently, VR-Kitchen dataset [22]. This makes it possible to study the task of action recognition as a composition of several sub-tasks (e.g. object detection and action verb recognition).

Our goal is to recognize manipulation actions that change the state of objects $s_i \in S$. The state change can appear in the object's shape, its appearance, or its location. Examples of object states include: closed, opened, full, empty, whole, and cut. However, to best of our knowledge, state labels are not available in any of the current video action-recognition benchmarks. In order to generate these labels, we define a state transition rule as well as a state transition function as follows.

State-transition rules For each action, we manually define a state transition rule that expresses the possible change in the state of the corresponding object. Each rule is defined from the action's verb v and a set of objects (nouns) n as follows:

$$A(v, n) : S_{before} \rightarrow S_{after} \text{ where } S_i \in \text{ States} \tag{5.1}$$

For example, the action *open fridge* changes the fridge state from opened to closed and thus we define its transition rule as *Open fridge*: *Fridge opened* $\rightarrow$ *Fridge closed*.

In the EPIC Kitchen dataset, we group each action into one of three different groups, depending on the type of effect that is caused: (1) changes to the object's shape, (2) changes in color or appearance, and (3) changes in location of the object.

[1]Cambridge University Press. (2019). Cambridge online dictionary, Cambridge Dictionary online. Retrieved at April 3, 2019

Examples of these categorized action are shown in Table 5.1. This categorization leaves out some action verbs, such as check, that do not change the state of an object class. For these actions, we define a *no-state* state where these actions are not supposed to change a state.

Type of effect caused by a state-changing actions:		
Shape	Color	Location
cut	dry	pour
squeeze	empty	put
open	fill	move
close	insert	scoop
remove	mix	throw
turn-on	peel	take
turn-off	wash	adjust
turn	shake	
press		
flip		

Table 5.1: Types of effects caused by state-changing verbs.

In some cases, this state transition can be defined directly from the type of action verb v_i. However, we have noticed that in some cases, a single verb is not enough to distinguish an action effect. For example, the verb *remove* can mean *open* in the action "*remove lid*" and can mean *peel* in the action "*remove the skin of the garlic*". Therefore, the state transition must take into account both action verbs and nouns. For that, we take into account the action noun in addition to the verb when defining action rules (as in rule 5.1). Examples of these state-transitions are shown in figure 5.4.

State-transition function Second, we define a state transition function F that calculates the ground-truth label of a state given the frame number. This function returns a continuous value of objects' states for each frame depending on the frame position in the video segment. We use state transition rules to find the mapping between actions and states, and the exact value of the state is calculated with the state transition function.

For the choice of the transition function, we assume that states change gradually over time. In the early frames of an action, object states are more likely to be identified with their pre-state. This likelihood gradually changes as the sequence moves to the end of the action. Thus, we use a soft assignment of states per frame. Hence, the transition function returns a real value of each state depending on the frame position in the video segment. As in Figure 5.1 the object starts in its initial state that gradually fades out and the post-state starts to appear as we advance in the video. Here, we suppose that the state changing frame is the middle frame of the trimmed action clip.

The transition of states associated with actions, can be modelled using an inverted 1D Gaussian as follows:

$$(x) = 1 - \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{5.2}$$

where x is the frame number, μ is the centre of the Gaussian peak where the transition starts, and here is the middle frame, and σ is the variance and modeled on the figure 5.2 by the width of Gaussian peak.

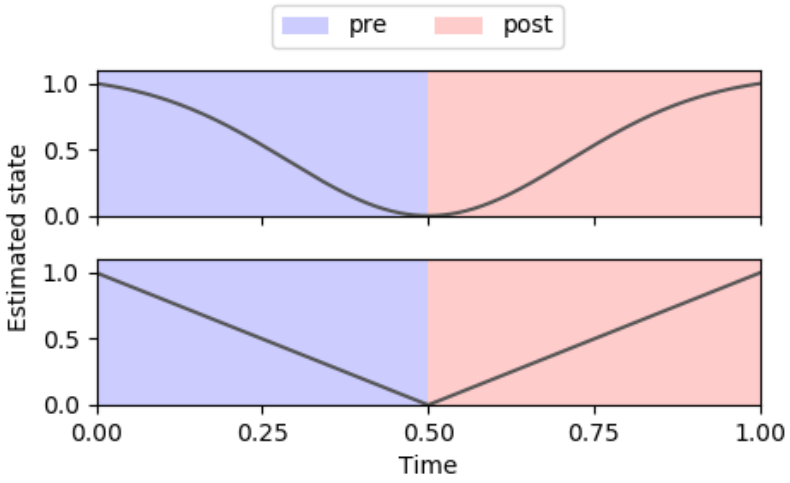


Figure 5.2: Example of state transition functions. (Top) inverted Gaussian function. (Bottom) inverted triangular function.

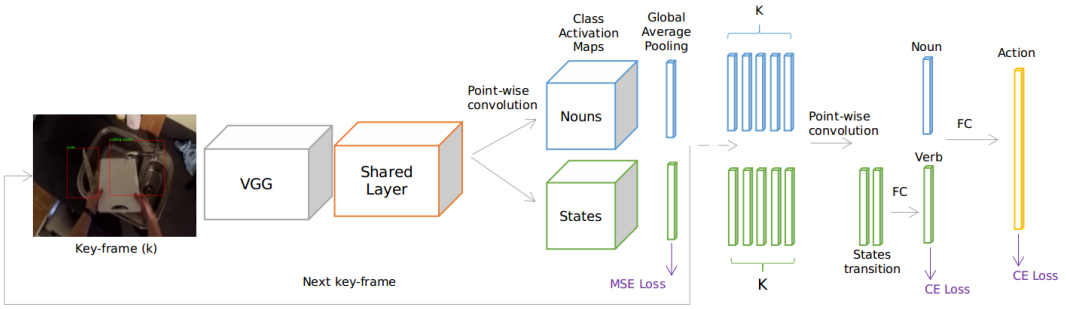


Figure 5.3: Proposed architecture of learning action recognition as state transformations.

Another possible transition function is the triangular function, shown in Figure 5.1, which is a piece-wise linear function, and we define state transitions as follows:

$$\mathrm{F}_j(x) = \begin{cases} 1 - (x - x_0)/(x_j - x_0) & x_0 \leq x < x_j \\ 1 - (x_1 - x)/(x_1 - x_j) & x_j \leq x < x_1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where x_j is the number of the middle frame in a video segment, x_1 length of video segment, and $x_0 = 0$ as we start from the begining of the video segment. These transition functions return a value between $[0, 1]$ for each state depending on both transition state rule and frame position. The state value is used to train the model to estimate the objects' state from the training frames.

5.1.2. Model architecture

The model architecture start by first identifying objects and their states of a number of sampled frames from a video segment. Then, we combine these identified objects and states through time for the learning of action verbs. Given a video segment, we first split it into k sub-segments of equal length and sample a random frame from each sub-segment.

For each sampled frame, the first part is responsible of learning two conceptual classes (object classes and object states) separately in a Multi-task manner. This part of the

architecture is the same as used in Food-CAM and explained in chapter 3 [1]. We start by extracting deep features using a VGG16 network with batch normalization [77] pre-trained on ImageNet dataset [15]. In building our network architecture, we attempted to minimize the number of parameters. Thus, VGG layers are frozen during the whole training process. VGG features provide the input to a 3×3 convolutional layer shared with both tasks (object classes and states). After that, the learning of object attributes is separated into two branches: one for object classes and the other for object states. Each attribute is learnt with an independent loss. For each frame, one noun vector and one state vector are extracted using Global Average Pooling over corresponding Class Activation Maps.

The second part of the architecture is responsible for combining the object and state features over time. Thus, we concatenate all k vectors from all k sampled frames. On this concatenated matrix, we perform a point-wise convolution on the temporal dimension to extract one noun vector and 2 state vectors (one is supposed to represent the pre-state and the other represent the post-state). The verb layer is a fully-connected (FC) layer which takes the two state vectors as well as the noun vector and its output is the verb classes. Both action attributes (verb, nouns) are fused using a FC layer for action classification.

Object nouns in EPIC dataset are chosen to be the first noun that occurs in the narrated sentence by the subject. This noun is not always the only one that appears in the frame. For that, we chose to predict all of the nouns in the scene in the first part of the architecture using multi-label learning and we use MSE for the loss. For states estimation, object state changes gradually in each frame. Thus, we use MSE as well to calculate the error.

Thus, for every sampled frame, four vectors are predicted during training. For every sampled frame, we predict one vector for the states and one for all the nouns in the scene. For every clip, two more vectors are predicted for the final action, one for the noun and one for the action verb. For learning, we use a joint loss function that calculates the error for each of these vectors. One calculates the error in state estimation using MSE, one for all nouns in the scene using MSE, one for the final action = (noun, verb) each is learned using Cross Entropy loss function.

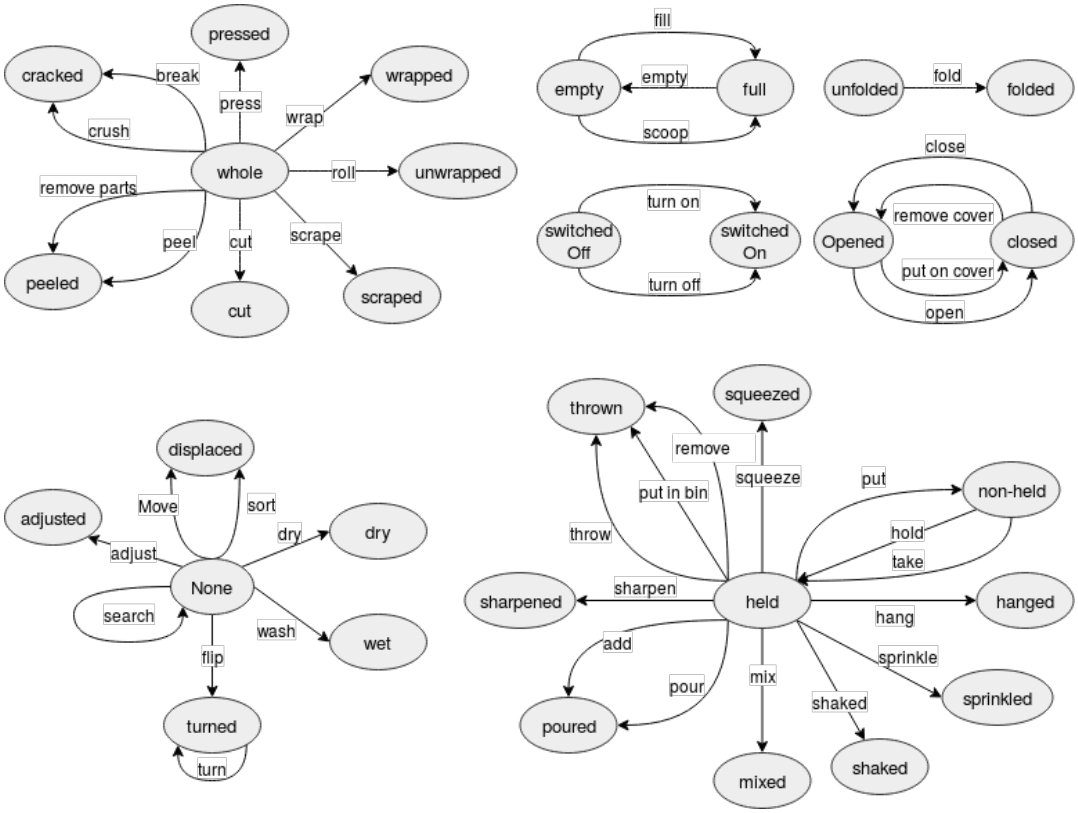


Figure 5.4: Snippet of manually defined state transition rules on EPIC-Kitchen actions. The circles represent states while arrows are actions that transforms a state to a new one.

Training. We use EPIC Kitchen video segments for training our model. For each segment, we extract a clip which is a collection of k randomly sampled frames from k equal length sub-segments. This clip represents the corresponding action video segment. This strategy has been used in multiple works with similar problems [87, 7]. We split EPIC videos in 80% for training and 20% for validation. We chose the validation set to have only samples from many-shot actions, and all samples of few-shot actions are in the training split.

In training, we used the Adam optimizer and an initial learning rate of $1e-3$ that decreases following Reduce on Plateau scheduling method. The implementation code was written using Pytorch and is available online[2].

EPIC-kitchens dataset. The EPIC Kitchen dataset is a large dataset of egocentric videos of people cooking and cleaning. In this dataset, an action label is composed of a tuple of $a_i =$ (verb v_i, noun n_i) extracted from a narrated text given for each video action segment. The EPIC verb represents the action verb, while the EPIC noun is the action object. To generate state-labels for training our model, we defined state transition rules for each state-changing action. We have noticed that some action sequences are a continuation of previous sequences in the video and thus, they do not change the state from a pre-state to a post-state. We find these sequence from the narrated sentence; if a continuation word exists in the narrated sentence we consider this sequence a continuation of the previous sequence and thus we only label the frames of this sequence with the post-state. Considered continuation words are continue, still, and continuing. As a result, we defined 49 rules and 31 different states. A diagram of these manually defined state transition rules is shown in Figure 5.4.

5.2. Results on EPIC-kitchens Dataset

EPIC challenge evaluation. For evaluation, we aggregate the results of 10 clips as in [7] by averaging the predictions of the 10 randomly sampled clips from the same video segment. To compare the results with the baselines of the challenge, we report our results using the same evaluation metrics provided by the EPIC challenge [14]. In the EPIC-kitchen challenge, they propose two sets of metrics: aggregated (micro)

[2]Code is available at `https://github.com/Nachwa/object_states`

metric and per-class (macro) metric. The computed evaluation metrics are: the micro-accuracy of the top-1 and top-5 results, as well as the macro-precision and the macro-recall. To explain the difference on the accuracy metric for example, a micro-average accuracy aggregates the contributions of all classes to compute the average metric. Thus, giving weight to each class proportionately to their frequency in the test set under evaluation. A macro-average accuracy computes the accuracy for each class independently and then take the average of all classes, hence, giving equal weight to all classes regardless of their prevalence. Here are the equations of both types of metrics on accuracy:

$$\textit{Macro-average accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_{C_i}+TN_{C_i}}{TP_{C_i}+TN_{C_i}+FP_{C_i}+FN_{C_i}} \tag{5.4}$$

$$\textit{Micro-average accuracy} = \frac{\sum_{i=1}^{n} TP_{C_i}+TN_{C_i}}{\sum_{i=1}^{n} TP_{C_i}+TN_{C_i}+FP_{C_i}+FN_{C_i}} \tag{5.5}$$

where n is the number of classes and C_i is the class i.

As the number of samples in the dataset is imbalanced over classes, following EPIC-kitchen challenge organizers, we report on the macro metrics (precision and recall) for many shot classes only. They define a many shot class to be a class that has more than 100 samples in the training set. There exits 26 many shot verbs and 71 many shot nouns in the dataset. For actions, the set of many shot actions is the cross product between the many shot verbs and many shot nouns classes given that the action appears at least once in the training set. This gives 819 many shot actions in this dataset.

5.2.1. Results on EPIC-kitchen challenge

We report the results of our model, mentioned as *CAM-State*, in Table 5.2 on EPIC Kitchen dataset for the action recognition task. In our model, we only use RGB channels. As the test sets are not publicly available yet, we compared our results to two baseline techniques, 2SCNN model [76] and TSN model [87], as reported in [14] using RGB channels only.

	Seen kitchens (S1)				Unseen kitchens (S2)			
	Acc T1	Acc T5	Prec.	Recall	Acc T1	Acc T5	Prec.	Recall
	Action							
CAM-State	19.76	36.98	9.83	**10.23**	9.08	19.46	3.68	4.77
2SCNN[76]	13.67	33.25	6.66	5.47	6.79	20.42	3.39	3.01
TSN[87]	**19.86**	**41.89**	**9.96**	8.81	**10.11**	**25.33**	**4.77**	**5.67**
	Verb							
CAM-State	**47.41**	81.33	31.20	20.43	34.35	69.24	15.09	11.00
2SCNN[76]	40.44	83.04	33.74	15.9	33.12	73.23	16.06	9.44
TSN[87]	45.68	**85.56**	**61.64**	**23.81**	**34.89**	**74.56**	**19.48**	**11.22**
	Noun							
CAM-State	28.31	53.77	21.21	22.48	17.48	37.56	10.71	12.55
2SCNN[76]	30.46	57.05	28.23	23.23	17.58	40.46	11.97	12.53
TSN[87]	**36.8**	**64.19**	**34.32**	**31.62**	**21.82**	**45.34**	**14.67**	**17.24**

Table 5.2: Results on the EPIC kitchen dataset of our model compared to baseline methods (2SCNN and TSN) as reported by Damen et al. [14]. All models in this table uses RGB channels only.

Our model has 20M parameters and only 5M trainable parameters which is significantly lower than both competing techniques, i.e. for each input modality: 2SCNN model [76] uses 170M trainable parameters and TSN model [87] has 11M trainable parameters. Even though, our model outperforms 2SCNN model [76] in most reported metrics of actions and verbs, our model is not designed to predict action nouns.

5.2.2. Results on state-changing actions

To evaluate our model on state-changing actions, we report on results from our validation set in Table 5.3. The model is trained to learn state changes and shows better performance on state-changing verbs than on the rest. Confusion matrix is shown in Figure 5.5. The model reports some confusion between semantically similar

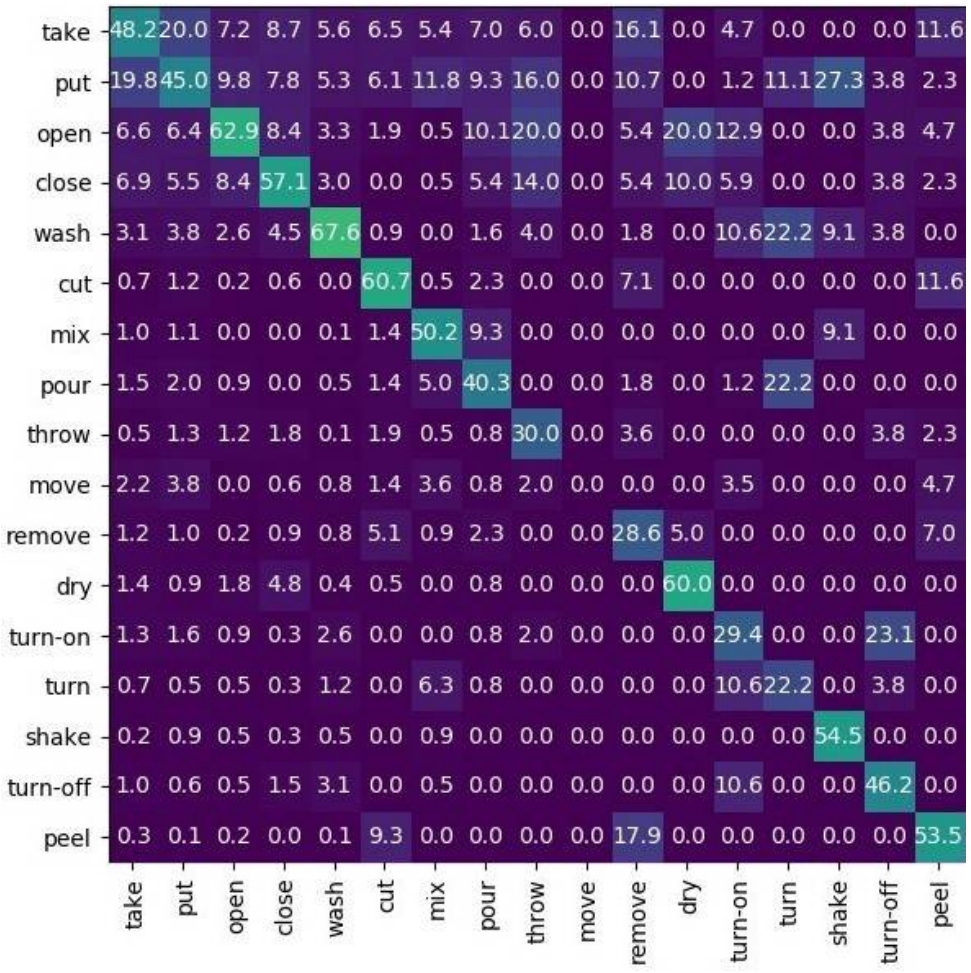


Figure 5.5: Confusion Matrix on the validation set on most frequent verbs.

verbs such as (insert and put, or put and move to) and verbs that have visually similar states such as (wash and fill - where fill examples refers to filling water from the tap). This observation has also been reported recently in [90]. Furthermore, our model suffers from detecting actions that do not change object states (e.g. move and walk).

5.3. Foveated vision for manipulation action

In many cases, recognition can be improved by attention. In order to explore the effects of attention, we modelled the regions around hands as a region of interest

	take	put	open	close	wash	cut	mix	pour	peel	Avg
Precision	56.7	59.3	58.8	39.8	80.1	74.7	68.9	39.1	37.7	57.23
Recall	48.2	45.0	62.9	57.1	67.7	60.7	50.2	40.3	53.5	53.96

Table 5.3: Model performance on validation set on state-changing verbs.

for processing. This is inspired by the human visual system in which a small high-resolution fovea is actively fixated on objects during manipulation actions. Thus instead of passing the full scene to the model, in this section, we discuss concentrating only on foveal regions.

The fovea is a small region at the center of the retina where the vast majority of photoreceptors are concentrated. When performing manipulation actions, human tend to fixate the fovea on the manipulated object in order to control the effects of the action. Inspired by eye foveal vision system, we study the foveal vision for the task of action recognition where hands is detected in the peripheral vision and directs the foveal attention to the hand region. Our intuition is that in the case of manipulation actions, hands are the subject that transforms objects state. This concentrates visual processing on the manipulated objects.

The following section reports on results concentrating visual processing on a region of interest positioned using the hands. To model fovea regions, we crop the scene a crop centred on hands position in the image. As a comparison, we also concentrate processing a central region of the image, regardless of the position of the hands. This is supported by the hypothesis that subject is attending (pointing the camera) to the most relevant part of the scene.

We begin with a review of previous work on using foveation for computer vision. We then discuss how hands regions can be used to define the center of attention for foveated vision during manipulation, providing a comparison with centre-cropped scenes for the fovea for the task of manipulation action recognition on EPIC Kitchen dataset.

5.3.1. Foveal vision

The fovea is a small central pit of the retina responsible for sharp detailed vision. A common approach used by computer vision researchers to simulate the effects of foveal and peripheral vision is to use two cameras per eye: One camera captures the foveal image while the other captures a peripheral image. The foveated camera is used to obtain a high definition image for object detection and other processing, with the peripheral camera used to detect nearby phenomena or to track previously detected objects [24].

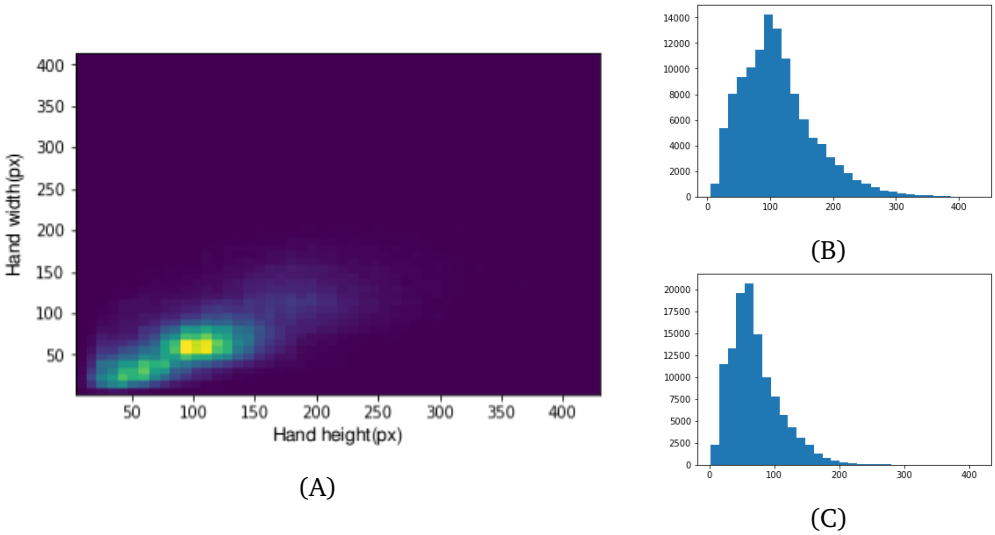
Other efforts to mimic human foveal vision system have used lenses with spatially variant resolution, providing high resolution in the centre for fovea and low-resolution for peripheral region [70]. Researchers have also used zoom lenses, with zoom-in to provide foveal images and zoom-out for peripheral image region. However, this system does not have the advantage of getting the two images simultaneously.

5.3.2. Hands as fovea for manipulation actions

For manipulation actions, the region around the hands is of special interest. A hand detector can be used to focus attention, defining a region of interest that can be fed to a network for recognition and interpretation.

(A) (B) (C)

Figure 5.6: Fovea on frame scene from EPIC kitchen dataset. (A) Full image scene (B) Region around image center. (C) Region around Hand center.



(A) (B) (C)

Figure 5.7: Detected hand sizes in EPIC kitchen dataset. (A) 2D histogram of hands height and width in frames of EPIC kitchen dataset. (B) Histogram of hands height. (C) Histogram of hands width.

To calculate the fovea window size, we need to have an idea of the image scale. In EPIC kitchen dataset, image scale is not provided. The videos are egocentric videos and thus, the image scale changes in the video. We chose to estimate the image scale using the detected hand size in the frame. For hand detection, we used the object masks of Mask-RCNN [29] provided by Baradel et al. [7]. These masks are results of pretrained Mask-RCNN model on COCO Dataset [53] which has the Person as one object class.

The average human hand size is $(h = 180, w = 80)$mm [3] – the average male hand size is $(h = 189, w = 84)$mm, the average female hand size is $(h = 172, w = 74)$mm –. The average detected hand sizes in EPIC kitchen dataset is $(h = 100, w = 70)$ pixels. The histograms of the detected hands are shown in Figure 5.7.

In order to place the region of interest around the hand, we defined a foveal window size that is three hands wide and two hands long ($h = 200, w = 210$ pixels). This window is cropped around the center of the detected hand box. Examples of the cropped scenes are shown in Figure 5.6.

[3] ttp://www.theaveragebody.com/average_hand_size.php

5.3.3. Results on EPIC kitchen dataset

The results of foveal vision on EPIC kitchen dataset are shown in Table 5.4. From the results table, we can see that using image centre as a region of attention in egocentric videos may remove important information for the task of manipulation action recognition compared to the hand-centred fovea. On the other hand, the hand-centred fovea does not perform as well as the full scene. One hypothesis for this degraded performance can be the hand detector in MASK-RCNN as it may fail in finding any hand in the scene. In case of absence of detected hand in a certain frame, we used the image centre instead. We think that this may be one reason of the degraded performance compared to the full scene performance. However, we could not verify the reason of the performance degrade.

	Seen kitchens (S1)				Unseen kitchens (S2)			
	Acc T1	Acc T5	Prec.	Recall	Acc T1	Acc T5	Prec.	Recall
	Verb							
Full scene	**47.41**	**81.33**	**31.20**	20.43	**34.35**	**69.24**	15.09	11.00
Hand region	45.23	80.75	30.72	**24.85**	32.74	67.32	**15.10**	**11.45**
Scene center	34.41	78.37	23.49	17.67	23.62	66.95	11.65	8.67
	Noun							
Full scene	**28.31**	**53.77**	21.21	**22.48**	**17.48**	**37.56**	**10.71**	**12.55**
Hand region	24.39	48.30	**21.75**	20.40	14.27	31.88	9.05	10.68
Scene center	21.12	45.71	18.80	17.23	13.17	30.35	9.71	9.70
	Action							
Full scene	**19.76**	**36.98**	**9.83**	**10.23**	**9.08**	**19.46**	**3.68**	**4.77**
Hand region	14.94	30.71	8.67	7.17	7.75	16.66	3.64	4.09
Scene center	9.92	24.31	5.43	4.66	4.67	13.75	3.37	2.47

Table 5.4: Results on the EPIC kitchen dataset using foveated images on the hands compared to scenes foveated on the image center (Seen and Unseen subsets). Highest values are in bold.

The training process on the foveated images is around three times faster than the full scenes as we cropped the foveated image to half the full image size. This is the case because the hand masks are already precomputed for this dataset. Thus, for the following experiments, we test the ideas on the foveated scenes, while all reported results in this thesis are on the full scene unless specified.

5.4. Generalizability of state-transformations method

State transformation method studied in the previous section concentrated on the use of state transformation for action recognition using a VGG as a backbone. However, this method can be used on top of other stronger backbones specialized on action recognition. In this section, we study one use case of generalizing the idea of learning state transformation for action recognition instead of learning verbs directly. We show how to update one frame-based action recognition architecture to adapt our proposed method of recognizing actions from state-transformations.

5.4.1. State transformations for the TSN model

We use Temporal Segment Networks (TSN) [87] as use case. Wang et al. [87] have developed TSN which is a Two Stream Network, one for RGB and one for optical flow. Here, we concentrate on RGB stream only. The input video to TSN is divided into segments of equal temporal length and they sampled a random frame from each segment. Then, the selected frames are processed using a Two Stream Network. A final result is obtained by aggregating the scores from each segment. However, the TSN reported on EPIC Kitchens [14] includes both predictions of verbs and nouns. For the detection of verbs and nouns, Damen et al. [14] adjusted the output layer of TSN to predict both verb and noun classes jointly, with independent losses.

To implement state-transformations, we do similar adjustment as in [14] to predict noun classes and state classes. However, we do not aggregate the vector scores over time, instead these noun and state vectors are concatenated then used as input to a point-wise convolution to produce one noun vector and 2 state vectors (state transitions) as in figure 5.3. The verb layer is a fully connected layer that takes as

input the noun vector as well as the 2 state vectors. We do not include the action layer in TSN-State model. Actions probabilities are computed as in [14] as follows:

$$p(a = (v, n)) = p(v) * p(n) \tag{5.6}$$

where a, v, n are action, verb, and noun respectively and $p(x)$ is the softmax probability of x.

Training For TSN, we retrained TSN from the pytorch implementation [99]. For TSN and TSN-State, we use only the RGB stream of the TSN network. We use Inception architecture [82] as a backbone with Batch Normalization [33] which is pre-trained on ImageNet [15]. The number of segments is set to 3 in all compared models. All other hyper-parameters are the same hyper-parameters mentioned in [14].

5.4.2. Results on the validation set

To report results in this use case, we use the same training split as in Baradel et al. [7], where participants 1-25 are used for training and the rest are used for validation. All mentioned results in this section are averaged with an ensemble of 4 test runs of the trained model. As metric, we use macro-accuracy computed on Top-1 and Top-5 predictions. The results are reported in table 5.5. From the table, we can see that TSN-State is outperforming TSN in all metrics and more significantly on verb accuracy.

	Verbs		Nouns		Actions	
Method	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
TSN	36.05	77.60	21.36	45.33	9.39	25.15
TSN-State	**39.10**	**78.89**	**21.76**	**46.20**	**10.718**	**26.26**

Table 5.5: Comparison between results of TSN and TSN-State on validation set of EPIC-kitchen dataset. Both models take 3 segments as input.

Varying the number of segments in TSN-State Here we report on TSN-State while varying the number of segments. We experimented with 3, 4, and 5 segments. In table 5.6, we can see adding segments in training improved the results in most measures.

	Verbs		Nouns		Actions	
TSN-State	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
#Seg=3	39.10	**78.89**	21.76	46.20	10.72	26.26
#Seg=4	40.24	78.55	23.06	46.98	11.36	27.61
#Seg=5	**40.39**	78.24	**23.75**	**48.15**	**12.20**	**28.76**

Table 5.6: Varying the number of segments #Seg to TSN-State. Results are computed on the validation set.

5.5. Reversing actions for data augmentation

In this section, we discuss the idea of using the inverse of actions using state transition for data augmentation. To best of our knowledge, this is a novel idea for data augmentation of action segments. It is a straightforward advantage of the definition of state-transition rules for state-changing actions.

5.5.1. Reversible actions

Some manipulation actions can be reversed (e.g. open the door can be reversed to close the door). Sequence samples in Figure 5.8 shows two examples from the dataset where the verb is visually similar but in reversed order. Other actions such as cut tomato and throw in the garbage are not reversible since we cannot find in the dataset a verb that can transform a cut state into the whole state.

We first identify reversible actions from the state transition rules. A reversible action is an action A defined as (S_{pre}, S_{post}) of which there exist in the dataset another action B defined as (S_{post}, S_{pre}). Figure 5.8 shows some examples of the reversible actions extracted from EPIC-kitchen dataset. Then, for each reversible action, we reverse the video samples of these actions and add them to the training set for training as shown in Figure 5.9.

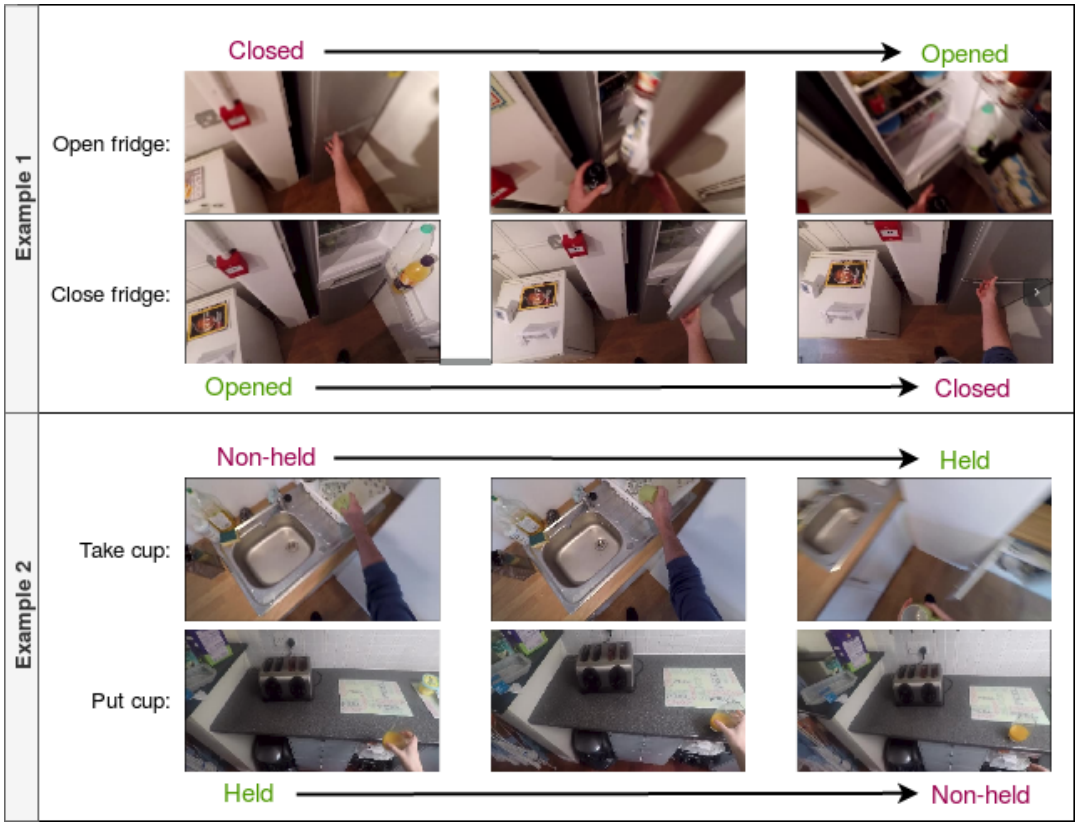


Figure 5.8: Examples of reversible actions extracted from EPIC-kitchens dataset.

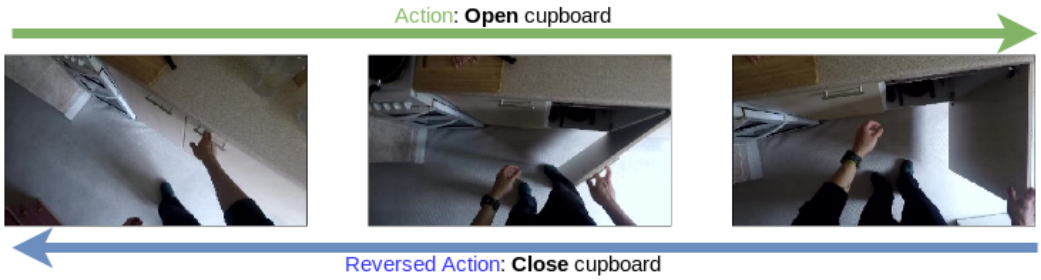


Figure 5.9: Reversing an action sequence for data augmentation

For each action composed of verb v and noun n, a state transition rule transforms the noun's state from S_{before} to S_{after}, and it is defined as follows:

$$R(v,n) : S_{before} \rightarrow S_{after} \text{ where } S_i \in \text{ States}$$

We define the inverse of state transition rules $A^{-1}(v,n)$

$$A^{-1}(v_i,n) : \underset{v_k}{\arg}(S_{k_{before}} = S_{i_{after}} \text{ and } S_{k_{after}} = S_{i_{before}}) \text{ foreach } i,k \in \text{State rules}$$

In order to keep the set of verbs unchanged, we do not add new verbs but restrict each reversed verb to those that exist in the dataset.

The inverse of the state-changing actions results in 14 reversible verbs. The samples of these verbs are added to the training process. Here is the list of identified reversible verbs:

- take $\rightarrow$ put
- put $\rightarrow$ take
- open $\rightarrow$ close
- close $\rightarrow$ open
- turn-on $\rightarrow$ turn-off
- turn-off $\rightarrow$ turn-on
- empty $\rightarrow$ fill
- fill $\rightarrow$ empty
- wrap $\rightarrow$ unwrap
- roll $\rightarrow$ unroll
- fold $\rightarrow$ stretch
- unwrap $\rightarrow$ wrap
- stretch $\rightarrow$ fold
- unroll $\rightarrow$ roll

5.5.2. Results

The intuition is that augmenting the dataset with reversed samples can overcome the imbalanced number of samples per class in the dataset. We report results on two sets: validation and test sets of EPIC-kitchen challenge.

Results on the validation set We report on 2 TSN-State models, one is trained with reversible actions for data augmentation and the other is the same reported in section 5.4 with 4 segments. The results are shown in table 5.7. From the table, we can see an important improvement in the results of training TSN-State with reversible actions for data augmentation compared to the results of the same model without using reversible actions.

	Verbs		Nouns		Actions	
TSN-State	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
w/o Rev	40.24	78.55	23.06	46.98	11.36	27.61
with Rev	**45.18**	**78.66**	**23.39**	**47.00**	**13.50**	**28.88**

Table 5.7: Training TSN-State with and without reversible actions. Results on validation set.

Results on EPIC-kitchen test sets To participate to the challenge, we trained a the TSN-State model using the reversible actions. TSN-state is trained on the same parameters of TSN; this includes the number of segments (3 segments) and using RGB images. The results, in Table 5.8, show that TSN-State trained with reversible actions is achieving better performance compared to the reported performance of the original

	Seen kitchens (S1)				Unseen kitchens (S2)			
	Acc T1	Acc T5	Prec.	Recall	Acc T1	Acc T5	Prec.	Recall
	Verb							
CAM-State[3]	**47.41**	81.33	31.20	20.43	34.35	69.24	15.09	11.00
TSN-State+Rev	45.33	**86.45**	42.82	**24.90**	34.86	74.26	16.32	**11.41**
TSN[87]	45.68	85.56	**61.64**	23.81	**34.89**	**74.56**	**19.48**	11.22
	Noun							
CAM-State[3]	28.31	53.77	21.21	22.48	17.48	37.56	10.71	12.55
TSN-State+Rev	**38.73**	**64.84**	**37.20**	**34.16**	21.13	44.55	**17.19**	16.76
TSN[87]	36.80	64.19	34.32	31.62	**21.82**	**45.34**	14.67	**17.24**
	Action							
CAM-State[3]	19.76	36.98	9.83	10.23	9.08	19.46	3.68	4.77
TSN-State+Rev	**21.34**	**43.97**	**12.27**	**10.70**	9.73	24.89	**5.50**	**5.95**
TSN[87]	19.86	41.89	9.96	8.81	**10.11**	**25.33**	4.77	5.67

Table 5.8: Results of TSN-State with reversible actions on the EPIC kitchen dataset compared to the results of our model CAM-State [3] and original TSN [87]. All models in this table uses RGB channels only. Highest values are in bold.

TSN as well as to CAM-State. This is the case especially on nouns recognition which in its turn boosted the action recognition results. However, for the unseen kitchens, we can not draw a clear conclusion with TSN-State + Rev. Further investigation may be needed to separate the contribution of the state-transformation and the reversible actions in the results.

5.6. Discussion

In this chapter, we reported on results of our method to the recognition of manipulation actions. We based our method on the fact that an object with its attributes is more apparent from a single frame than the action itself. The method proposes the recognition of changes in object attributes from a small set of frames. We implemented this idea using a simple model that predicts objects and their states independently and uses state transition to infer about the action. We reported on results of our model on the challenge of EPIC kitchen dataset and compared these to two baseline techniques.

For the action recognition task, our model outperforms one of the baseline techniques using 34 times less training parameters and achieved comparable results with the other. We showed that our model performed especially well on state-changing actions where the object state can be visualized from a still image. However, the model suffers in recognizing actions such as *move something*, and *turn on the oven*.

We also discussed the idea of adding an explicit attention on the region of the image concerned by the action. We defined this region of the scene as the fovea, and we study two ways of modeling fovea regions from an egocentric point of view: an image-centred fovea and a hand-centred fovea. We showed that for manipulation actions, hand-centred regions are better adapted to the manipulation action task than image-centred regions. While, the results do not outperform recognition using the full scene, foveated vision provides $\sim 3\times$ increase in speed (decrease in computation time). Thus, if a fast way to localize hands is available -as on most Augmented Reality headsets-, using hand-centred regions can be used to trade some performance for the computational cost.

We also showed that our CAM-State model [3] could be adapted to other competitive frame-based techniques for action recognition. In particular, we showed an example of how to use state transformations method on top of the TSN [87] model. We called TSN-State this adapted version of TSN. We also investigated the idea of reversible actions for data augmentation while training. The results on validation set of EPIC-Kitchens shows a significant improvement in the results. The results of training our TSN-State model using reversible actions show that this trained TSN-State model achieved better results than using TSN alone for action recognition. We evaluated our method on the challenge of EPIC-Kitchens dataset.

Chapter 6

Conclusion and Future Perspectives

Many human actions involve manipulating objects in order to change their state. Yet, most common approaches to visual action recognition concentrate on recognizing actions as spatio-temporal motion patterns [85, 36, 10], without regard to the changes these actions may have on objects or the environment. A common result with these techniques is a repetitive list of recognized actions with no information about how or why an action has been performed.

While some human actions may be directly recognized from motion, describing manipulation actions with motion patterns results in an incomplete description. In this research we have sought to complement such approaches with a description of how manipulation actions change objects in the environment and to situate these changes in a richer description of that explains how an action affected the environment.

We have used food preparation as the test domain for our investigation. Food preparation is an appropriate domain for such an investigation for a number of reasons. For one, almost everyone has some experience with food preparation, and sharing of stories about food preparation is quite common. As a result, there is a wealth of available information in the form of recipes and how-to cooking tutorials. Cooking is

also a domain that has received attention from the computer vision community with the publication of datasets such as 50 Salads, EPIC kitchens, and TACoS dataset [68].

6.1. Thesis summary

This thesis is organized as follows: the work starts by defining key terminology in the context of our work in chapter 2, with definitions for terms such as object, action and manipulation action. We then review the existing techniques and challenges for object and action recognition tasks using machine learning. This is followed by a presentation of Convolutional Neural Networks (CNNs) that are used in our solution as well as in a large number of competing methods.

In chapter 3, we studied the visual recognition of objects in the scene, along with their locations and states. Even though many video datasets about cooking have been made available, none has provided annotations for both foodstuff and their states in the scene. To remedy this situation, we have created a new annotated dataset by scraping images from Google Images for foodstuff in a specific state. We used transfer learning to adapt a model previously trained on ImageNet to recognize the food classes and food states in this new dataset. To estimate the location of food classes, we used a weakly-supervised technique that helps the network to locate objects using activation maps. We showed that joint learning of objects and states provides better performance than learning objects and states separately. However, due to the lack of large-scale datasets, the evaluation experiment was performed on a relatively small number of food classes and food states. We believe the availability of a large-scale dataset that considers object states in addition to object classes would help the community in investigating alternative techniques for understanding and evaluating human actions.

In chapter 4, we performed an analysis to study the link between objects and the task of manipulation action recognition. The analysis addressed the question of how information about objects in the scene can contribute to the recognition of manipulation actions. This study assumes a perfect object detector in a video sequence that learns to derive the action happened in that video sequence from the provided information about the objects. We investigated the following object-related information: the temporal order of which objects are present in the scene, the spatial location of these

objects in the scene, as well as their states in each frame. We used different neural network architectures for this analysis, including CNNs, RNNs, and MLPs. The results of this oracle study show that information about the object state has a significant influence on the recognition of manipulation actions. On the other hand, the object location does not seem to improve the performance of the action recognition task.

In chapter 5, we used what we have learned from object recognition and the analysis study to adapt our object detector to action recognition task using state transformations. The method proposes the recognition of changes of object attributes from a small set of frames. We demonstrated that this can provide efficient recognition of manipulation actions.

We evaluated this model by participating in the challenge of EPIC Kitchen dataset. Our model outperforms one of the baseline techniques and achieves comparable results as the other baseline with fewer training parameters. We showed that our model performed better on state-changing actions where the object state can be recognized from a still image. However, the model suffers from recognizing actions such as *move something*, and *turn on the oven*.

In the same chapter, we also studied an idea of foveated vision principle on egocentric videos. Our goal is to compare the fovea in an egocentric scene as the centre of the scene and as the region around the hand. From the results, we have discovered that hand-centred regions are more informative for the task of action recognition than image centre. Even though hand-centred regions do not outperform the full scene, the results show a trade-off between performance and speed. This was our case as we used pre-computed hand regions which allowed us to use hand-centred regions for faster experimenting.

At the end of this chapter, we introduced a novel concept for action data augmentation: reversible actions. The state transition model, as we defined it allows a straightforward discovery of reversible actions and use these action for data augmentation. We also showed a use case where we generalized our state transformation model to one competing methods, TSN [87]. We refer to the adapted model as TSN-State and we showed that training this model with reversible action helps in improving the performance of the original TSN model. However, more investigation is needed to

better understand the effect of reversible actions and state transformation on the original model.

6.2. Limitations and open questions

In this thesis, we propose an alternative method for modeling manipulation actions as changes of object state. We believe that this method is promising. However, our model suffers from detecting non-state changing actions. Methods that recognize actions as spatial-temporal patterns of motion can better detect such actions. Thus, a model that incorporate these two methods of modeling actions in one architecture needs to be investigated. This design can be accomplished using a two-stream pipeline [87, 76, 38, 10, 78] where one stream is dedicated to detecting objects and theirs states in sampled frames, and the other considers the temporal patterns of frame sequences.

One challenging fact about manipulation actions is its semantic ambiguity [90]; actions such as "pour water from the bottle", and "fill the glass with water" can refer to the same action but observed from two different perspectives. For example, the action "pour water from the bottle" changes the state of the bottle from full to empty and at the same time, it changes the state of the glass from empty to full. The ability to associate an object to a certain state may help in avoiding this ambiguity.

This work also opens some questions for further investigations. One question is about the state transition function for each action. We supposed that an action splits the video sequence in half where the first half represent frames from the pre-state and others from the post-states. This assumption may need to be reconsidered, as some actions may happen instantly (not gradually) and change the state of the scene at a specific moment such as "turn on the light".

We believe that the concept of state transformation for action recognition needs more attention from the community. Since the time of performing experiments in this thesis, the State-of-the-Art on action recognition has moved very fast. Generalizing the idea of representing manipulation actions as state transformation to more techniques is one interesting path for future research. Finally, we worked in this thesis in the cooking domain but we believe the usability of this method can be extended to other domains

as well. Domains such as instructional videos [103], medical surgeries, furniture assembly, or even social communications [84] can be potential candidates for an extended evaluation of our method.

6.3. Future perspectives

In this thesis, we have concentrated on recognizing entities and their properties in the scene. From a sequence of frames, we are able to detect and identify a set of interesting human actions and produce a sequence of events. However, this sequence do not provide an understanding of the full story of the video.

A full understanding of human actions requires: recognizing **what** action has been performed, predicting **how** it will affect the surrounding environment, explaining **why** this action has been performed, and **who** is performing it [83]. Approaches to action recognition interpret a spatio-temporal pattern in a video sequence to tell *what* action has been performed, and perhaps *how* and *where* it was performed. A more complete understanding requires information about *why* the action was performed, and *how* it affects the environment. This face of understanding can be provided by explaining the action as part of a narrative.

A narrative is an account of connected events. Finding possible connections within a sequence of events is an essential goal in constructing a narrative account of events. A narrative is not simply a record of a series of events, but a compiled story that situates events within a context. Context enables rich descriptions for events that may not be directly observable, including hypothetical or abstract events, as well as events that occurred in the past.

One application can be an automated **recipe following** of human activities. The system needs to follow someone performing a predefined activity and provide assessment and guidance. Following activities such as furniture assembly and cooking recipe requires not only the visual recognition of human actions but also putting them in the context of the undergoing activity. Thus, detected events need to be interpreted as a causal sequence of voluntary actions, providing a narrative for the implementation of the recipe. Figure 6.1 illustrates the scheme of such a system. It takes a sequence of

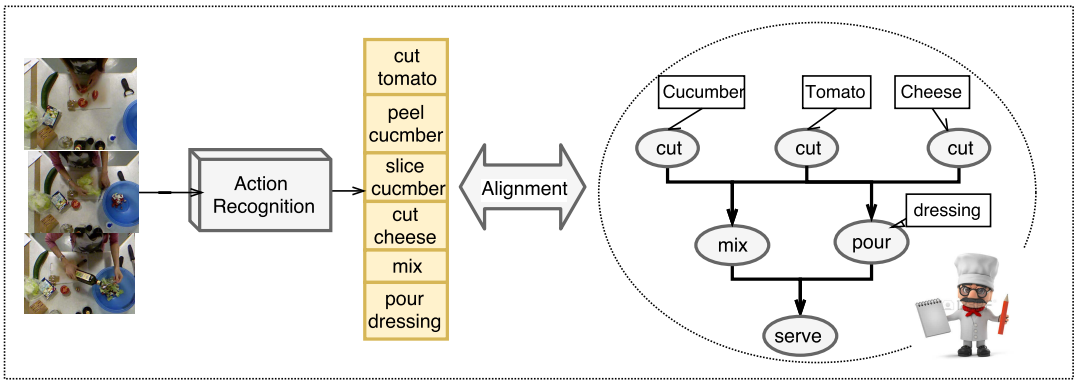


Figure 6.1: Automatic construction of cooking narratives

frames and extracts visual events about ingredients and objects in the scene. These events can be used to align the observed actions with steps in the recipe making it possible to explain why a specific step has been performed.

Scripts of human activities such as recipes for cooking are available in semi-structured textual form. Modeling recipe knowledge in a machine-understandable form helps in following the sequence of events performed for food processing. It also helps in recognizing which recipe is being prepared and why a particular action is performed and thus assist people in completing their activities.

Modeling a story and locating events in this story model has been studied in the literature in two directions: formal methods and statistical methods. Examples of formal logic models are Petri Nets [49], Context-free grammar (CFG) [63, 95, 46, 26], Combinatory categorial grammar (CCG) [96]. Examples of statistical models are Finite State Machines, Hidden Markov models [56, 18, 4], Graph Neural Networks [51]. Even though these methods are very promising, this task is still challenging when applied to semi-structured activities such as recipes. Statistical models require data about all possible ways to reach a goal and logical models require formal modeling of all possible ways.

One reason for why cooking recipes are challenging is the semi-structured nature of the recipe; a recipe goal can be achieved in different ways. Modeling all variations makes the recipe model complex to write and to maintain. On the other hand, a recipe

encodes knowledge that can be used for reasoning (i.e. why a cook is doing a certain action?) and predicting what he/she might do next.

Another application can be the task of **video narration**. Given a silent video, the goal of video narration is to produce a narrated video where the video events are told in synchrony of the video, providing context to the video events to convey the focus message of the whole video. Video narration problem involves both locating events and finding dependencies between the events. It requires the generated story to be coherent and provide an understanding of the full story that the video is telling visually, then paraphrase it into written captions. Narration is different than captioning as it aims to incorporate information about the video context, the reasoning about the story events, and why a certain event has been performed. An example illustrates the difference between video captioning and video narration is demonstrated in figure 6.2.

Being able to narrate a video requires both the understanding the story delivered by the video, and locating video events in the context of this story. The automatic understanding of the video story from videos only is a challenging task. It requires

Figure 6.2: Illustration of differences between video captioning, dense video captioning, and video narration. The frame examples and dense captioning annotations are of a sample from ActivityNet captions dataset [44]. Video narration is the transcript of the voice-over extracted from the original video.

modeling all possible paths (narratives) that can be performed to reach the same story. For example, taking a recipe as a story, to model the recipe of making a salad, the model need to be able to include all correct possible compilations that will lead to making a salad.

Most current datasets for video description [44, 92] use workers force to describe a video. However, narrating a video while watching it for the first time may not be accurate as it does not place the description in the context of the whole video (figure 6.2). The good news is that some instructional videos are generally prepared ahead with very well scripted narrations that are told in the synchrony with the video. Thus, we believe that voice-over of instructional videos is a very promising source for creating a video narration dataset.

Publications and Source code

During the course of this thesis we communicated on our experiments with the following published articles.

- Nachwa Aboubakr, James L. Crowley, and Remi Ronfard. *Recognizing Manipulation Actions from State-Transformations (Technical report)*. The forth international workshop on Egocentric Perception, Interaction and Computing at EPIC@CVPR19. hal-02197595, Jun 2019, Long Beach, USA. (Technical report, accepted for a poster) [2].

- Nachwa Aboubakr, James L. Crowley, and Remi Ronfard. *Recognizing Manipulation Actions from State-Transformations.* The forth international workshop on Egocentric Perception, Interaction and Computing at EPIC@CVPR19. arXiv preprint arXiv:1906.05147, Jun 2019, Long Beach, USA. (Single-blind review, accepted for a presentation) [3].

- Nachwa Aboubakr, Rémi Ronfard, and James Crowley. *Recognition and Localization of Food in Cooking Videos.* CEA-MADiMa 2018-Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management. ACM, Jul 2018, Stockholm, Sweden. pp.21-24, 10.1145 /3230519.3230590. (Double-blind review, accepted for a presentation) [1].

- Nachwa Aboubakr, James L. Crowley. *Histogram of Oriented Depth Gradients for Action Recognition.* ORASIS 2017, GREYC, arXiv preprint arXiv:1801.09477. Jun 2017, Colleville-sur-Mer, France. (Double-blind review, accepted for a poster) [6]. (not included in this thesis).

Here are the list of source code links and project pages developed during this thesis:

- [link] Project page of Action recognition from state transformations [2].
- [link] reproduction code of the Oracle study.
- [link] Project page of Recognition and Localization of Food in Cooking Videos [1].
- [link] EPIC-kitchen dataset viewer: video player with object and action annotations on each annotated frame of the dataset.

Abbreviations

Acc	Accuracy
ANN	Artificial Neural Networks
BN	Batch Normalization
CAM	Class Activation Maps
CE	Cross Entropy
CNN	Convolutional Neural Network
EPIC	Egocentric Perception, Interaction and Computing
FC	Fully Connected layer
FCN	Fully Convolutional Network
FP	False Positive
FN	False Negative
GP	Global Pooling
GAP	Global Average Pooling
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LSTM	Long-Short Term Memory
mAP	Mean Average Precision
MSE	Mean Square Error
MTL	Multi-Task Learning
Prec	Precision
ReLU	REctified Linear Unit
Rcl	Recall
R-CNN	Region proposals CNN
RNN	Recurrent Neural Network
RPN	Region Proposal Network
SPP	Spatial Pyramid Pooling
SSD	Single Shot Detector
TP	True Positive
TN	True Negative
VGG	Visual Geometry Group network
YOLO	You Only Look Once network

Bibliography

[1] N. Aboubakr, R. Ronfard, and J. Crowley. Recognition and localization of food in cooking videos. In *ACM International Conference Proceeding Series*, 2018. ISBN 9781450365376. doi: 10.1145/3230519.3230590.

[2] N. Aboubakr, J. Crowley, and R. Ronfard. Recognizing manipulation actions from state-transformations (technical report). 2019.

[3] N. Aboubakr, J. L. Crowley, and R. Ronfard. Recognizing manipulation actions from state-transformations. *arXiv preprint arXiv:1906.05147*, 2019.

[4] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016.

[5] J.-B. Alayrac, I. Laptev, J. Sivic, and S. Lacoste-Julien. Joint discovery of object states and manipulation actions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2127–2136, 2017.

[6] N. A. Bakr and J. Crowley. Histogram of oriented depth gradients for action recognition. *arXiv preprint arXiv:1801.09477*, 2018.

[7] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori. Object level visual reasoning in videos. In *ECCV*, 2018.

[8] S. Blusseau, A. Carboni, A. Maiche, J.-M. Morel, and R. G. von Gioi. A psychophysical evaluation of the a contrario detection theory. In *2014 IEEE*

International Conference on Image Processing (ICIP), pages 1091–1095. IEEE, 2014.

[9] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[10] 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*, pages 6299–6308, 2017.

[11] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[12] O. Chomat and J. L. Crowley. Probabilistic recognition of activity using local appearance. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 104–109. IEEE, 1999.

[13] J. L. Crowley. A representation for visual information. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1981.

[14] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[15] 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*, pages 248–255. Ieee, 2009.

[16] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[17] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. Long-term recurrent convolutional networks for visual recognition and description. *Proceedings of the IEEE Computer Society*

Conference on Computer Vision and Pattern Recognition, 07-12-June-2015:2625–2634, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298878.

[18] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6341–6350, 2019.

[19] A. Fathi and J. M. Rehg. Modeling actions through state changes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2586, 2013.

[20] F. Fleuret, T. Li, C. Dubout, E. K. Wampler, S. Yantis, and D. Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108(43):17621–17625, 2011.

[21] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[22] X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S.-C. Zhu. Vrkitchen: an interactive 3d virtual environment for task-oriented learning. *arXiv preprint arXiv:1903.05757*, 2019.

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

[24] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Messner, G. Bradski, P. Baumstarck, C. Sukwon, and A. Y. Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. *IJCAI International Joint Conference on Artificial Intelligence*, pages 2115–2121, 2007. ISSN 10450823.

[25] A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. 2007.

[26] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2012–2019. IEEE, 2009.

[27] A. Hashimoto, T. Sasada, Y. Yamakata, S. Mori, and M. Minoh. Kusk dataset: Toward a direct understanding of recipe text and human cooking activity. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 583–588, 2014.

[28] 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*, pages 1026–1034, 2015.

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

[30] Y. He, S. Shirakabe, Y. Satoh, and H. Kataoka. Human action recognition without human. In *European Conference on Computer Vision*, pages 11–17. Springer, 2016.

[31] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[32] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160 (1):106–154, 1962.

[33] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL `http://arxiv.org/abs/1502.03167`.

[34] A. B. Jelodar and Y. Sun. Joint object and state recognition using language knowledge. *arXiv preprint arXiv:1905.08843*, 2019.

[35] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[36] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.59. URL `http://www.ncbi.nlm.nih.gov/pubmed/22392705`.

[37] Z. Jiang, V. Rozgic, and S. Adali. Learning Spatiotemporal Features for Infrared Action Recognition with 3D Convolutional Neural Networks. 2017. URL http://arxiv.org/abs/1705.06709.

[38] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Joint learning of object and action detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4163–4172, 2017.

[39] 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.

[40] 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*, pages 1725–1732, 2014.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[42] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. 2008.

[43] Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018.

[44] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*, 2017.

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

[46] 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.

[47] I. Laptev. On space-time interest points. *International journal of computer vision*, 64(2-3):107–123, 2005.

[48] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[49] G. Lavee, A. Borzin, E. Rivlin, and M. Rudzsky. Building petri nets from video event ontologies. In *International Symposium on Visual Computing*, pages 442–451. Springer, 2007.

[50] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[51] R. Li, M. Tapaswi, R. Liao, J. Jia, R. Urtasun, and S. Fidler. Situation recognition with graph neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4173–4182, 2017.

[52] Y. Li, M. Liu, and J. M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 619–635, 2018.

[53] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[54] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[55] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. Peter Graf. Attend and interact: Higher-order object interactions for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6790–6800, 2018.

[56] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558*, 2015.

[57] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[58] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *CVPR 2009-IEEE Conference on Computer Vision & Pattern Recognition*, pages 2929–2936. IEEE Computer Society, 2009.

[59] K. Nelissen, G. Luppino, W. Vanduffel, G. Rizzolatti, and G. A. Orban. Observing others: multiple action representation in the frontal lobe. *Science*, 310(5746): 332–336, 2005.

[60] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[61] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[62] M. V. Peelen and P. E. Downing. Category selectivity in human visual cortex: Beyond visual object recognition. *Neuropsychologia*, 105:177–183, 2017.

[63] M. Pei, Y. Jia, and S.-C. Zhu. Parsing video events with goal inference and intent prediction. In *2011 International Conference on Computer Vision*, pages 487–494. IEEE, 2011.

[64] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[65] 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*, pages 91–99, 2015.

[66] Z. Ren, O. Gallo, D. Sun, M.-H. Yang, E. Sudderth, and J. Kautz. A fusion approach for multi-frame optical flow estimation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2077–2086. IEEE, 2019.

[67] J. Revaud. *Contributions to a fast and robust object recognition in images*. PhD thesis, 2011.

[68] A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German conference on pattern recognition*, pages 184–195. Springer, 2014.

[69] M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, pages 1–28, 2015.

[70] S. Rougeaux and Y. Kuniyoshi. Robust tracking by a humanoid vision system. In *Proc. IAPR First Int. Workshop on Humanoid and Friendly Robotics, Tsukuba, Japan*, 1998.

[71] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[72] E. L. Schwartz. Spatial mapping in the primate sensory projection: analytic structure and relevance to perception. *Biological cybernetics*, 25(4):181–194, 1977.

[73] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360. ACM, 2007.

[74] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision*, 2017.

[75] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484, 2016.

[76] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[77] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICRL)*, 2015.

[78] 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. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1970, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.216. URL `http://ieeexplore.ieee.org/document/7780585/`.

[79] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[80] S. Stabinger, A. Rodríguez-Sánchez, and J. Piater. 25 years of cnns: Can we compare to human abstraction capabilities? In *International Conference on Artificial Neural Networks*, pages 380–387. Springer, 2016.

[81] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. ACM, 2013.

[82] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `http://arxiv.org/abs/1409.4842`.

[83] M. Thioux, V. Gazzola, and C. Keysers. Action understanding: how, what and why. *Current biology*, 18(10):R431–R434, 2008.

[84] P. Vicol, M. Tapaswi, L. Castrejon, and S. Fidler. Moviegraphs: Towards understanding human-centric situations from videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[85] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

[86] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin. Action recognition by dense trajectories. 2011.

[87] 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*, pages 20–36. Springer, 2016.

[88] X. Wang, A. Farhadi, and A. Gupta. Actions $\sim$ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667, 2016.

[89] X. Wang, S. Zheng, R. Yang, B. Luo, and J. Tang. Pedestrian attribute recognition: A survey. *arXiv preprint arXiv:1901.07474*, 2019. URL `https://arxiv.org/abs/1901.07474`.

[90] M. Wray and D. Damen. Learning visual actions using multiple verb-only labels. *arXiv preprint arXiv:1907.11117*, 2019.

[91] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatio-temporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[92] J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[93] M. Xu, M. Gao, Y. T. Chen, L. S. Davis, and D. J. Crandall. Temporal recurrent networks for online action detection. *arXiv:1811.07391*, 2018.

[94] W. Xu, Z. Miao, J. Yu, and Q. Ji. Action recognition and localization with spatial and temporal contexts. *Neurocomputing*, 333:351–363, 2019.

[95] Y. Yang, A. Guha, C. Fermuller, and Y. Aloimonos. A cognitive system for understanding human manipulation actions. *Advances in Cognitive Sysytems*, 3: 67–86, 2014.

[96] Y. Yang, Y. Aloimonos, C. Fermüller, and E. E. Aksoy. Learning the semantics of manipulation action. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on*

Natural Language Processing (Volume 1: Long Papers), pages 676–686, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1066. URL https://www.aclweb.org/anthology/P15-1066.

[97] G. Yao, T. Lei, and J. Zhong. A review of convolutional-neural-network-based action recognition. *Pattern Recognition Letters*, 118:14–22, 2019.

[98] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *2009 IEEE 12th international conference on computer vision*, pages 492–497. IEEE, 2009.

[99] X. Yuanjun. PyTorch Temporal Segment Network. https://github.com/yjxiong/tsn-pytorch, 2017.

[100] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015.

[101] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[102] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition*, 2016.

[103] L. Zhou, C. Xu, and J. J. Corso. Towards automatic learning of procedures from web instructional videos. *arXiv preprint arXiv:1703.09788*, 2017.

[104] Y. Zhou, B. Ni, R. Hong, M. Wang, and Q. Tian. Interaction part mining: A mid-level approach for fine-grained action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3323–3331, 2015.