DISSERTATION

SOMETHING IS FISHY! - HOW AMBIGUOUS LANGUAGE AFFECTS GENERALIZATION OF VIDEO ACTION RECOGNITION NETWORKS

Submitted by

Dhruva Kishor Patil

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: J. Ross Beveridge
Co-Advisor: Nikhil Krishnaswamy

Francisco R. Ortega
Benjamin Clegg

Copyright by Dhruva Kishor Patil 2022

All Rights Reserved

ABSTRACT

SOMETHING IS FISHY! - HOW AMBIGUOUS LANGUAGE AFFECTS GENERALIZATION OF VIDEO ACTION RECOGNITION NETWORKS

Modern neural networks designed for video action recognition are able to classify video snippets with high degrees of confidence and accuracy. The success of these models lies in the complex feature representations they learn from the training data, but the limitations of these models are rarely linked on a deeper level to the inconsistent quality of the training data. Although newer and better approaches pride themselves on higher evaluation metrics, this dissertation questions whether these networks are recognizing the peculiarities of dataset labels. A reason for these peculiarities lies in the deviation from standardized data collection and curation protocols that ensure quality labels. Consequently, the models may learn data properties that are irrelevant or even undesirable when trained using only a forced choice technique. One solution for these shortcomings is to reinspect the training data and gain better insights towards designing more efficient algorithms.

The Something-Something dataset, a popular dataset for video action recognition, has large semantic overlaps both visually as well as linguistically between different labels provided for each video sample. It can be argued that there are multiple possible interpretations of actions in videos and the restriction of one label per video can limit or even negatively impact the network's ability to generalize to even the dataset's own testing data. To validate this claim, this dissertation introduces a human-in-the-loop procedure to review the legacy labels and relabel the Something-Something validation data. When the new labels thus obtained are used to reassess the performance of video action recognition networks, significant gains of almost 12% and 3% in the top-1 and top-5 accuracies respectively are reported. This hypothesis is further validated by visualizing the layer-wise internals of the networks using Grad-CAM to show that the model focuses on relevant salient regions when predicting an action in a video.

ACKNOWLEDGEMENTS

I came to CSU on August 9th, 2015 and immediately the next day, I sent an email to Dr. Ross Beveridge asking 'I intend to do a PhD in Computer Vision, can you please guide me?'. Today after 7 years, I stand at a point in my life where my dream is at its completion. This journey has been most exciting and truly rewarding in every regard, and possible only because of the people in my life.

I would like to expresses my gratitude to Dr. Nikhil Krishnaswamy for giving me the freedom to run with my ideas and for being the constant support I needed during my anxious moments when I hit a wall. I hope I am still able to compile our conversations as a souvenir of my research. I am immensely grateful to Dr. Ross Beveridge, who trusted me with my research and inspired me to look at a problem differently and not follow the rat-race. He instilled in me the two guiding principles of Vision Group that will stay with me forever: Bumper Sticker and Elevator Talks. A special thanks to Dr. Francisco R. Ortega, who always considered me a part of his family and looked out for me as one of his. Your support will always mean a lot to me. The one comment from Dr. Ben Clegg gave me a deep confidence within myself when he said 'Your writing was so easy to understand, that I understood BERT today!'. Thank you for your guidance, Dr. Clegg.

This dissertation would not have been possible without the loving support of my family. Thank you Baba, for being a constant pillar of support during my last stage experiments. Thank you Dada, for asking me tough questions and pushing me beyond my comforts. And thank you Aai and Vahini, who put up with my craziness through all this. A special mention to my friends Aditya Raikwar, Aniket Parandkar, Jason Yu, Sean Chen and Mark Hinds. I would also like to mention Dr. Bruce Draper who instilled in me the love for Computer Vision. I am grateful for the constant support of the CS department over all these years.

And last but not least, a special mention to my partner in crime, Lauren, for everything. You taught me to enjoy life and you make my life beautiful. I am extremely lucky to have you by my side.

DEDICATION

I would like to dedicate this dissertation to my family, friends and my mentors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Actions form an essential part of non-verbal communication between two or more people. In a given context, actions can convey a definitive meaning to a conversation. Differences in background, illumination, camera motion or orientation can influence how an action is perceived or interpreted by humans and machines alike, considering that actions have a large range of motion. Actions involving human-object interactions are particularly interesting because there is a correlation between the positions of the human and objects over time. How a human interacts with an object informs us of its properties, specific regions where the object can be handled *i.e.* affordances, as well as the effect of interaction of that object with other objects *e.g.* cutting a carrot placed on a board with a knife. These factors largely impact what eventual action is performed by the individual. The numerous reasons stated above help us understand why action recognition is a challenging topic, one that has long been studied and progressed over the years.

An image freezes this action in the moment, but a video allows it to progress sequentially and more gracefully. A static image or a video frame can give us details about various spatial elements like objects, actors, scenes, etc. Videos involve a temporal dependency of past, current as well as future frames. Between those frames, there is an interaction between different elements to form a meaningful action. These complex spatio-temporal interdependencies make it a challenging task to design models that can effectively solve action recognition in videos.

The data collection and curation of datasets in Computer Vision follow a standardized process that ensures quality of the data fed to models for training and evaluation. This process involves searching for images or videos of classes on the internet and getting them reviewed by human labelers typically on a crowd-sourced platform. The review is posed as a binary identification problem wherein human labelers confirm if the sample belongs to the associated class or not. One

important aspect to note here is that the quality of the original data sourced is rich because it is mined from expert websites. A second layer of quality control is induced when a labeler reviews it, resulting in a dataset with an acceptable label noise.

The Something Something dataset is a popular video action recognition dataset focused on understanding human-object interaction in videos. Contrary to established routines of data formulation, the conception of this dataset takes a different approach. Users on a crowd-sourced platform like MTurk are given a sentence i.e. a video label and asked to record a video based on the label. The given sentence is also known as a caption template. This caption template is of the form *something* action *something* where the user can select an action from a range of descriptions. The user also uploads the object name used during recording of the video. The result of such an approach can be a large variation of the quality of the videos in the dataset. While a particular video seen individually may not be wrong, it might not be close to the expected label. There is lot of overlap between two labels in both visual as well the linguistic sense. For example, *picking something up* and *lifting something up completely without letting it fall*. The two labels are semantically very close to each other, but the variation in the interpretation of characteristics like affordance can result in a distinction between the two classes. Similarly, examples like *moving something up* vs *picking something up* vs *lifting something up completely without letting it drop down* highlight the linguistic similarity between the labels. This gives us to understand that solely based on the visual appearance of the progression of an action in a video, it can have more than one correct labels.

Modern neural networks designed for the task of video action recognition are able to classify actions in video snippets with high confidence and accuracy. Ideally, we would expect the current state of the art networks to be able to disambiguate between the visual and linguistic subtleties of different classes in the Something Something dataset. However, studying the pattern of misclassifications reveals the systematic inaccuracies in the source itself, *i.e.* the videos and the corresponding legacy labels. This dissertation tries to address these data inconsistencies and presents a robust way of data reannotation using a multi-person assessment of legacy labels. Further, this

dissertation attempts to shed light on the idea that the existing video action recognition networks warrant a greater trust in their ability to generalize than what they are assumed to.

1.2 Current Techniques and Shortcomings

The study of video action recognition has seen a steady progress in research and development over the years. This domain has been revolutionized due to the recent advancements in the field of deep neural networks. Before the advent of deep learning methods, handcrafted features were used to classify actions in videos [3–5]. These handcrafted features were domain specific and years of research went into developing efficient features. Deep neural network architectures quickly replaced the handcrafted features due to their ability to automatically learn abstract feature representations of video frames.

Convolutional Neural Networks (CNN) are the state-of-the-art deep learning approach for extracting feature representations from a video frame [6–9]. Karpathy *et al.* [1] and Donahue *et al.* [10] laid the foundations of using CNNs for video classification. The approaches computed the embeddings of every frame in the video using a 2D CNN architecture and modeled the temporal information in the video sequence using fusion strategies like Long Short Term Memory(LSTM) [11].

Simoyan and Zisserman proposed the Two Stream Network [12] which was later augmented by Ng *et al.* [13]. This approach processed spatial information by an RGB stream and temporal information, or motion between frames using optical flow fields. Despite the many successes in static image feature representations, 2D CNN attempts had a major limitation. The temporal information in videos was squashed within the first few 2D convolutional layers. Further attempts at better model designs aimed to preserve the temporal information by making changes to the convolution operations in the architecture.

Tran *et al.* added a third dimension to the convolutional kernels to preserve the temporal information when they proposed 3D Convolutional Networks (C3D). The C3D model was efficient, yet computationally very expensive. This parameter explosion was solved by Carreira and Zisserman

with the I3D model [14]. The core idea inflated the filters and pooling kernels from 2D to 3D and leveraged the pretrained weights used for training ImageNet dataset. This model for video action recognition had significantly fewer parameters than C3D (25M vs 79M). Similar networks aimed at achieving the efficiency of 3D CNNs and the computational complexity of 2D CNNs [15–17].

An idea similar to the Two Stream network was proposed by Feichtenhofer *et al.* with the SlowFast network [18]. Compared to the Two Stream network, this approach had two major differences. First, instead of using optical flow as motion information as an additional modality, the model only used raw data as input. Second, the network had two pathways that selectively focus on processing the spatial and temporal information. The Slow pathway processed spatial information, while the Fast pathway processed temporal information. This design was inspired by the human vision.

Around the same time as the SlowFast algorithm, Lin *et al.* proposed a Temporal Shift Module (TSM) which shifted the channels along the temporal dimension. In an offline setting where a complete video segment is available, a channel shift ensures that any time instance has information from the current frame as well as previous and future frames. This approach is one of the state-of-the-art networks achieving a very low processing latency and low trainable parameters. This research analyzes the predictions obtained from SlowFast and TSM to understand the nature of misclassifications seen in video action recognition.

In analysing videos, humans can often reason about complex scenes and interactions based on visual and linguistic information. The popular datasets for action recognition e.g. Kinetics [19], UCF-101 [20], HMDB-51 [21] have labels that comprise of a single word or two word description of the action. The Something Something dataset [22] is another dataset that provides fine-grained textual descriptions of actions involving human object interactions. However, it has a more complex description of labels in the form of *something* action *something* where *something* serves as a placeholder for the objects. This is denoted as a caption template by the authors. This dissertation investigates the impacts of the complex video labels in the Something Something dataset on the classification performance of video action recognition models.

Recent studies using the Something Something dataset discuss shortcomings in the network architecture design, but none investigate the impact of dataset shortcomings on model performance. This research direction has been very recently pursued in the image domain with ImageNet [23]. The authors Beyer *et al.* , in [24] argue that images in the ImageNet dataset have multiple objects present, but only a single corresponding label for one of those objects. This adversely affects the performance of image classifiers networks that might predict one of the many objects, but still be penalised because it does not match the human labeled object in the validation data. To mitigate this issue, the authors introduce a human-in-the-loop method to disambiguate labeling flaws. The Reassessed Labels or ReaL consists of all objects seen in the image as reviewed by the annotators. The popular networks are then re-evaluated; not re-trained, and their systematic gain in performance is reported. While this dissertation closely resembles the above approach, it identifies the same pattern of data collection and curation inconsistencies in the validation data with reference to videos. Compared to multiple objects in images, the source of ambiguities the Something Something dataset stems from the visual and linguistic overlap of videos and their corresponding labels. A comparative systematic gain is reported when the networks are re-evaluated thus implying that relevant properties are learnt by them.

The success of CNNs lies in their ability to model complex feature representations learned from training data. Despite their superior performance, it becomes a hindrance that there are no easily interpretable and understandable explanations when these networks fail catastrophically at a particular task. To understand why a network predicts what it predicts, there have been many recent attempts at decomposable visualizations of the feature representations of CNNs [25]. Selvaraju *et al.* in [25] introduce Gradient-weighted Class Activation Mapping (Grad-CAM) to produce a visual explanation of the final convolutional layer. This approach highlights the salient regions in an image that contribute positively to the network's decision in classifying a particular class for the image. This work uses the inferences of Grad-CAM for a single image, and extends it to understand how the reference models predict an action within a video.

1.3 Summary of Research

The goal of this dissertation is not to propose another state of the art model for video action recognition. It takes a step in the direction less explored, and delves into addressing the shortcomings of the dataset that is used to train the networks. The focus of this study is the Something Something dataset, a popular video dataset focused on human-object interaction. The labels are of the form *something* action *something* where something is a placeholder for the object.

There is a large overlap in the linguistic and visual nature of the videos labels and corresponding videos across different classes. In order to examine this claim, this research looks at the closeness of the labels when they are transformed to a numerical representations. This step is intended to bring the natural language perceptions about legacy labels into a visually interpretable domain. SentenceBERT [26] a natural language embedding architecture is used for this transformation. It takes an individual label from the dataset and outputs a fixed size encoding of it. Subsequently, all the legacy labels now transformed into this linguistic embedding space are plotted onto a TSNE plot [27] to examine different correlations and overlaps between labels. Cosine similarity is used as a measure of similarity between the two representations. This visualization helps to understand how seemingly uncorrelated natural language labels are positioned so close to each other that it triggers potential confusions in the model predictions.

Next, the predictions of two popular state of the art architectures for video action recognition are analysed: SlowFast [18] and TSM [16]. The resultant sample set is obtained by the union of all misclassifications in both top-1 and top-5 categories of the two networks. A pilot experiment is set up wherein users are asked to select all possible descriptions of a video from the available choices. This study revealed that 84.6% of samples had multiple selections for videos. The results of this study allowed the scaling up of video samples to include both top-1 and top-5, totalling approximately 11,857 videos for review. The scaled up human-in-the-loop pipeline, is released as a video annotation task on Amazon MTurk, a crowd-sourcing platform. Each video is presented with options obtained from the intersection of predictions from the two networks. Every individual

worker, *i.e.* the human labeler, selects all the options that describe the video displayed. To mitigate the influence of bias caused by faulty selections by participants, it is ensured that each video is reviewed by three participants. Figure 1.1 shows an illustration of the annotation task as seen by the workers on MTurk.

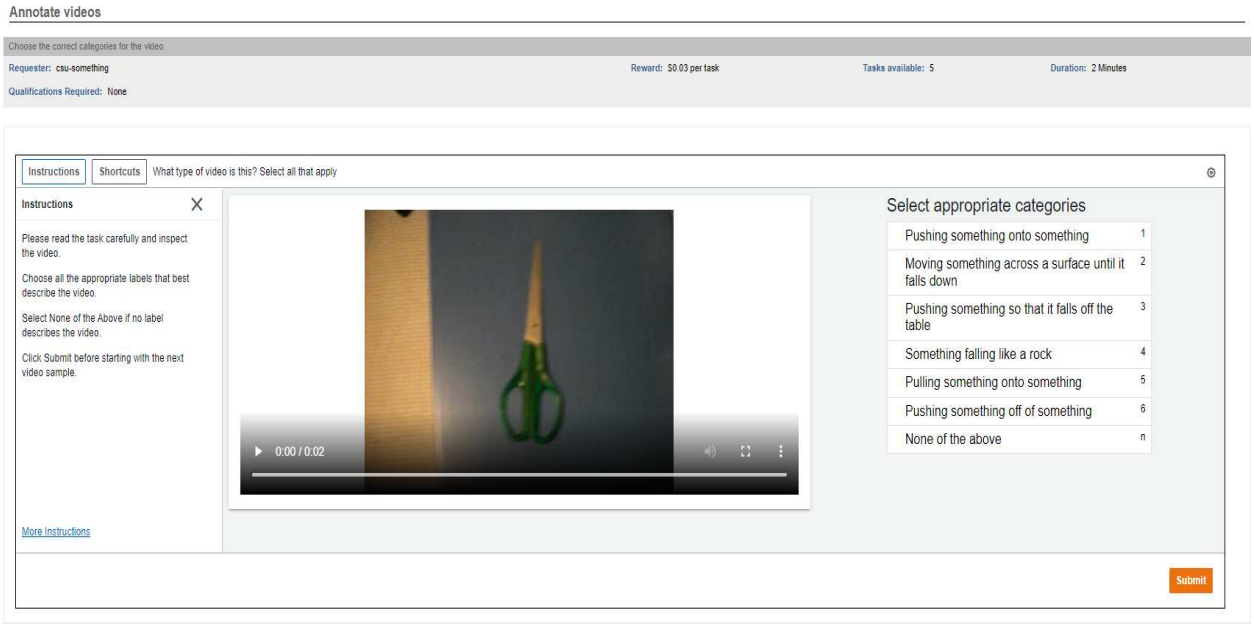


Figure 1.1: An illustration of the video reannotation tool as presented to the workers on Amazon MTurk, a crowd-sourcing platform. Each worker is shown a video and the corresponding labels describing the video. Workers are tasked with selecting all the relevant labels that describe the video. The options available are obtained from the union of the predictions from SlowFast and TSM.

The reannotated video labels thus obtained are consolidated for further analysis. MTurk requires approving or rejecting a worker's task so that they get paid. If a task is approved, the worker is paid, and rejected otherwise. This middle step that needs to be carried out for quality control of worker annotations, is facilitated by a tool developed called ReviewMe (2.4). Every video sample thus has four passes that it goes through: three from MTurk workers, and one to check the relevancy of all three by using the ReviewMe tool. The worker task verdict is leveraged to reassign the annotated labels to the samples using a Majority Voting of the approvals. If the original label is present along with any additional labels from multiple participants, then the sample under consideration gets all the additional labels. Similarly, if the target video label is not present in either

of the selections, but a common label is seen, then the label of the video sample will be updated to include both labels once reviewed. Lastly, the new set of labels obtained for the validation data is denoted as REAssessed Labels or ReaL data, similar to the notation in [24]. The two models are then re-evaluated with the ReaL data for changes in the performance. An increase of almost 12% in the top-1 accuracy and 3% in the top-5 accuracy is observed for both networks when they are reevaluated on the revised data.

To investigate what the networks learn within each convolutional layer, the layer-wise visualizations using Grad-CAM [25] are explored. For a given class index corresponding to the label in the dataset, activation maps are obtained for the video in the forward pass and gradients from backward pass of the layer with respect to the input. A weighted score is obtained for each feature map in the layer under consideration by global average pooling each pixel in the map. Only the regions that have a positive effect on the class's output are visualized. A heatmap of the weighted feature maps is overlayed on every respective video frame. Figure 1.2 shows selected frames from the original video and the corresponding heatmaps obtained from Grad-CAM overlaid on the video using the TSM network. The visualization gives an insight into how TSM network focuses on salient regions in the video with the class label *moving something down*. The network focuses on the main object that is interacted with as seen from the highlighted regions in the bottom row of the figure. Also, the model has awareness of the future frames which is evident from the focus on the bottom part of the object, or the trajectory of moving it down.

1.3.1 Contributions

This dissertation investigates the systematic inaccuracies in performances of video action recognition models caused due to ambiguous language and visual overlap between distinct classes. The main contributions of the work are:

- Exploration and visualization of different natural language embedding techniques of labels in the Something Something dataset [22].

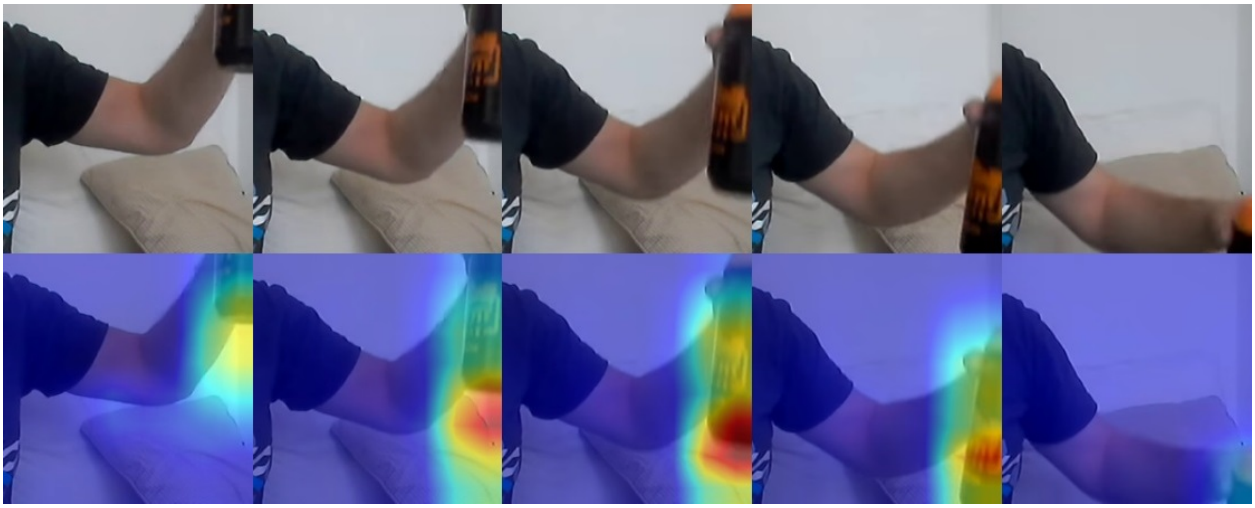


Figure 1.2: Grad-CAM visualization of a video with class label *moving something down* as processed by TSM. The network focuses on the main object(s) that is interacted with as seen from the salient regions in the bottom row of the figure. Also, the model has awareness of the future frames which is evident from the focus on the bottom part of the object, or the trajectory of moving it down.

- Comparisons and investigation into the misclassifications by two state-of-the-art networks for video action recognition for Something Something dataset: SlowFast [18] and TSM [16].

- A robust human-in-the-loop procedure to review labels of video samples in the validation set using MTurk, a crowd-sourced platform. The results are aggregated using Majority Voting relying on worker approval.

- The Reassessed Labels thus obtained used to re-evaluate SlowFast and TSM and the metric is called ReaL accuracy. The ReaL accuracy reports an increase of 12% top-1 and 3% top-5 over the original validation accuracy.

- A visual investigation into each convolutional layer of SlowFast and TSM using GradCAM. This approach examines the failures of the model as well as a comparison of model learning for semantically similar class labels.

1.3.2 Scope

Many factors can impact the performance of a video action recognition system. Unfortunately, evaluating all factors is beyond the scope of this dissertation. This dissertation does not focus on the evaluation or optimization of any of the following:

- Exhaustive video action recognition models: To examine the validity of our hypothesis, I chose two state of the art networks for analysis: SlowFast [18] and TSM [16]. Other more recent networks like MViT [28] have not been explored.

- Variability of datasets for video action recognition: Given the rich correlation between the labels, my study is focused only on the Something Something dataset as there is a large overlap of different labels both linguistically as well as visually. Other popular video action recognition datasets have not been explored.

- The focus of this dissertation is on reassessing and reannotating the labels in the validation set. Samples in training data have not been considered given the influence of data augmentation like random cropping and resizing that can remove the main object interacted with during training.

- Data augmentation techniques: Only resizing and random cropping for training and center cropping for testing have been used as discussed in Section 4.5.1. The classes in the dataset rely heavily on directionality of actions (left-to-right, right-to-left). Thus horizontal flipping is not considered in data augmentations.

- The use of GradCAM [25] for visualization as a proof of concept of the internal workings of networks. Other visualization techniques like temporal-GradCAM [29], GradCAM++ [30] and XGradCAM [31] have not been considered in this research.

- A TSNE visualization of legacy labels in the Something Something dataset is presented using SentenceBERT [26] as motivation for the main work. Other embedding techniques have not been considered for the given scope of the dissertation.

1.3.3 Document Outline

The rest of the document is structured as follows: Chapter 2 discusses the prior literature of the techniques used to design video action recognition models, the evolution of embeddings in natural language processing as well as the investigative visualization techniques used to examine the internal workings of CNN architectures. Chapter 3 briefly introduces the background concepts necessary for the reader to understand the basis of our experiments. This chapter also gives an overview of the popular datasets for video action recognition, and describes the dataset in focus, Something Something dataset, in a greater detail. Chapter 4 explains the approach to study the misclassifications of video action recognition networks and the patterns of confusions between class labels. This chapter also introduces the human-in-the-loop approach to obtain annotations for re-evaluation of the models. Chapter 5 describes the experimental setup and case studies of few examples as predicted by the existing and proposed models. It highlights the importance of broadening the scope of the networks with regard to the allowed class labels for the videos. Additionally, this chapter illustrates the failure cases of the networks and the patterns of confusions between visually and linguistically similar classes. Chapter 6 concludes the dissertation with a discussion and research ideas for future work that can be undertaken in this area.

Chapter 2

Prior Literature

This chapter aims to provide the reader with the necessary background of literature to understand the following chapters. Readers may want to skip to subsections pertaining to their interests. Section 2.1 explains the evolution of different techniques to design networks to classify actions in videos. Section 2.2 explains the concept of Human Object Interaction and the various attempts aimed at understanding the local information of the objects and the actors to classify actions in videos. Section 2.3 discusses the embedding techniques used to convert the domain of natural language into numerical data for analysis using the concept of embeddings. Section 2.4 describes the recent attempts to look at the shortcomings of the data and its effects on the model performance. Section 2.5 discusses the attempts aimed at a visual investigation of the internal workings of popular CNN architectures.

To summarize, this work takes advantages of numerous methods mentioned in this chapter. First, the use of SBERT to visualize the linguistic closeness of labels in the Something-Something dataset to each other (2.3). Further, SlowFast and TSM are run in inference mode to filter out the misclassifications for review (2.1). Inspired by the work with ReaL accuracy, a crowd sourced review and reannotation of the legacy labels in the validation set is proposed. The resulting new label are used to reevaluate the networks (2.4). As an additional validation of the correctness of model predictions, the visualization of layer-wise salient regions of networks using Grad-CAM when they predict an action class for the label is presented (2.5).

2.1 Video action recognition

Video action recognition in the domain of Computer Vision is largely influenced by factors like differences in background, color, illumination, camera motion, orientation, etc. Additionally, there is an intra class variation based on how different people perform the same action as well as an inter class variation or similarity between different actions. While a static image freezes an action in

the moment, a video brings in a spatio-temporal dependency between the frames. These various reasons help us understand why video action recognition is a challenging topic, one that has long been studied and progressed over the past decade.

The different attempts to classify actions within videos have been revolutionized by the use of deep neural networks. Before the advent of deep learning methods, handcrafted features were used to classify actions in videos. These handcrafted features were domain specific, and years of research went into developing efficient features. The features quickly became out of date as modern deep neural network architectures started automatically learning abstract feature representations of video frames. Presented below is an examination of the evolution of video action recognition. We will briefly dwell in the handcrafted features and then discuss the more recent neural network approaches.

2.1.1 Handcrafted features for video action recognition

Previous strategies for video classification involved building a classifier trained on a "bag of features". These features, known as handcrafted features, required a deep understanding of the key elements in images and videos for their formulation. The features, can either be sparse, as proposed by Dollar [32] and Laptev [33, 34] or dense as proposed by Wang in improved Dense trajectories (iDT) [5]. iDT was able to achieve the state of the art results on HMDB51 dataset (60.1%)[3.2] in 2011. The iDT features are notable due to their competitive performance with several deep learning models. Building on 2D optical flow by Efros [3], the computation of 3D optical flow by Ballin [4], continues to influence many video understanding models today. The introduction of Convolutional Neural Networks (CNN) in 2012; however, transformed the domain of visual understanding.

2.1.2 Deep learning for video action recognition

Deep neural networks quickly garnered tremendous attention after their use by Krizhevsky *et al.* for image classification on ImageNet [6]. The deep network's ability to automatically learn feature representations was a huge improvement from the once tedious process of handcrafting

features. The research community working on video action recognition quickly incorporated deep neural networks and achieved high results on prevalent datasets. A review of some of the prominent strategies employed for understanding videos via deep neural networks is discussed below. This section is described as 'vanilla action recognition' because despite the evolution of the architectures to address shortcomings of previous approaches, the inherent objective to improve the classification performance of human actions in videos remains consistent. Hence the name 'vanilla'. Modern vanilla action recognition architectures can be studied in primarily five categories. Table 2.1 describes these approaches, as well as the development of these methods over the years. As a benchmark, the performance of these models is demonstrated on UCF101 dataset, a popular dataset for action recognition[3.2].

2.1.2.1 2D Convolutional Neural Networks

In 2014, Karpathy *et al.* [1] laid the foundations of using CNN for video classification. They proposed using two streams for processing information in the spatio-temporal domain. The context stream processes frames at a lower resolution and fovea stream processes frame at a higher resolution center crop. These activations from both streams are concatenated and fed to the first fully connected layer; however, the convolution operation on the frames in the first layer, results in an image, thereby essentially losing the temporal information. The papers in video action recognition that followed used a better way to model the temporal information within the video for action classification.

Donahue *et al.* [10] proposed the Long term Recurrent Convolutional Network (LRCN) which addressed a major shortcoming of efficient modeling of long term dependency of temporal information within videos. The CNN features computed for every frame, were passed to a Long Short Term Memory (LSTM) [11] for sequential modeling. Figure 2.1 shows the design of the LRCN architecture. This strategy demonstrated the advantage of using LSTM, by showing the improvement of almost 17% compared to the CNN model by Karpathy *et al.* [1] on RGB frames. While this architecture appealed to the research community with its use of CNNs for spatial modeling and

Table 2.1: There are five main approaches to video action recognition using deep neural networks. The table provides a brief overview of the methods, the papers as well as the comparative performance of these networks on UCF101, a popular dataset for video action recognition. For two methods, the authors did not evaluate on UCF101, and hence it is left blank.

Approach	Author (*et al.*)	Year	Performance (UCF101)
2D CNN	Karpathy [1]	2014	65.4%
	Donahue [10]	2015	82.6%
	Kar [35]	2017	93.2%
Two Stream	Simoyan [12]	2014	88.0%
	Ng [13]	2015	88.6%
	Wang [36]	2016	90.3%
	Wang [37]	2018	94.9%
	Feichtenhofer [18]	2019	-
3D CNN	Tran [38]	2015	83.4%
Inflated CNN	Carreira [14]	2017	**98.0%**
	Girdhar [39]	2019	-
Improved 3D CNN	Xie [15]	2018	96.8%
	Lin [16]	2019	95.9%
	Jiang [17]	2019	96.2%

RNNs for sequence modeling, it was very expensive to train due to the high number of parameters, and limited training data available then.

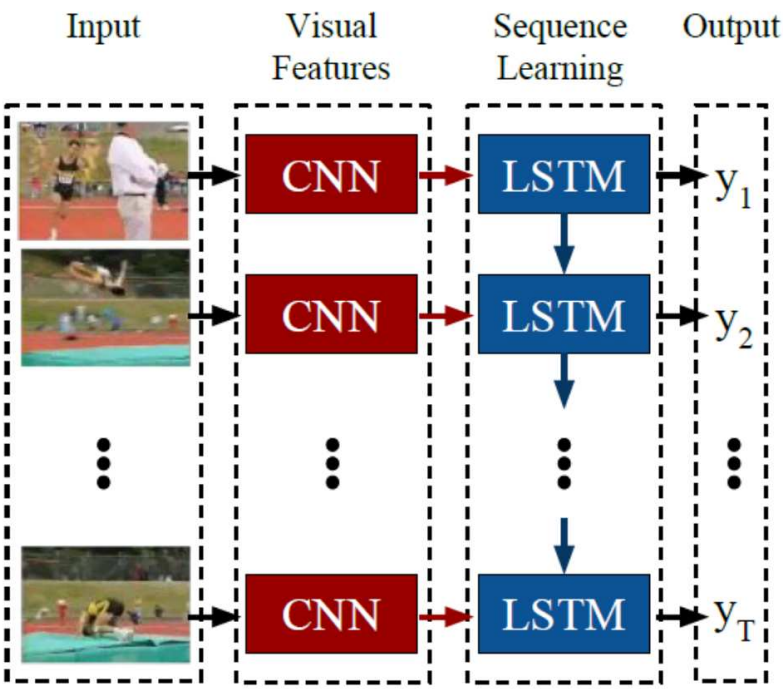


Figure 2.1: The CNN+LSTM architecture. Each frame in the video is processed individually by the CNN and the output feature representations are passed through a single or stacked LSTM (Long Short Term Memory) for sequential modeling. This approach showed an improvement of almost 17% over the previous approach in [1].

Kar *et al.* [35] proposed a novel architecture known as AdaScan that learns to adaptively pool the discriminative key frames from a video and then train a classifier on them to better classify an action within the video. This pooling and classification are done in a single scan of the video. The approach outperformed several action recognition methods on the UCF101 and HMDB51 datasets at the time.

Despite the many successes, the 2D CNN attempts had a major limitation. The temporal information in videos was squashed within the first few 2D convolutional layers. Further attempts at better model designs aimed at preserving the temporal information by making changes to the convolution operations in the architecture.

2.1.2.2 Two Stream networks

Simoyan and Zisserman [12] in 2014, laid the foundations for an important approach that has influenced multiple papers since its publication. Known as the Two Stream network, this novel architecture draws from the two pathways design of the human visual cortex. In the human visual cortex, there is one pathway that relies on processing the spatial information from visual input, called the ventral stream. Another important stream, known as the dorsal stream, processes the motion. Inspired by this, the authors designed the Two Stream network such that one stream processed spatial information, or appearance from still images, while the other processed temporal information or the motion between frames. The authors demonstrate the effectiveness of using dense optical flow to model the motion between consecutive frames by showing an improvement in the classification results between Spatial ConvNet (RGB only) and Temporal ConvNet(RGB + Optical flow) by almost 10% on the UCF101 [20] dataset. Another contribution of this paper is using the idea of multitask learning to improve the overall performance of the network, while increasing the amount of training data. The network is trained on two action datasets, UCF101 [20] and HMDB51 [21] having a separate loss for each dataset and aggregating the individual losses for the network's final training loss. A multitask training using both UCF101 and HMDB51, increased the network's overall evaluation performance on HMDB51 by a staggering 9% more than just training independently on HMDB51 itself.

Ng *et al.* developed an architecture inspired by the Two Stream architecture [13]. Both Ng and Donahue exploit the advantage of LSTMs for sequential modeling, yet, while Donahue used only RGB frames as input, Ng used both stacked RGB in conjunction with optical flow information (as in the Two Stream architecture) as visual inputs and then use LSTM to model sequential data over it. The final predictions of the frames to the video level prediction are done as a linearly weighted sum of the prediction scores of the individual frames, which improved the performance over the Two Stream network by a small margin on UCF101. Despite the successes, the problem of an exhausting number of training parameters and insufficient training videos still persisted.

Feichtenhofer *et al.* [18] presented the SlowFast network, inspired by the Two Stream architecture by Simoyan and Zisserman [12]. The SlowFast network draws on the same idea of spatial and temporal processing separately, as explained in the Two Stream network. The model consists of a Slow pathway operating at a low frame rate to capture the spatial semantics, and a Fast pathway operating at a high frame rate, which captures the motion in the video. This method has two key differences from the two stream architecture. Firstly, the use of variable frame rates for the Slow and Fast pathways as opposed to the fixed frames from both spatial and temporal channels in the Two Stream method. Another significant difference is the use of only raw data as input, compared to using optical flow as motion information in the previous network. This design is partially inspired by the biological human retina, which consists of the Magnocellular (M-cells) and Parvocellular (P-cells). The M-cells focus on the high temporal frequency, and are responsible for fast motion in sight, but not the spatial information. The Fast pathway is comparable to the M-cells. The P-cells, on the other hand, are more responsive to spatial detail and color, but not to motion. The Slow pathway is similar in concept to the P-cells. The SlowFast network boasted state-of-the-art performance in action recognition on the Kinetics400 dataset (79%) [19].

Temporal Segment Networks (TSN) [36] prized as the state of the art in action recognition in 2016. This idea focused on two major problems in CNNs prevalent then. First, to design a video recognition framework that can model a long range temporal structure. The variation of information between consecutive frames in a video is not pronounced. Hence, a dense sampling method for video recognition would result in highly related samples, thereby redundant for the model. To solve this, the authors propose a sparse sampling method, which extracts short snippets uniformly distributed over the video and randomly picks a frame from them. A segmental consensus function takes as input the information from these frames and aggregates them over the whole video. To fully take advantage of the ConvNets, the authors perform an ablation study using different modalities like RGB image, stacked RGB difference, stacked optical flow field, and stacked warped optical flow fields. The frame sampling method proposed by the authors has now become the standard for sampling videos frames in the domain of video action recognition.

The conclusion was that the network outperformed the prevalent state of the art networks on both UCF and HMDB using three modalities together namely RGB, optical flow, and warped optical flow. In 2018, this work was extended with three major additions [37]. First, the handling of untrimmed videos was proposed, as compared to only trimmed videos in the prior work in Temporal Segment Networks. The untrimmed video is divided into short windows of fixed duration, and action recognition is performed on each window independently. A hierarchical aggregating strategy called Multi-Scale Temporal Window Integration (M-TWI) predicts the final results for the untrimmed video. Second, to produce video level prediction results, the authors proposed a top-K pooling or attention weighting on the snippet scores as an aggregation function. Third, the work introduced the concept of cross modality initialization wherein learned representations from one modality like RGB, can be successfully transferred to another modality like optical flow.

2.1.2.3 3D Convolutional Neural Networks

Despite taking stacked multiple images and treating them as different channels, a 2D convolution results in an image. Thus the temporal information is lost at the first convolution step itself. By explicitly adding a third dimension to the convolutional kernels, the model is able to preserve the temporal information without loss. Tran *et al.* with 3D Convolutional Networks (C3D) formulated a new stream to tackle the problem of understanding spatio-temporal data in videos [38]. The authors proposed 3D convolution, where convolution and pooling are done spatio-temporally to preserve the temporal information. Instead of a (k x k) kernel for convolution, it is changed to (d x k x k) where d is the temporal depth of the kernel. Since the network is cumbersome to train on large video datasets, the model performance is evaluated on UCF101 [20]. A linear classifier (SVM) is applied on the compact features learned from the C3D, and achieves pretty high accuracy on UCF101. Using only ten dimensions to achieve a high baseline result (54%), the authors demonstrate the effectiveness of the information captured by C3D in the videos. The disadvantage of this method, however, is that it is computationally very expensive to train and resource heavy. The C3D model uses a VGG-like architecture that resulted in a massive 79 M trainable parameters.

2.1.2.4 Inflated 3D CNN

Carreira and Zisserman proposed the I3D model in 2017, which is the state of the art in action recognition [14]. The motivation for this idea is twofold. First, the reuse of existing 2D CNN models developed for image classification pre-trained on ImageNet for 3D convolutions. For this, the authors use Inception-v1 pre-trained on ImageNet, the state of the art 2D CNN model for image classification, and inflate its filters and pooling kernels from 2D into 3D. This led to very deep spatio temporal classifiers. In addition to the RGB stream, the Two Stream I3D model takes input from the optical flow stream. The authors hypothesized that this model performs better since it can capture fine-grained temporal structure of actions. Secondly, the authors validated the importance of pre-training models by pre-training their Inflated 3D ConvNet model on the Kinetics400 dataset and finetuning it to the respective dataset (UCF101 [20] or HMDB51 [21]). The resulting pre-trained I3D model outperformed the previous existing state of the art models by a significant margin on the two datasets under consideration. A Two Stream I3D pre-trained with ImageNet and Kinetics dataset is able to achieve 98% accuracy on UCF101 dataset.

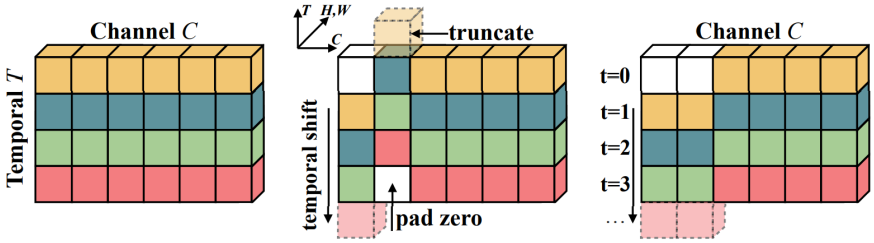
In the Action Transformer model, Girdhar *et al.* proposed to aggregate features around the person of interest to explore the action of a person based on their interactions with other people and objects around them in a scene [39]. An action in the current frame also requires a context of events that have happened in previous scenes, to establish a context of past events for better learning. The Action Transformer network aims to utilize this contextual information in classifying an action of a person. Building upon the I3D model for spatio-temporal features and a region proposal network for person localization, these features serve as input to the transformer model. As noted by the authors, this network focused on the hand and face regions for cues, as these serve to be the most salient features for discriminating an action. This architecture is a little different from the other architectures described above. The network solved the problem of detection of humans in the video as well as classifying the actions performed by the individual actors. This problem sub-domain is known as action detection and localization.

2.1.2.5 Improved 3D CNN

3D CNNs are efficient, yet they are computationally expensive. The future attempts at designing architectures for video action recognition aimed at preserving the efficiency of 3D CNNs and reducing the number of computations at the same time. Since this network architecture is inspired by 3D CNN networks, it is studied in this subtype of action recognition architectures. The motivation behind the work by Xie *et al.* [15] is to develop an architecture that is efficient much like the 3D CNNs, yet has the complexity of 2D CNNs. This task is achieved by replacing certain 3D convolutions with 2D convolutions, to make the model more computationally efficient, while retaining the accuracy. Taking the I3D architecture [14] by Carreira and Zisserman (discussed in 2.1.2.4), the approach explores retaining the 3D temporal convolutions at bottom most layer (called as Bottom-Heavy I3D) or the top most layer (called the Top-Heavy I3D) of the I3D network, and having 2D convolutions for the rest of the layers. The authors find that the Top-Heavy I3D is better performing, which leads to an interesting insight that the temporal information is more relevant for the upper layers with higher spatial information abstraction. Another contribution is reformulating the 3D convolution module as a separable 3D CNN module, defined as S3D, which performs better than the I3D using only RGB frames, while enjoying less computational cost. This model also performs better than I3D on the Kinetics [19], UCF [20], HMDB [21] as well as Something Something [40] datasets using RGB frames as input.

The advantages of dense optical flow as an important modality have been leveraged by the video action recognition models to add local motion information between consecutive frames [12,13,18]. The computation of optical flow, however, is very expensive and thus limits the model's ability to be deployed in real time during inference. Thus, efforts made by Stroud *et al.* [41] in Distilled 3D Networks (D3D) aimed at incorporating the optical flow computation within the C3D model so as to eliminate the requirement of computing the optical flow signal offline. The authors propose a Student Teacher network that distills the knowledge of the optical flow computation into an S3D model [15]. The resulting D3D model is able to achieve performance at par with the models that use the additional optical flow signal, but only using the RGB modality as input signal.

Lin *et al.* discuss a similar question of revisiting the problem of 3D CNNs as in S3D [15]. The authors proposed the Temporal Shift Module (TSM), which shifts the channels along the temporal dimension [16]. A video A is represented as $A \in R^{N \times C \times T \times H \times W}$ where N is the batch size, C is the number of channels, T is the number of frames or the temporal channel, H and W are the height and width of the frame. The channel shift is done, such that the temporal information at any time instance T_i has the information from T_{i-1} and T_{i+1}. Figure 2.2 illustrates the core idea of TSM. The authors hypothesize that this mingling of neighboring frames is very important for complex temporal modeling. The rest of the convolution is similar to a 2D CNN. This method can handle trimmed (offline) as well as untrimmed (online) videos, with a simple shift in the frames considered at any time instance. This method won the first place in the Something Something challenge for action recognition, while achieving very low processing latency.



(a) The original tensor without shift. **(b)** Offline temporal shift (bi-direction). **(c)** Online temporal shift (uni-direction).

Figure 2.2: Illustration of the Temporal Shift Module (TSM). At any given instance T_i, the network is aware of channel information from the past frame T_{i-1} as well as the future frame T_{i+1}. In an online setting where future frames are not available, the network only relies on the past frames for sequential information.

This dissertation takes two networks to establish baseline experiments for the main idea: SlowFast and TSM. Trained on the Something Something dataset, the models are run on the validation data in inference mode and the patterns of misclassifications are analysed.

Jiang *et al.* design the Spatio Temporal Module (STM) with a goal to incorporate temporal information in the 2D CNNs [17]. The authors propose two modules that form the STM module, namely Channel-wise Spatio-Temporal Module (CSTM) and Channel-wise Motion Module

(CMM) to efficiently encode motion features. The STM module is introduced as a replacement to the residual blocks in ResNet, thereby creating a computationally lighter model for action classification in videos.

The models discussed in vanilla action recognition rely on using the complete frame to learn about actions. For actions involving a focused interaction between the actor and the objects in the scene, using the complete frame information poses as a shortcoming. The next section delves into the challenges involving human-object interactions in actions, and how different models attempt to solve it.

2.2 Human Object Interactions in videos

Classifying videos involving Human Object interaction (HOI) is challenging as it involves analysing the action the actor is performing and detect the object in the video to model the inter-relationships between the human and the object. Figure 2.3 displays the complex dynamics of human-object interaction. A lot of insight is obtained about the object by knowing how an actor interacts with it, *i.e.* object properties like hardness, softness, elasticity, etc.. Affordances, or the different behaviors an object allowing the execution of also give information about the object, and thereby how an action is performed. There is a temporal dependency or a cause and effect in how one or more objects in a scene interact with each other as the video progresses. This section discusses the different attempts at understanding Human Object Interaction in videos using these insights.

Escorcia *et al.* discussed the dynamic nature of human-object interactions over time [42]. This method focused on actions where the human and the interacted object might be the same, yet the evolution of the action over time is different. To explain this better, the authors give an example of two actions, namely *answering the phone* and *dialing a phone*. They consider features like relative location, relative object sizes, and relative overlap between the detected human and objects over several time intervals. These features are then aggregated into descriptors and trained using a linear SVM for a video level class prediction label.

Figure 2.3: A video from Something Something of moving a mouse near a pen. Human Object interactions have inherent correlations between the subject, object, other objects as well as scene semantics.

Neverova *et al.* focused on understanding the relationships between objects in a human-object interaction video. The aim of this paper was to study the manner, order and effect of the object interactions of the human [43]. Humans have the faculty known as reasoning that can deduce the sequential dependency of actions on objects before reaching a final state. Using the example of cutting a carrot placed on a cutting board, the authors explained the temporal interactions between the human, carrots, and the knife (the carrot was chopped by the human). By considering time as the explicit causal signal, this paper studied the object level reasoning in videos. The goal was to identify the causal event A before B, not from the pixels in the video frames, but from the object level perspective. The state of the art models in object detection can help us detect the objects present in a video frame. Despite this information, determining their semantic sense of interactions between actors and objects in a scene is a more challenging task. The paper proposed Object Relation Network (ORN) to reason between objects in space and time. The ORN took as input the object detection masks over different object categories and temporal occurrences and modeled relationships between detected objects through the video.

Human actions seen across available action recognition datasets belong to a variety of coarse action groups such as sports actions, daily actions, cooking actions, etc. Ji *et al.* exploited the context information from videos to design a coarse to fine approach to classify human actions [44]. The authors proposed a context knowledge map to classify an action into a coarse grained group. The coarse grained classifier took as input all informative sources like scene, objects, pose estimation, visual data, and optical flow features from the video and explores the relations between them. The context knowledge map summarized this information and modeled their relationships to form

a coarse grained action group, rather than fusing all information directly. Utilizing the features from the coarse grain group thereafter, a fine grained classifier realized the accurate action in the video.

One of the foremost limitations of the current human-object interaction recognition models is that they consider the entire scene for the feature representation, and hence fail to recognize and model the specificity of the objects and relations with the actor. Materzynska *et al.* addressed the compositionality of actions by looking into the dynamics of the objects in consideration and the subjects interacting with them [40]. The novel recognition method, known as Spatio-Temporal Interaction Networks (STIN) relied on the state of the art object detectors for its analysis. A spatial interaction module establishes the reasoning between the objects in the video, and a temporal reasoning module encodes the relations and transformations between the objects and the human. For activities without a prominent object-actor interaction, the model relied on scene-level spatio-temporal feature representations for predictions. To understand the composability of the model, the combinations of verbs and nouns in the training set do not occur in the test set.

Another attempt to tackle the local information processing for efficient human-object interaction is proposed by Martinez *et al.* in [45]. The authors proposed altering the last few layers of an existing network to extract local object information. In addition to the global frame information, the network had discriminative spatial filter banks that corresponding to each action in the dataset. Visualizing the filters with the highest response in the video frame for each action, the authors demonstrated the increase in the model performance as it discerns the finer details of the specific objects correlating with certain actions.

2.3 Natural Language embeddings

Embedding is a commonly used term in the domain of statistical modeling and machine learning. It is a representation of an abstract concept encoded into a real valued vector. With regard to Computer Vision, an embedding can be a feature representation of an image or a video. Similarly, in the domain of Natural Language Processing, an embedding can be a representation of a word or

a sentence. An efficient embedding technique is able to bring closer two or more concepts that are semantically relevant or similar. Mathematical operations can be performed on the embeddings to show the relation between those concepts. For example, the cosine similarity of representations between two similar entities will be closer to each other in a high dimensional space than two dis-similar entities. In practice, the embeddings are extracted from the penultimate layer *i.e.* the network layer before the final classification layer. For example, an image embedding from Resnet-50 [9] is of 2048 dimensions, while the embedding from BERT [2] is 768 dimensions.

In this section, the common embedding techniques used in domain of Natural Language Processing are reviewed. The two main types of embeddings discussed are Word Embeddings and Sentence Embeddings. While word embeddings have complexities like context change for the same word in a sentence, like *The **bank** robber hid in a village by the **bank** of the river*, sentence embeddings have a challenge of encapsulating the semantic meaning of all words into one fixed length vector. The evolution of techniques tackling these challenges will be discussed.

2.3.1 Word embeddings

Word embedding involves the process of mathematical projection of a word into a real valued feature space. Using methods like neural networks, co-occurrence matrices, probabilistic models etc, words or phrases from a vocabulary are mapped onto a continuous vector space of a fixed dimension size.

2.3.1.1 One-hot encoding

The earliest approaches to word embeddings assigned a unique one-hot vector of finite dimensions to a word. Based on the number of words in the vocabulary, every word was expressed with a 1 in place of the word as in occurred in the vocabulary, and 0 for all other words. Suppose there are 4 words: 'red', 'green', 'blue', 'yellow'. The one hot encoding for each word will be as follows: red: [1,0,0,0], green: [0,1,0,0], blue: [0,0,1,0], yellow: [0,0,0,1]. Despite kickstarting the domain of distributional semantics, this method had several limitations in its use:

1. **Similarity issue**

   Every word in the vocabulary is mapped onto a unique one hot encoded vector. For example, a cat and a tiger are two similar concepts. However, a one hot representation of either removes any similarity one concept to another.

2. **Vocabulary size**

   Often the words or phrases in the vocabulary range in thousands depending on the language modeling application. The size of the one hot vector is equal to the number of words belonging to the vocabulary. Thus, for a word in a vocabulary of 10,000 words, every word results in a very sparse one hot vector of dimensions 10,000. A much larger input vector space than the training data size limits the ability to build reasonably generalizable language models.

3. **Computational issue**

   Such a large sparse feature dimension can limit the computational memory and storage capacity of machines processing them. While there are algorithms that process sparse vectors, the neural networks commonly used for language modeling are limited in the ability due to high dimension input data.

2.3.1.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is an important numerical representation of text used in information retrieval. It is also used as a document retrieval in different search engines. Proposed by Salton and McGill [46] in 1986, this method weights a particular word in the vocabulary to the frequency of appearance in the document and the relative importance with respect to other words. For stop words like *the, at, is*, the tf-idf vector is very low. The similarity between two documents is computed using the cosine similarity of the two corresponding tf-idf vectors. Larger the cosine similarity, closer is the target document to the existing document. As seen from the term, TF-IDF is a combination of two concepts: Term Frequency and Inverse Document Frequency [47]

Term Frequency: Term Frequency corresponds to the logarithm of the raw count of a word in a

document. Mathematically, it is expressed as:

$$tf(t,d) = log(1 + freq(t,d)) \tag{2.1}$$

Inverse Document Frequency

Inverse Document Frequency measures how common or rare a word is in the document. Rarer the word higher is the IDF, while if the word is more common then its IDF will be closer to zero. The word in consideration, the current document and the total document set are considered for calculating the IDF score. It is calculated as the logarithm of the total number of documents in the document over the number of documents that contain the word in consideration. Mathematically, it is expressed as:

$$idf(t,D) = log(\frac{N}{count(d \in D : t \in d)}) \tag{2.2}$$

Rarer the word, the idf score will be closer to 1, while for the most commonly occurring words, the idf score will be 0. The TF-IDF score is then the product of the tf and the idf scores for the word in a document. It is expressed in the eqution 2.3.

$$tfidf(t,D) = tf(t,d) \cdot idf(t,D) \tag{2.3}$$

Higher the tf-idf score of a word, more relevant it is in the document.

2.3.1.3 Word2Vec

Word2Vec is an embedding technique proposed by Mikolov *et al.* [48] that is used to transform text in a document to a real valued numerical form interpretable by machine learning algorithms. It differs from the TF-IDF and One hot embedding techniques as this approach involves the use of a neural network to compute text to embeddings. The neural word embeddings are obtained from training words against other words that neighbor them in the input corpus. The Word2Vec approach is highly efficient at modeling the relations between words that are semantically close to each other. Also, vector operations can be performed on these embeddings of words. Considering

the example of four words: *Man, King, Woman, Queen*. If we take the difference of the vector representations of *King* and *Man* and add the representation of *Woman*, the output representation obtained is that of *Queen*. The cosine similarity of pairs of representations for Man and King and Woman and Queen are also very high, owing to the high conceptual similarity between the pairs of words. Due to the high efficiency of the Word2Vec technique, it is used in many applications including recommendation systems, text data mining, machine translations, etc.

2.3.1.4 Glove

Global Vectors (GloVe), proposed by Pennington *et al.* [49], is another learning based word embedding approach similar to Word2Vec. Compared to Word2Vec that uses a neural network for representation of a word, GloVe learns by constructing a global co-occurrence matrix by estimating the probability a given word will co-occur with other words. Given the vast number of words in the corpus, the co-occurrence matrix is also a huge matrix of dimension (number of words X context). The context is the number of column in the matrix, and each row shows the count of the word in the given context. Owing to the combinatorial nature of the words pairings that can establish context, the number of columns is large, and dependent on the window size chosen for the approach. The co-occurrence matrix is then factorized to achieve a lower dimension representation and high variance within the data using dimensionality reducing techniques PCA [50].

2.3.1.5 BERT

BERT (Bidirectional Encoder Representations from Transformers), designed by Devlin *et al.* [2], is the most recent architecture used in a variety of language modeling tasks like Question Answers [51], Text Summarization [52], Machine Translation [53], etc. It is a context based language representation. The context of a particular word in the sentence is obtained by parsing the sequence in both directions left-right as well as right-left. The sequential model used to establish the textual context is a Transformer [54].

A Transformer, on a very high level of technical detail, works on the principle of multiple stacked self attention modules. The self attention module applies attention mechanism to every word in the sentence or a group of sentences to decide the weight of a particular word.

The main difference between the Word2Vec, GloVe and BERT approach is that Word2Vec and GloVe construct vectors for words independent of context. A word found in the corpus will have the same representation independent of its meaning in the given context. For example, in the sentence *He went to the prison cell with his cell phone to extract blood cell samples from inmates*, the word *cell* will have the same Glove and Word2Vec representation even though it has different meanings in the context. However, BERT takes into consideration the context of the word and thus each *cell* will have a different vector representation.

Another difference is that both Glove and Word2Vec approches cannot handle words that are Out Of Vocabulary (OOV), as they are trained on all the words only found within the corpus. BERT works on the sub-word approach, so it can be used to get a representation for words not found in the corpus as well. Section 3.1 discuss the working of BERT in a greater detail.

2.3.2 Sentence embeddings

BERT model is very efficient at finding the embeddings for atomic words in a sentence. Finding encodings for a contiguous sentence, however, is a different task. One approach could be to find the mean or sum of individual encodings of every word in a sentence. By doing this, the semantically most important words are lost as the unimportant words also contribute equally in the summation operation. Another approach is to take the [CLS] token embedding from BERT model as a representation for the sentence. This technique has been found to have a significantly lower embedding quality than averaging the GloVe word vector embeddings [26]. Attempts to find an efficient sentence embedding technique have been made since 2015.

2.3.2.1 SentenceBERT

SentenceBERT or SBERT derives fixed length embedding vectors for sentences in a text [26]. A similarity measure like cosine similarity can be used to find the semantic alignment between two

sentence level encodings. The proposed network relies on computing a mean of the 768 dimension BERT embeddings for each word in the sentence. It differs from the process discussed above in that the SBERT makes use of a siamese and triplet network to derive meaningful sentence level embeddings. This is achieved by training the model on datasets that contain pairs of sentences and a relative similarity score between the two. This joint training helps the model associate the encodings of the sentences together. For the preliminary experiments, this research makes use of the SBERT model to examine the semantic relevance of different labels in the video dataset.

2.4 Reassessment of labels

Different domains of Computer Vision have relied on certain datasets to highlight the landmark achievements of networks that can solve the problem for which the datasets were designed for. In the image domain, MNIST [55] was one of the foremost datsets, followed by CIFAR [56] and relavtively recent and most popular ImageNet dataset [23].

Numerous attempts going back over a decade starting with AlexNet [6] have continously displayed a steady increase in the performance in image classification on ImageNet. While a systematic gain in performance is reported, the meaningful generalizability of these models is largely an unanswered and underexplored question. The source of this gap in research can be traced back to the dataset and its inherent flaws that seep into the networks. It is now being found that the generalizability of recent state of the art image classification networks has been significantly weaker and in some cases, even impacting negatively.

Beyer *et al.* in [24] discuss the shortcomings in the validation set of ImageNet and collect a new label set that addresses these shortcomings. The dataset contains 1000 categories of real world objects. Most of the images in that belong to those classes have a single identifiable objects that maps to the single label. However, many images contain multiple objects that are also present in the class set. The existence of only one single label in such cases can lead to a bias resulting in inaccuracies. Similarly, there is an ambiguity between identical group of classes within the dataset. Thus, even when an image classifier correctly identifies one of the many objects in an image, it can still be

penalized if the predicted label does not match the intended label. A new annotation procedure as proposed by the authors aims to mitigate this label noise and get an empirical understanding of the generalizability of image classifiers.

A human-in-the-loop approach using a crowd-sourcing platform is undertaken such that every image is annotated by 5 distinct humans. This accounts for any variability assumed due to error by the annotators. A new evaluation metric called ReaL accuracy from Reassessed Labels is presented by consolidating and analysing the results of reannotations. A ResNet 50 evaluated on ReaL labels saw an increase of 6% on the validation data (76.10% on original ImageNet labels vs 82.94% on the ReaL labels).

Yun *et al.* in [57] take a very similar approach as above to address the label inaccuracies in ImageNet. There are certain key differences between the two approaches. The authors focus on re-labeling the training set compared to validation set in [24].Human annotations were replaced by machine annotations. These machine annotations were obtained by using a state of the art image classifier trained on a much larger dataset like JFT-300M [58] and InstagramNet-1B [59]. While it saves a lot of manual review, this approach is still prone to the biases and inaccuracies of the image classifier and the datasets it is trained on.

Similar to [24], Meding and Buschoff *et al.* in [60], discuss the similarity of model performance for various state of the art image classification models. The authors focus on the influence of inductive bias on the difference in performance of the networks. Inductive bias is defined as the set of assumptions and choices available to the decision space before the data is introduced. Or in other words, the factors that contribute to the model initialization before exposing the training data to them. The authors observe that this inductive bias has negligible influence on the variation of performance and the major contribution comes from certain patterns in the available data. These patterns collectively result in the quality of data known as Dichotomous Data Difficulty (DDD).

In DDD, 46.2% images in the data are correctly classified by all models. This type of data is denoted as 'trivial' by the authors. Similarly, 11.5% images are hard for all the models to classify. These are known as 'impossible'. The remaining 42.5% images decide the difference

in the decision boundary of two models. There are some erroneous samples within the data that require cleaning. The authors note that removing this erroneous data is helpful, but does not change the DDD nature of the datasets. Another interesting observation noted is that there is a shared notion of easy and difficult level of images for both humans and CNNs. Meaning, humans with no machine learning experience are able to reliably and consistently predict which images are trivial or impossible for CNNs.

2.5 GradCam visualizations

Convolutional Neural Networks have undoubtedly changed the way problems in machine learning are tackled. Their high efficiency has revolutionized solutions for domains like image classification [6], image captioning [61], visual question answers (VQA) [51], semantic segmentation [62] etc. The success of these networks lies in their ability to model complex feature representations learned from training data. Despite the superior performance, the research community still lacks trust in the interpretable capabilities of these networks. As they continue to remain a blackbox for researchers and developers alike, the failure cases of these networks often leave users wondering for a coherent explanation for the same. Recent attempts in establishing a quantifiable trust in the internal complex representation of CNNs aim to answer this very question *why a network predicts what it predicts*. Thus, it is required that the internal feature representations of CNNs be decomposable and presented in a manner that is intuitive, interpretable and easily comprehensible.

The earliest attempts to investigate visual interpretations of CNN networks was proposed by Zeigler and Fergus [63] and extended by Zhou *et al.* [64] with Class Activation Maps (CAM). The authors illustrated what a CNN learns when it classifies an image into a particular class. Although this was a positive step in peeling the metaphorical CNN onion, there was a striking limitation to this approach. The design involved convolutional layers followed by global average pooling and then a softmax layer. However, this architecture differs from the many modern CNN architectures which have a fully connected layer following the convolutional layers before a softmax layer. The generalizability of CAM was fairly limited as it could not include commonly used CNNs.

To solve this, Selvaraju *et al.* proposed Grad-CAM [25] that accounts for the intermediate fully connected layer in CNN layout and opens up the visual interpretability for almost all common CNN designs. The core idea proposed by the authors relies on two important concepts in neural networks: *activations* obtained in the network forward pass, and *gradients* available in the backward pass. For a particular class label usually the class label corresponding to the input image, the activations and gradients are processed to determine the salient regions in the image that maximize the score of the class label. The resulting salient regions are overlaid onto the input image as heatmaps with focus on pixels with a positive influence on the class score. This dissertation relies heavily on Grad-CAM as a way to examine network learnability as well as getting an insight into their failure cases.

While Grad-CAM paved for a more structured interpretation of CNNs by highlighting salient regions, there were some shortcomings in the approach. The method struggles to localize multiple occurrences of the same class in an image. Additionally, for a given class, the network does not cover the complete area of the class object in the image. To solve this, Aditya and Sarkar *et al.* proposed Grad-CAM++ [30] to extend Grad-CAM visualizations and address the flaws. Most of the idea of the first approach is retained in this one, with an important change. When calculating the weight of the activation map, the authors consider a second order differential of the class score with the activation map. The intuition behind this is to account for the multiple occurrences of the class object in the image which fade away in the first approach.

Further attempts in this direction ask questions rooted deeply in the technical details of the implementation. Fu *et al.* propose XGrad-CAM [31] as a way to address why Grad-CAM uses the average of gradients for weight of activation maps and its effect on network visualization. To achieve better explainability, the authors propose using two axioms that the networks must conform to for more theoretical and reliable visualization. These axioms are sensitivity and conservation. Sensitivity states that the importance of each feature map should be reflected in the class score when it is removed, or replaced by zero values. Conservation ensures that the feature maps are the only dominant contributor of the class score and not any unexplained factors.

The numerous approaches described above focus on obtaining interpretability for a single image. The methods highlight salient regions, or the spatially influential information in the input. For videos however, there is an additional challenge of the temporal information contributing to an action or event. This remains unaccounted for by previous CAM attempts. In [29], Manttari and Broome *et al.* propose an approach to identify the temporal information most meaningful to the network for its classification. Meaning, the proposed idea shows which frames in a video contribute most to the class score. To do this, the a learned mask is applied on the temporal dimension. This mask is either a 'freeze' removing motion data from the video, or a 'reverse' that inverts the sequential progression of frames. When the order of frames is reversed, theoretically, it should drastically change the model prediction. Similarly, if an entire video is frozen to one frame, the score should be affected except for in videos with no observable motion. This helps identify the salient frames sampled from the video. The authors conclude that 3D convolutions focus their temporal attention on short sequences and spatial attention on more contiguous areas. Coincidentally, this work also validates this claim as will be seen in the further chapters.

Chapter 3

Background

This chapter is intended to provide an overview into some of the key concepts utilised in our work. Section 3.1 provides the reader with an introduction to BERT, the current state of the art network used for language modeling. The advantages of BERT are then extended from obtaining word embeddings to obtained sentence level embeddings of labels in the video dataset. Section 3.2 provides an overview of the popular datasets used for video action recognition. In particular, this section gives a detailed description of the Something Something dataset which is the focus of my experiments. Section 3.3 provides an introduction to Grad-CAM, an investigative visualization technique to get an insight into important regions of the image for an action class prediction. The approach generates heatmaps for convolutional layers in the video action recognition models and overlays them on the video frames. This method helps to examine the failure cases of the networks and what regions are focused on while making the wrong predictions. Readers are encouraged to skip to the sections pertaining to their interests.

3.1 BERT

BERT (**B**idirectional **E**ncoder **R**epresentation from **T**ransformers) [2] is a concept presented in the domain of Natural Language Processing by Google in 2018, that has created a huge stir in the field of Machine Learning. It is a context based language representation and the state of the art architecture across various applications like machine translation [53], sentiment classification [65], visual question answers [51] etc. The pre-trained model is available for the community to build on and that helps in two ways: greatly reducing the efforts and resources needed for training such a heavy model and to build highly efficient applications with a simple plug and play with the model.

The key idea in BERT is the bi-directionality to understand language context in a deeper sense. Combining the advantages of a Transformer [54], another state-of-the-art attention model in NLP, the architecture parses text from both left-to-right as well as right-to-left to learn contextual rela-

tions between words in a text. In traditional uni-directional language models, every word is input sequentially as it occurs in the direction of parsing. BERT, however, takes the entire sequence of words at once. Hence, although the name suggests bi-directional, a more relevant description of the approach would be non-directional. A relative importance of a word is decided based on all of its surrounding words.

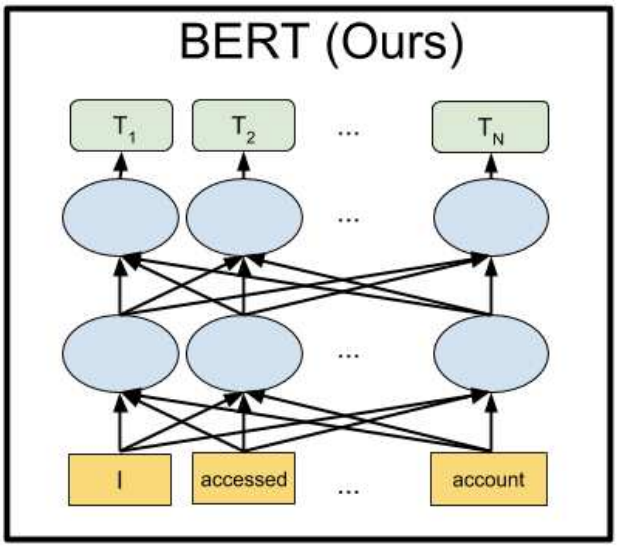


Figure 3.1: The architecture of BERT as proposed by the authors in [2]. The network parses the input sequence of words from both left-right and right-left for efficient context modeling. The output of BERT is a fixed length vector representation of every word based on the context in the input text.

Figure 3.1 shows an illustration of the BERT architecture. The input to BERT is a sequence of tokens. A token is an individual entry in the sentence or sequence. Every token is replaced by its ID in an embedding table. The embedding table is simply a dictionary that has the keys as the words used during training, and the corresponding values as the indices of the words as they occur on the text. Two special tokens are added to the input [CLS] and [SEP]. [CLS] token is inserted at the beginning of the first sentence and [SEP] token is inserted at the end of every sentence. Let's consider the example sentence: *The cat is sitting on the fence*. After word tokenization, our example sentence looks like: ['The', 'cat', 'is', 'sitting', 'on', 'the', 'fence']

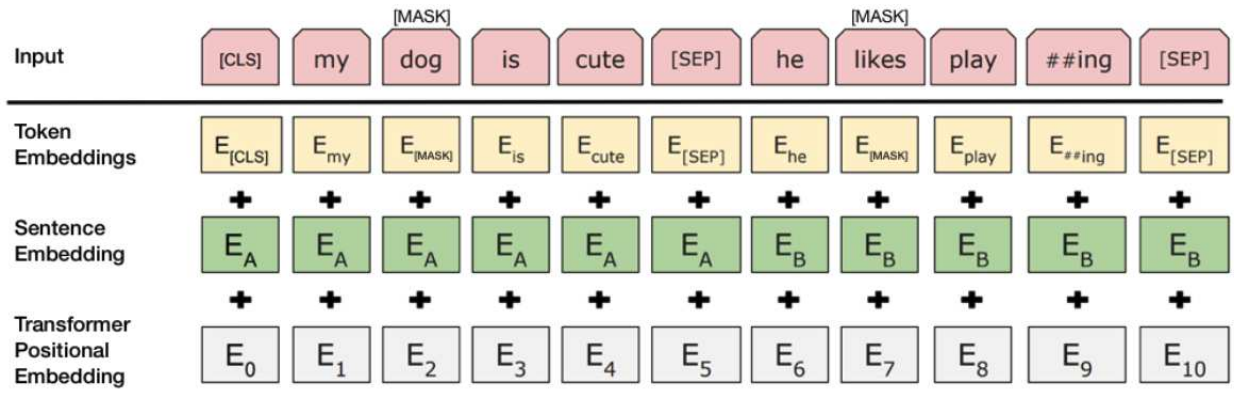


Figure 3.2: The input to the BERT model is a summation of three different embeddings: the token embedding for every word as well as the [CLS] and [SEP] tokens, the sentence embedding indicating the position of the sentence in the input text (0 or 1) and the positional encoding of every word as it occurs in the input text.

Thus the tokenized input to BERT is now a sequence of token IDs of the words along with the token IDs for [CLS] and [SEP] tokens. In addition to the tokens, BERT has two more inputs: the sentence number encoding and the positional encoding of every token in the input sentences. The three encodings (tokenized encoding, sentence encoding and position encoding) are summed together as input to the network. Figure 3.2 illustrates the input to the BERT model. The output of the network is an embedding vector of 768 dimensions for each token. This embedding of tokens can be used for various applications as required. Suppose we want to understand whether a text is spam or not, we take the representation of the [CLS] token which encodes the meaning of all the text in itself, and train a binary classification model to classify the given text into spam or not spam.

Due to its training process, BERT has an advantage over the previous attempts at word embedding and language modeling. Despite their effectiveness, the previous attempts at word embedding (GloVe, Word2Vec) faced a major limitation. These methods were not able to encode words that were not seen in the vocabulary of the training corpus of the networks. This is called the Out Of Vocabulary (OOV) problem. BERT, on the other hand, with a training corpus of 30K words compared to millions required for others, is able to handle the OOV problem. The network splits such words into subwords and characters and uses that information in their encoding. The subwords in

the the tokenization process are assigned the # character. Suppose *playing* is a OOV word, but *play* and *ing* are seen in the training corpus. The embedding for *playing* is thus obtained by splitting it into the sub-words 'play' and '#ing' and then summed for the representation.

3.1.1 SentenceBERT

In order to apply BERT embeddings of words to a sentence to find a sentence level embedding, Reimers and Gurevych proposed SentenceBERT or SBERT [26]. The context of the sentence is lost when embeddings from all words are simply pooled together. Consider two sentences *putting something and something on the table* and *putting something, something and something on the table*. Semantically, the two sentences are very similar. However, the BERT representation of each word will be different, and thus in pooling BERT token embeddings for sentence level, the similarity between the two sentences is lost. Another approach would be to consider the [CLS] token for sentence level mapping. The authors state that this approach does not give good sentence level embeddings either. Thus, to tackle the problem of mapping similarity between like sentences, the authors propose a Siamese network type training.

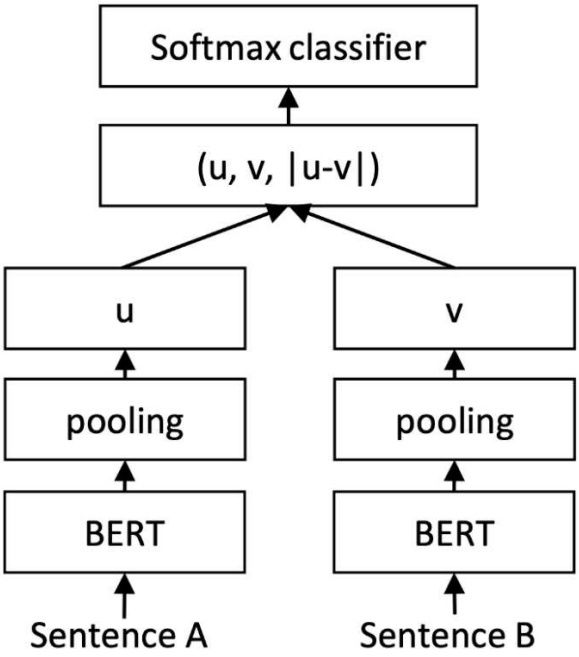


Figure 3.3: The SBERT model consists of the same BERT model replicated multiple times where the weights are tied together. Two sentences A and B are passed through BERT to get their embeddings. Each branch gets its embeddings pooled together to handle different sentence lengths. The objective function of the two pooled sentence embeddings is reduced.

Figure 3.3 shows the SBERT network architecture. It consists of two replicas of the network operating on two sentences together, with the weights tied. Sentence u and Sentence v as shown in the figure are passed through BERT individually to obtain their word embeddings. In order to handle sentences of different lengths, the embeddings thus obtained for the sentences are pooled using a pooling operation. The authors state that Mean pooling gave the best results for sentence level embeddings. The two sentences interact with each other using an objective function given by:

$$o = softmax(W_t(u, v, |u - v|)) \tag{3.1}$$

where W_t is a weight vector of dimensions R^{3n} where n is the dimensionality of the embedding and k is the number of classes in the dataset. Since u,v and $|u - v|$ are concatenated together, the total dimensionality is $3n$. The dataset considered by the authors for their experiments had 3 classes to denote the association of different sentences: *contradiction*, *entailment* and *neutral*. The cross

entropy loss of this objective function is minimized for learning. In inference mode, the sentence is passed through the SBERT network to obtain the sentence level embedding. This dissertation makes use of the sentence level embedding of the action labels present in the Something Something dataset. The input to SBERT is a class label while the output is a fixed length embedding vector.

3.2 Datasets

This section deals with the popular datasets available for designing models for visual understanding. While there is an exhaustive list of datasets available for video action recognition, a small subset that is referenced in a majority of the papers in the literature will be described here. The datasets referred are:

1. HMDB51
2. UCF101
3. Kinetics400
4. Something Something

Table 3.1 presents a summary of these datasets and their descriptions. The Something Something dataset is the focus of this dissertation. The design of the dataset as well as the TSNE visualizations and confusions of the labels will be described in adequate detail.

Table 3.1: Popular datasets used for video action recognition. The Something Something dataset is the focus of this dissertation (last row).

Name	Classes	Samples	Avg video length	Description
HMDB51 [21]	51	6766	2-3 secs	Digitized movies and Youtube videos
UCF101 [20]	101	13320	7 secs	User Uploaded web videos
Kinetics400 [19]	400	306245	10 secs	Youtube videos
Something Something [22]	174	220,847	2-6 secs	Crowd sourced videos

3.2.1 HMDB51

HMDB51 [21] is a popular dataset for action recognition and video understanding. It was designed to overcome the limitation of shortage of action classes and controlled conditions of the actors performing the actions. Taking digitized movies or Youtube as the video source, this dataset contains approximately 7000 videos over 51 action classes. The videos are available over varied distortions like camera motion, viewpoint variation, video quality changes, and occlusion. This dataset was designed to capture the complexity of human actions in videos. There are five main groups of the actions namely: facial actions (like chew, talk), facial actions with object interactions (smoke, eat, drink), general body movements (climb, dive, etc.), body movements with object interactions (brush hair, kick ball, ride bike) and body movements for human interactions (hug, shake hands, punch, etc.).

3.2.2 UCF101

Introduced in 2012, UCF101 [20] still remains one of the most popular datasets for action recognition and video understanding. It was designed to address two key limitations in the video datasets available then, namely the low number of classes and the unrealistically controlled environments and actions performed by the actors. The videos in this dataset are web videos that are user-uploaded and performed in an unconstrained environment. The videos consist of a wide variety of distortions including camera motion, illumination variations, partial occlusion, and low quality frames. There are 13320 videos across 101 action classes. The average length of a video is 7 seconds. The videos are divided into 5 major types namely: human-object interactions, body motion, human-human interaction, playing musical instruments and sports. Comparable to MNIST or CIFAR-10 datasets in the image domain as proof of concept of new research ideas, UCF-101 plays a similar role in the video domain.

3.2.3 Kinetics400

The Kinetics400 dataset [19] was introduced to address the need for a video dataset for action classification large enough for training deep neural networks. While the previous existing datasets, namely HMDB51 and UCF101 were still in use, many limitations in them begged the requirement of a much larger dataset. Both HMBD51 and UCF101 were designed as a solution to the low number of classes and training samples. The Kinetics400 dataset thus formed, contained 400 classes of actions, each containing 400 video samples per class. The video samples were collected from Youtube videos and thus were prone to distortions like camera motions, illumination differences, shadows, background clutter, etc.. As this dataset is specifically designed for action classification and not action localization, the videos are short length with an average of 10 seconds per clip. To make it available as a multi-modal dataset, both audio and visual information are available for the clips. Typically the actions belong to human-object interactions and human-human interactions and belong to one of the three groups, namely, *person action*, *person-person action*, and *person-object action*. While the UCF101 is a good dataset for classification, it has multiple videos of the same person doing the action, hence the participant variation in the dataset is low. Kinetics400 data collection step ensures that each video is unique, and has no repetitions throughout in terms of the actors performing the actions. The Kinetics400 dataset contains 306,245 video clips. To address the issue of multiple labels in a video sample (brushing teeth while dancing), the video will only have a single label, if both labels appear in the action set. Thus, the authors Kay *et al.* recommend a top-5 rather than a top-1 measure for evaluating classification performance for the networks. This idea is interesting as it considers the possibility of multiple actions being performed in a single video. This dissertation draws on similar intuitions of the data for analysis.

3.2.4 Something something

Humans often can reason about complex scenes and interactions based on visual and linguistic information. Modern neural nets, however, are not capable of this depth of reasoning. The labels in the available datasets like UCF101, HMDB51, Kinetics400 describe an action at a higher level

of abstraction than at a finer detail of objects and interaction methods. Studying objects from images does not give information about the change in their pose, or distance from the observer. Video data, on the other hand, can better describe object properties like rigidity, elasticity, softness, stiffness, etc. as the object is interacted with through the length of the video. Video data also helps understand if it can be used in relation to another object. For example, a sharp pointy object can be used to poke holes into another object. Objects and actions are predominantly inter-related, and analysing one independently from the other is not efficient. Words help in connecting the objects and actions to the visual world, and to abstract, everyday concepts.

3.2.4.1 Description

The Something Something dataset is designed to provide textual descriptions of actions performed with objects. The textual descriptions provide insights into the physical properties of objects, as well as their spatial relations or material properties. The average length of the videos is between 2 to 6 seconds. Videos are collected from crowd workers where they were asked to perform an action of the form *Holding something in front of something* and enter the noun phrase and the object before uploading the video. It consists of 220K videos for 174 action labels in the form of textual descriptions such as *pulling [something] from [something]* or *poking [something] with [something]*. Here, *something* serves as a placeholder for objects when the focus is recognition of action in the videos. The labels are known as caption templates. There are a varying degrees of granularity of textual descriptions available for the labels for each video.

Figure 3.4: A video frame of an action in the Something Something dataset labeled *holding something in front of something*. There is a large variety in the granularity of the labels as provided in the dataset.

Figure 3.4 shows a video frame of an action labeled *holding something in front of something*. The placeholders are replaced by the annotations for the objects interacted with in the video. Presented below are some of the degrees of details available in textual descriptions in the labels.

- Action level annotation: Holding [something] in front of [something]
- Object level annotation: Holding a cap in front of a shirt
- Fine grained annotation: Holding a blue plastic cap in front of a man's short sleeve

The labels are grouped into 50 coarse grained classes to simplify the action classification task. Table 3.2 shows an example of the variants of actions involving movement of a camera being grouped into one action group called 'Camera Motions'.

Table 3.2: Multiple class labels are grouped into one action group for simplicity of the problem. Eight class labels involving movement of a camera are grouped into an action group called 'Camera motions'.

Class Labels	Action Group
Moving away from [something] with your camera	Camera Motions
Turning the camera right while filming [something]	
Approaching [something] with your camera	
Turning the camera left while filming [something]	
Turning the camera upwards while filming [something]	
Moving [something] away from the camera	
Moving [something] towards the camera	
Turning the camera downwards while filming [something]	

Compared to the other datasets described above, the Something Something datset is chosen as the main focus of this work due to two compelling reasons. First, the intra-class variation or how different people perform the same action in a human-object interaction environment. Second, the high similarity of different action classes to each other both visually and linguistically.

Most datasets in Computer Vision follow a standardized protocol for collection and curation. This involves procuring images or videos of classes from Youtube or other sources. Next, human labelers are asked to review whether the video corresponds to the class attributed. The assessment checks employed thus ensure the quality of data available to the research community; however, the Something Something data collection deviates from this protocol by crowd-sourcing the videos. The quality control check verify the length and uniqueness of the videos, but are limited only to that. The resultant video will be produced based on how a person interprets a label, and thus open to a large variation from what is expected in the class label.

3.2.4.2 Confusion of classes in videos

Due to the nature of the labels, there is a lot of overlap in the actions based on how the objects are interacted with, e.g. *pushing something slightly so it falls* and *pushing something slightly but it doesn't fall*. This confusion can be seen in the linguistic sense with examples like *holding something in front of something* and *holding something next to something*, and also in the visual sense *uncovering something* and *unfolding something*. This research focuses mainly on trying to understand these ambiguities in the linguistic and visual sense of the classes and investigating the degree to which the networks are able to learn these ambiguities.

3.2.4.3 TSNE visualization of sentence embedding of labels

This work aims to understand the nature of the labels in the linguistic sense. I explore the different ways in which the labels can be visualized to get an insight into the plausible confusions between the model predictions. TSNE [27] visualization is used as a basis to visualize the labels in one singular space. TSNE requires data to be in the vector representation of fixed length. All data points are then projected into a 2D or 3D space for visualization.

Multiple approaches were explored to get a suitable understanding of the vector representations of the labels. First, a sentence is converted into individual words. Each word is converted into a GloVe vector embedding of 300 dimensions. The mean of embeddings of all words in that label is attributed as a final label level representation in 300 dimensions. Figure 3.5 shows the TSNE visualization of this approach

As discussed in section 2.3, this method has a limitation in that every word in the label has an equal contribution to the final representation. Words that are most frequent in the training corpus of GloVe models (Wikipedia corpus, etc), but have a high semantic relevance in the label set (in front of, on top of, next to) tend to get a lower weight in the mean representation. To solve this, an efficient label embedding would map the complete sentence by accounting for the words in context.

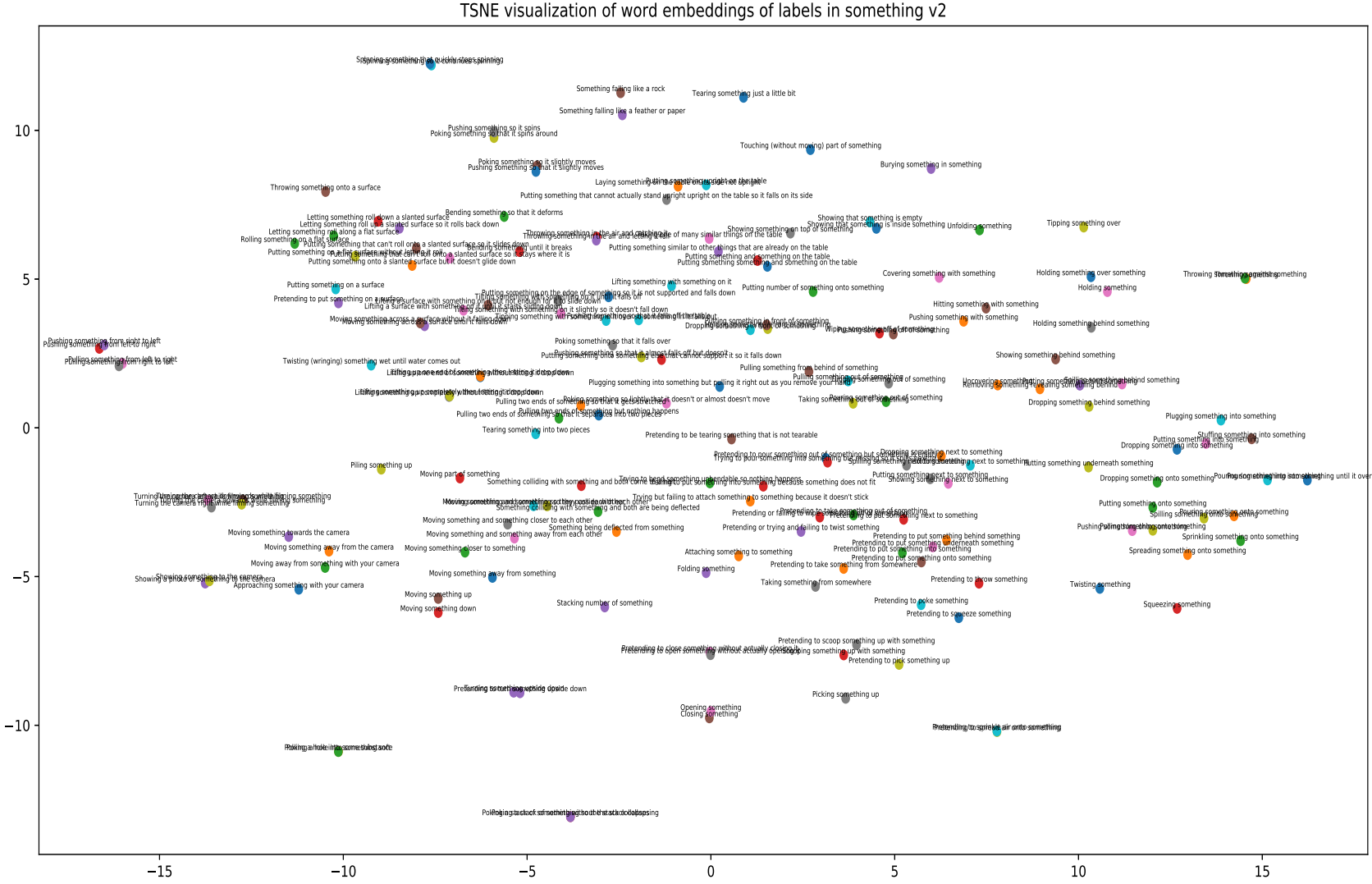


Figure 3.5: TSNE visualization of label embeddings. The GloVe embedding of every word in the label is considered. The final label level embedding is given by the mean of all the words in the label.

The SentenceBERT approach is used for a comprehensive mapping of the labels into the embedding space for our hypothesis. Every label is treated as a sentence in the set and an embedding is obtained for it. SentenceBERT outputs an embedding of 768 dimensions for every label. Every encoding is L2 normalized to have a unit norm for consistency. The stacked representation of the label encodings is a matrix of size 174×768 that is used for TSNE visualization. Figure 3.6 shows the 2D TSNE visualization of the labels.

It can be seen that there are groups of labels that are close to each other in the embedding space. *move something left to right* and *move something right to left* have a very similar representation. That makes sense as the model is not aware of the directionality in the sentence, but focuses on the words which only differ in the order of occurrence. Similarly, the labels *moving something up* and *moving something down* are very close to each other almost overlapping. The efficiency of the SBERT model can be seen in the separability for examples *lifting up one end of something then letting it drop down* and *lifting up one end of something without letting it drop down* as these two labels only differ by one word. The subsequent sections explain how the insight from the visualization is used to understand the degree of confusions the model makes in its predictions. Although this idea is important to understand the possible confusions between labels independent of any action recognition algorithms, it is not the main part of my work.

Comparing mean embeddings and sentence embeddings

A sense of differences of the two embedding approaches can be understood by considering some examples and examining their behavior in the two plots. Let us consider *unfolding something* and *uncovering something*. The mean embedding places the two apart from each other, given the variation in embedding of the root words uncover and unfold. However, the sentence embedding is able to map the semantic relevance between the two labels, and places them close to each other.

Similarly for the labels corresponding to the action group *Camera Motions*, it can be seen that the mean embeddings are placed very close to each other due to the large overlap of different words in those labels. However, the sentence embeddings are able to separate them according to

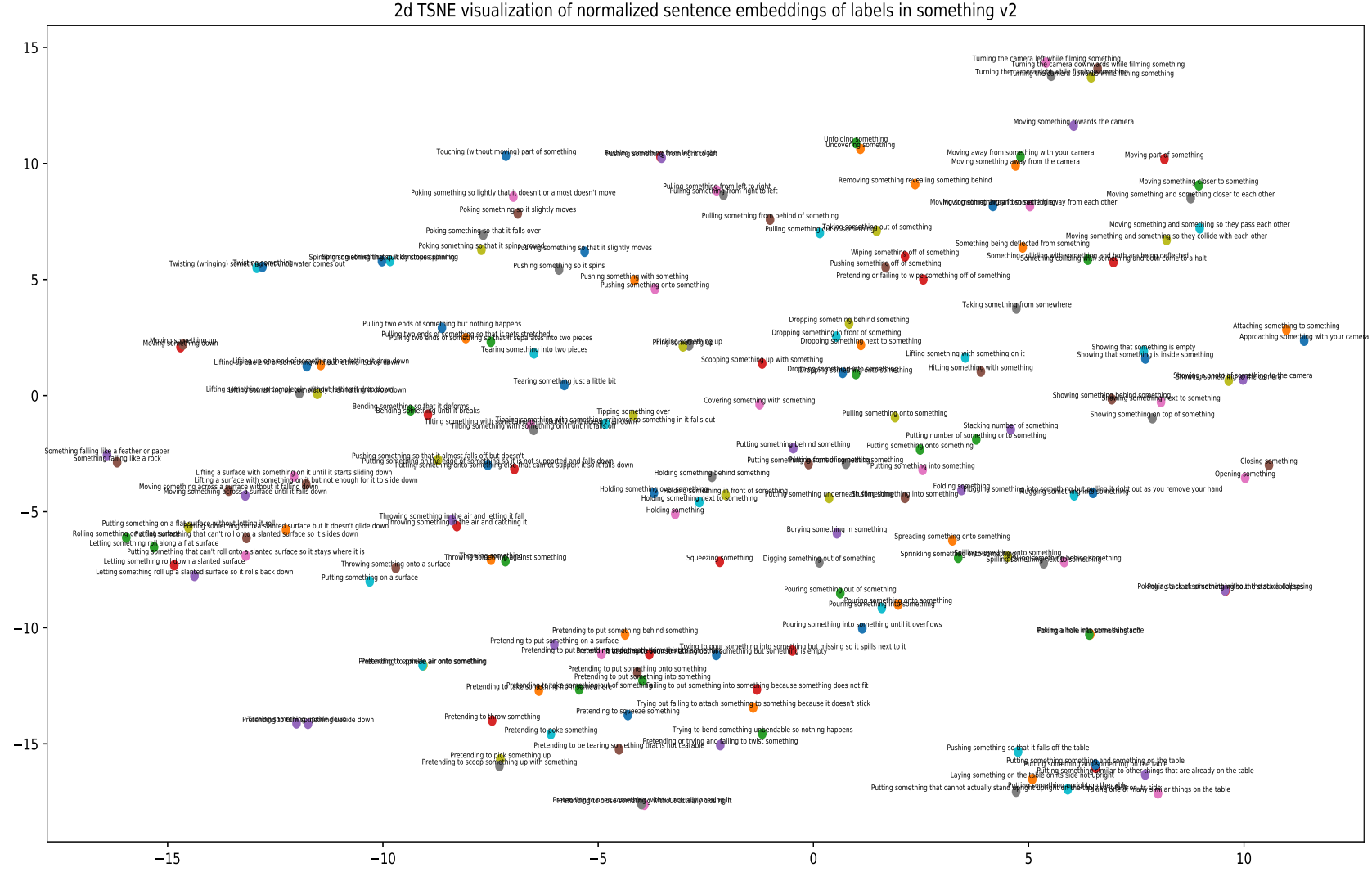


Figure 3.6: TSNE visualization of label embeddings using SBERT. Contrary to the mean embeddings, we take the label level embedding by passing every label in the dataset to SentenceBERT.

the semantic relevance each label has with the label it is most similar to, even if it does not belong to the same action group.

3.2.4.4 Color map of class labels

In addition to the TSNE plots for the sentence embedding, a color map displaying a pairwise equivalence of different class labels with one another is also plotted. Cosine similarity between two class labels is chosen as the measure of equivalence between them. The cosine similarity is expressed in equation 4.3. Figure 3.7 shows the color map of the pairwise equivalence of different labels. The diagonal of the color map represents the cosine similarity of one class with itself. Since it is the class label itself, the cosine similarity is 1.0 and is shown as a bright yellow box across the diagonal. Little clusters of a lighter shade can be seen spread around the diagonal. These classes represent the same action differing only by the preposition. Similarly, when an action is a part of another action, the equivalence score is high too. This can be seen from the pair *showing a photo of something to the camera* and *showing something to the camera* with an equivalence score of 0.9. this equivalence is also noted by the annotators when both labels are presented as video descriptions. There exist lines with a darker shade concentrated in the lower part of the plot. One pair having an equivalence of 0.05 is *pretending to squeeze something* and *taking something from somewhere*.

Surprisingly, there is a very high equivalence score of almost 0.98 between a specific pair of class indices 102 and 120. These classes are *putting number of something onto something* and *putting something, something and something on the table*. The efficiency of the sentence embeddings can be seen from these relations, as it is able to understand the context of the two labels and relate that one is a special case of the other.

Although these visualizations gives us to understand the patterns in the class labels, it only serves as the inspiration for the main idea presented in this thesis. Section 4.3 discusses how to design a video action recognition network that handles multi-output rather than one single class label. The effect of language embedding on model prediction is explained in that section.

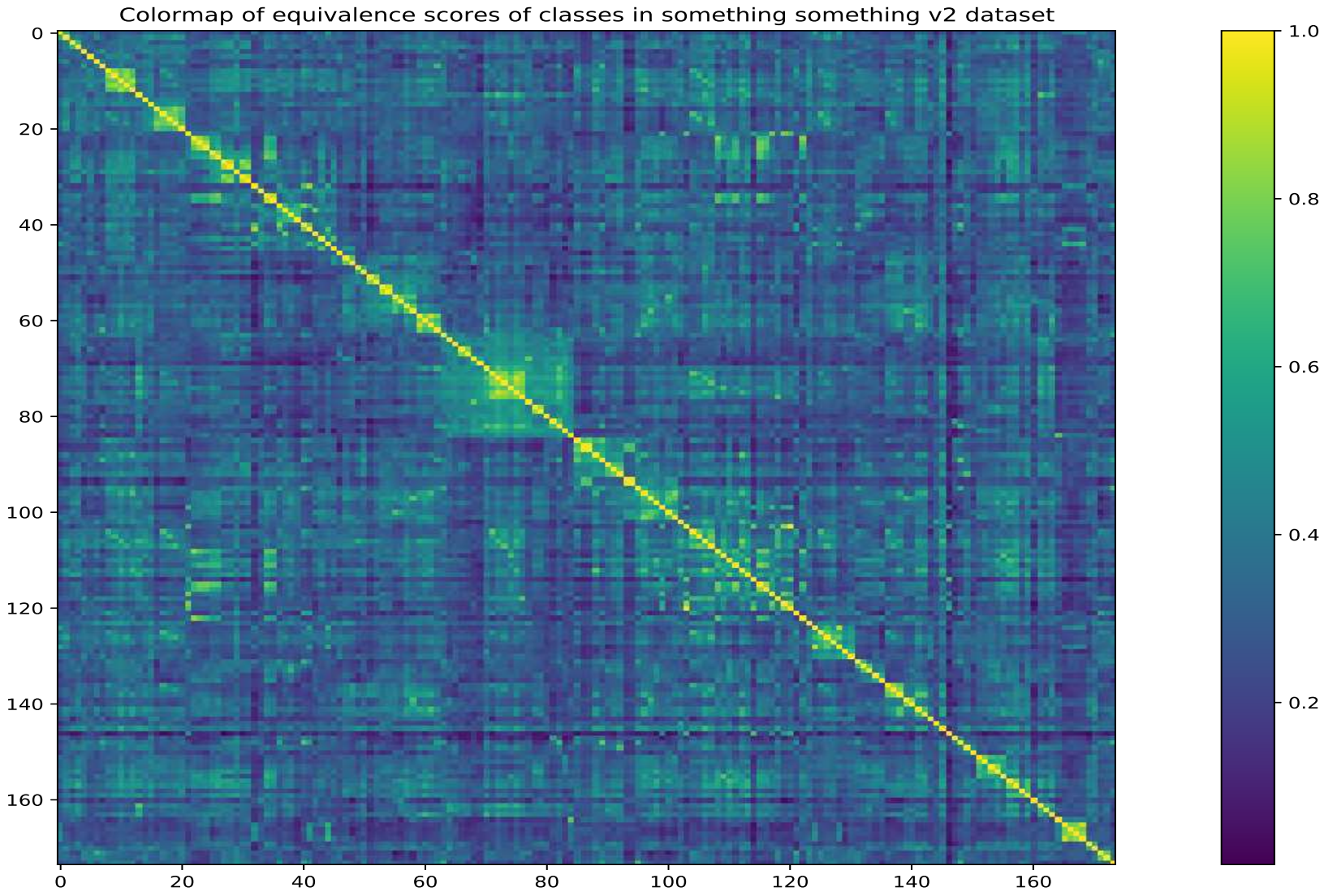


Figure 3.7: Colormap showing the pairwise equivalence of different class labels with one another. We choose cosine similarity as an equivalence measure of the sentence embeddings of class labels.

3.3 Gradient-weighted Class Activation Mapping (Grad-CAM)

Convolutional Neural Networks are highly efficient at many popular Computer Vision tasks like image classification, image captioning, visual question answers (VQA), semantic segmentation etc [6, 51, 61]. The success of these networks lies in their ability to model complex feature representations learned from training data. Despite their superior performance, these networks still remain a blackbox for researchers and developers alike. In events where the network fails catastrophically at a particular task, the users are left wondering for a coherent explanation for its failure. To be able to establish trust in the CNNs as they continue to become an integral part of our lives, we need better methods that explain *why a network predicts what it predicts*. Thus, it is required that the internal feature representations of CNNs be decomposable and presented in a manner that is intuitive, interpretable and easily comprehensible. Selvaraju *et al.* propose Gradient-weighted Class Activation Mapping popularly known as Grad-CAM [25] to highlight and visualize image regions that contribute most towards predicting an action class in a video. Figure 3.8 shows the salient regions that a ResNet 50 model focuses on when it predicts the detected object as *dog*. Consequently, when the network predicts a dog, it focuses on the regions highlighted in red in the image.

Figure 3.8: An illustration of Grad-CAM to visually investigate the internal workings of a common CNN architecture. The kernels in the last convolutional layer are overlaid onto the image as heatmaps to highlight salient regions in the image having a positive influence on the prediction of a class. As seen the classifier predicts a dog in the image, and the respective region around the dog's face is highlighted by Grad-CAM.

This approach extends the work of [64] where the architecture used for visualizations had convolutional layers followed by a global pooling and then a softmax layer. Grad-CAM can generalize to all popular CNNs used for image based analaysis. This method takes into consideration the fully connected layer that follows the last convolutional layer before getting a class specific score using softmax.

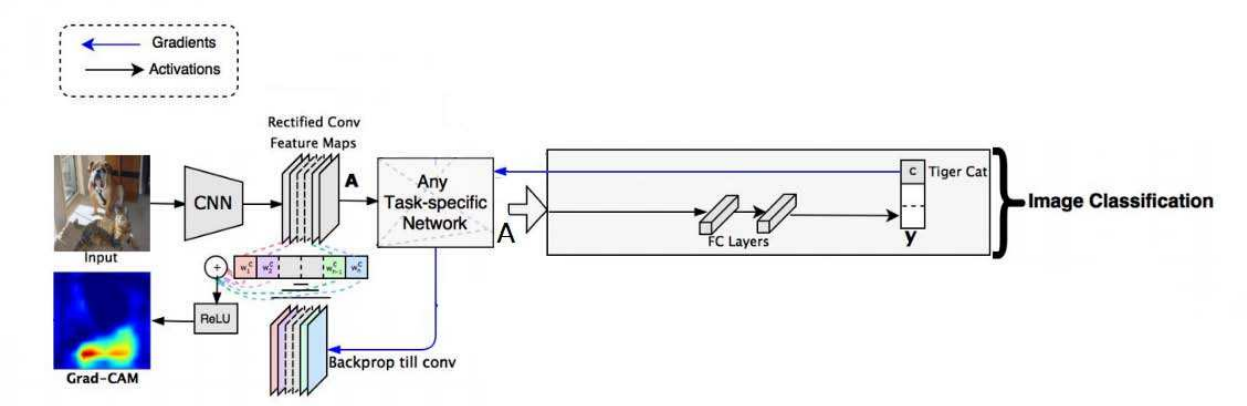


Figure 3.9: Architecture of a common image classification CNN architecture. The gradients are shown as blue arrows while activations are shown as black arrows. The two interact to generate heatmaps that are overlaid onto the source image for visual insights into the network's internal workings.

Figure 3.9 shows an illustration of how an input image propagates through a CNN for image classification and the resultant Grad-CAM visualization obtained for the last convolutional layer. The architecture is referenced from the main paper and edited to highlight only a single task of image classification for ease of comprehension. There are two main components of the neural network training protocol that make this interpretation possible: *Gradients* and *Activations*, shown in the top left corner.

The last convolutional layer has *K* feature maps generated by convolving kernels over the output of the previous layer. Each kernel has dimensions W where H is the height and W is the width of the output. The output from this layer is global average pooled and flattened before passing through a fully connected and softmax layer. If we denote the feature maps as A then $A \in \mathbb{R}^{\mathrm{WxH}}$ and any specific k^{th} feature map is denoted as $A^{\mathrm{k}} \in \mathbb{R}^{\mathrm{W}}$ where $1 \leq k \leq K$. The output of class C from the logit layer is denoted as y_{c}. This value is needed to compute the gradients that flow backwards into the network. The gradient of y_{c} with respect to a particular k^{th} feature map is calculated as:

$$\frac{\partial y_{\mathrm{c}}}{\partial A^{\mathrm{k}}}$$

The gradients also have the dimensions as A^{k} *i.e.* W. The gradient for every element in A^{k} can also be calulated as

$$\frac{\partial y_{\mathrm{c}}}{\partial A^{\mathrm{k}}_{\mathrm{i,j}}} \quad \text{where} \quad 1 \leq i \leq W \quad \text{and} \quad 1 \leq j \leq H$$

In order to calculate which feature maps that have the most contribution in the final output, a score is assigned to every feature map. The score is obtained by performing global average pooling as is given by ${\alpha_{\mathrm{c}}}^{\mathrm{k}}$. ${\alpha_{\mathrm{c}}}^{\mathrm{k}}$ is mathematically expressed as:

$${\alpha_{\mathrm{c}}}^{\mathrm{k}} = \frac{1}{Z} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{\partial y_{\mathrm{c}}}{\partial A^{\mathrm{k}}_{\mathrm{i,j}}} \quad \text{where} \quad Z = WXH \tag{3.2}$$

A positive value of ${\alpha_{\mathrm{c}}}^{\mathrm{k}}$ means that the kernel contributes positively t the decision and the value gives the magnitude of how much. With the score available for each activation map, every feature

map is weighted as a weighted sum ${\alpha_c}^k A^k$ of all K feature maps is given as:

$$S = \sum_{k=1}^{K} {\alpha_c}^k A^k \quad \text{where} \quad S \in WXH \tag{3.3}$$

To see the regions in feature maps that have a positive influence on the decision of predicting class C, all positive signal from S is considered. This is achieved by applying ReLU activation on S. The final heatmap L_c is given as:

$$L_c = ReLU(S)$$

The dimensions of L_c are WxH which is usually 14x14. However, the size of the image fed as input is usually a center crop of size 224x224. We resize L_c to 224x224 and overlay it onto the image. Figure 3.8 shows the visualization obtained by overlaying the heatmap L_c (right side) onto the input image (left side). A region in the image that activates or has a positive influence on the class score y_c is shown in red.

The algorithm and illustration described above is for a single image, and how it is processed by a CNN intended for images. In case of videos, however, there is an additional complexity of modeling temporal dependency between frames. In Chapter 4, section 4.6 it is discussed how image based Grad CAM can be extended to investigate video action recognition models and the insights derived from it. The advantages of Grad-CAM are used to examine the failure cases of networks and if it corroborates with what humans would intuitively look for in that video.

Chapter 4

Methodology

This dissertation examines the effect of ambiguous language in the labels of the Something Something dataset on the generalizability of video action recognition networks. A human-in-the-loop approach is introduced to address the systematic inaccuracies in the validation data. These inaccuracies are investigated by comparing and aggregating the results of two popular video action recognition models evaluated on the target dataset. The networks referenced in this study are SlowFast [18] and TSM (Temporal Shift Module) [16].

The misclassified samples are reannotated and denoted as ReaL or Reassessed Labels. The ReaL accuracy is reported by reevaluating the networks on the reassessed labels. The study reveals that the networks already learn relevant information of the action classes. The multi-person assessment of labels verifies the claim that there are more than one permissible interpretations of videos in the dataset, especially if the video labels have a high linguistic closeness between each other. The source of this confusion stems from the manner in which the dataset was collected. Because of the inherent ambiguities in the samples in the dataset, the video action recognition models suffer when they are penalised if the predicted label does not match the one associated label for the video. The restriction of a single label for a video can negatively impact its ability to generalize to real world data.

The rest of this section describes our techniques in more detail. Section 4.1 gives an introduction into how classical fully-supervised neural networks are trained. Section 4.2 gives an overview of Resnet-50, which serves as the backbone for the two video action recognition networks referenced in our study. Section 4.4 describes the working of the SlowFast architecture and the Temporal Shift Module (TSM) network. Section 4.5 gives an overview of how the misclassifications of the networks are compared and aggregated as well as the conception of the human in the loop approach to reannoate the videos. This section is the main idea of the thesis. This section also explains the ReaL accuracy and effect of reannotations on the model performance. Lastly, Sec-

tion 4.6 illustrates the layer-wise activation maps of the networks using the GradCAM approach. The progression of activations over the layers reveals that the networks indeed focus on expected regions of the video frames to learn an action label.

4.1 Forced Choice training

Neural networks are one of the highly efficient algorithms in the domain of machine learning. The models are able to learn an abstract representation of the data in the feature space that can be used for applications as required. Particularly for a supervised learning task where the labels are available for the data, the models are trained in such a manner that helps reduce the distance between the label corresponding to a sample in the training dataset and the label predicted by the model. This distance between the actual and the predicted label is known as the loss function.

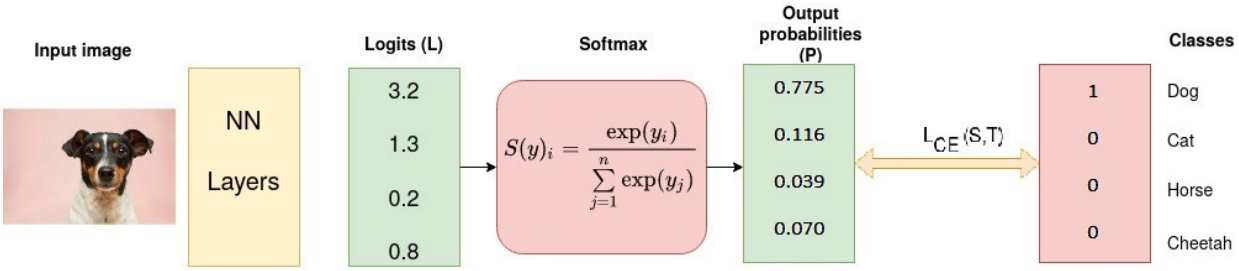


Figure 4.1: Training of neural networks. The label corresponding to the maximum probability in the output probabilities (P) is considered as the predicted class label. This is denoted as Forced Choice approach in this work.

Figure 4.1 illustrates how an image of a dog is classified by the neural networks. The output of the network is a real valued vector called logits. The logits are then passed through a softmax layer that transforms them into a probability distribution (P) for the input sample. The softmax layer is mathematically expressed as:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{4.1}$$

The input image is assigned the index of the entry having the maximum probability (0.775 for the index 0 in the above figure). For most classification problems available in the literature, the popular

loss function used in the training of neural networks is the Cross Entropy Loss. Cross Entropy is defined as a measure between two probability distributions for an event and mathematically expressed as:

$$L_{CE} = -\sum_{i=1}^{n} t_i log(p_i) \tag{4.2}$$

for n classes where t_i is the truth label and p_i is the softmax probability for the i^{th} class. The loss is then minimized using an iterative optimization technique for a successful training of the neural networks. This form of learning will be used in this research for video action recognition and will be denoted as the Forced Choice Learning approach.

4.2 Design of Resnet-50

Resnet [9] has been a very popular CNN architecture in Computer Vision ever since its inception in 2015. The key idea utilized in this network is the residual connection between layers or the output layer. In the design of the residual blocks, the authors used a shortcut connection or skip connection to propagate the information from the previous layer to the current layer, with an aim of improving the flow of information. Figure 4.2 shows the design of the residual block in Resnet. In addition, they are also used to extract multi-level features which have been found effective in a variety of applications like Visual Question Answers (VQA), image classification, medical image analysis etc [9, 51, 66]. This results in the ability to build deep convolutional networks for extract-

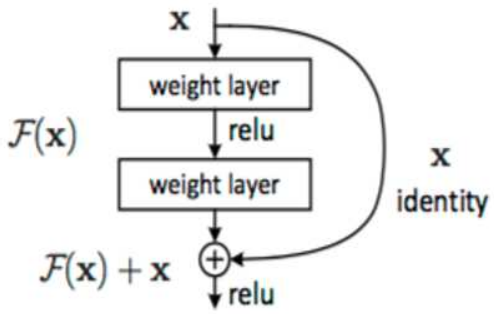


Figure 4.2: Residual block in a Resnet architecture. The skip connection between blocks is used to learn multi-level features from images.

ing spatial features from images. The number of layers in popular variants of Resnet are 18, 36, 50, 101, and 152. A Resnet 50 trained on ImageNet is referenced as the architecture to test an initial hypothesis in this work.

Resnet 50 has 50 layers that are grouped into 4 blocks. Each block has multiple convolutional layers with shortcut/residual connections. The number of kernels between two Resnet blocks are increased by a factor of 2 and the spatial resolution decreased by 2. Considering a convolutional and pooling block as a unit, the number of units distributed across the 4 blocks in Resnet are [3,4,6,3].

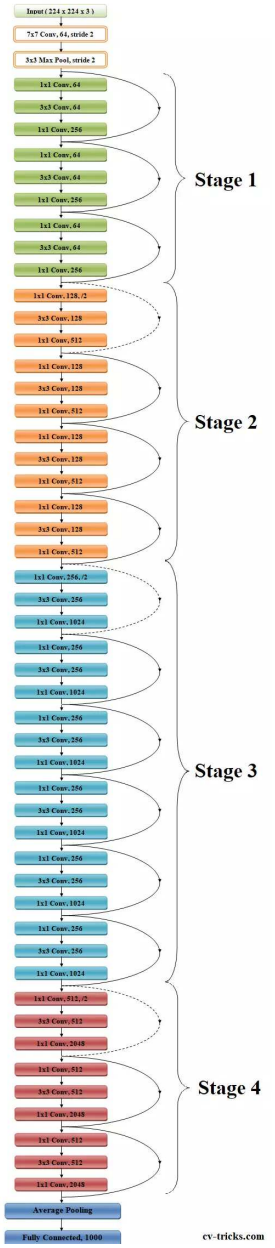


Figure 4.3: The block diagram of a Resnet-50 network. The shortcut connections are between blocks and used to extract multi-level features from images.

Fig 4.3 shows the architecture of Resnet 50. There are 256 kernels in the units of Block 1, 512 kernels in units of Block 2, 1024 kernels in units of Block 3 and 2048 kernels in units of Block 4. The feature representations from the last block are average pooled for a video level representation and passed to a FC layer for classification.

The architecture design described above is used as a backbone in all our experiments. In the initial experiments, we proposed a network that used a 3D Resnet-50 and incorporated language embeddings of the labels in the dataset as obtained from SBERT. This network was known as the Hybrid network. While the approach and its results laid the foundations of the bulk of experiments in this thesis, it was not pursued more in depth. The details of the hybrid network are presented in brief in section 4.3.

4.3 Hybrid network for language embedding

Extending the work by Carreira and Zisserman in I3D [14], the existing 2D CNN architecture is used for spatial feature extraction and convert it into 3D CNN for temporal modeling within the network. The authors inflate a 2D kernel into 3D by replicating the spatial kernels of size (k×k) in the temporal dimension. Similarly, the kernels in 2D Resnet 50 network are inflated to design a 3D Resnet 50 network. The features in the penultimate layer are global average pooled for a video level representation of the input video clip.

At this stage, the network splits into two branches: one branch focuses on the predicting the class index using the forced choice approach. The other branch focuses on predicting the language embedding of the labels where the classes are semantically relevant to the phrases most common in the linguistic sense to the embedding of the target label. Figure 4.4 illustrates the architecture of the Hybrid network.

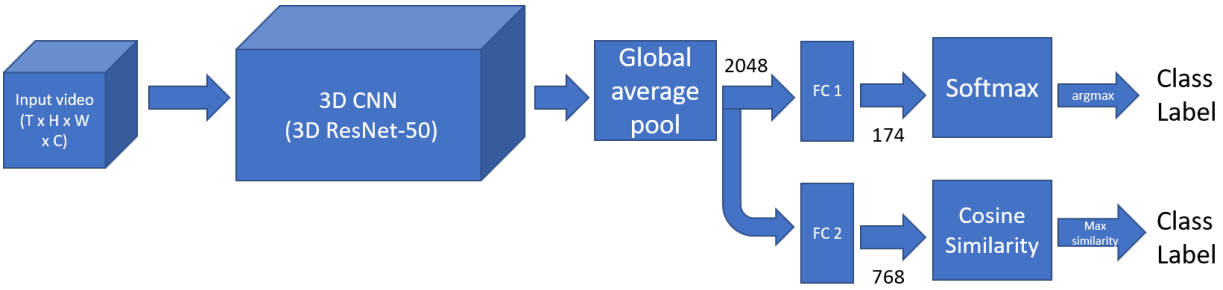


Figure 4.4: Architecture of the hybrid model that combines forced choice and language embedding. For inference, we only look at the output of the forced choice branch, but train for both the branches for more semantic relevance of the predictions.

The input to the model is a sequence of video frames. The output is a vector representation of that video by the model. The model output is divided into two branches. One branch focuses on correctly classifying the action in the video by the forced choice method. The output of this branch has a dimensionality corresponding to the number of classes in the dataset and is known as the logit layer. A softmax layer follows the FC layer, converting the logit layer into a probability distribution vector for the video. The final label assigned to the video corresponds to the index of the highest probability value in the vector. During training, the cross entropy loss between the label predicted by the model and the actual label for the video is minimized.

The other branch focuses on maximizing the cosine similarity between the predicted and the target language embedding for the class label. To do this, there is a slight change in the way the labels are fed to the network. Instead of assigning an integer to every label in the dataset, a 768-dimension vector representation of the labels using SentenceBERT [26] is considered. The language embeddings of the labels thus obtained are L2 normalized to have a norm of 1.0. Thus the output of this FC layer has a 768-dimension vector representation that is similar to SBERT output for the labels. These vectors are also normalized to unit length.

During training, the cosine similarity of the predicted and the target embedding is maximized. The cosine similarity is a measure of how close two vectors are in vector space. Mathematically, it is expressed as:

$$cos(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|} \tag{4.3}$$

The range of the values for the cosine similarity is in the range [-1, 1]. A value of -1 indicates the two vectors are in opposite direction with an angle of 180 between them. Similarly, a value of 1 indicates the two vectors are coinciding with an angle of 0 between them. Our model is trained to maximize the cosine similarity between the target and the predicted language embedding representation *i.e* make the angle between the two vectors 0. The loss corresponding to the cosine similarity is known as the Cosine Embedding loss. Mathematically it is expressed as:

$$loss(x_1, x_2) = 1 - cos(x_1, x_2) \tag{4.4}$$

Maximizing the cosine similarity in turn implies minimizing the cosine embedding loss.

The system is trained to minimize the sum of the cross entropy and cosine embedding loss. In order to have an equal contribution of both the classification loss and the cosine embedding loss, we multiply the cosine embedding loss by 5 so that it matches the range of values of the cross entropy loss. The final loss used for optimization is then given by the sum of the cross entropy and the weighted cosine embedding loss. Thus,

$$L_{hybrid} = L_{cls} + 5 * L_{embedding} \tag{4.5}$$

where L_{cls} is the cross entropy or the classification loss and $L_{embedding}$ is the cosine embedding loss.

4.3.1 Evaluation metrics

During inference, two forms of evaluation metrics are taken into consideration: **Top-1 accuracy** and **Top-5 accuracy**. Top-1 accuracy is the conventional calculation of accuracy. It corresponds to the number of samples correctly predicted by the network over the total number of samples in the validation or testing set.

Top-5 accuracy is a more specific concept of the Top-k accuracy where k is equal to 5. Top-k is described by considering the k top predicted classes by the model in decreasing order of their probabilities. If the correct label belongs to any of the k predictions for the video sample, then the classification is considered correct. For example, if we have 5 samples in the dataset, and the target label for the videos belongs to the k top labels predicted by the model for 3 samples, then our top-k accuracy is 3/5 or 60%.

During evaluation, the Top-1 and Top-5 accuracy are kept consistent as the evaluation metric. However, for the language embedding branch, the cosine similarity of the L2 normalized predicted language embedding is calculated with the L2 normalized representation of every class in the dataset. The classes are arranged in decreasing order of their cosine similarities. The class label having the highest cosine similarity is assigned to the video for calculation of the Top-1 accuracy.

For the Top-5 accuracy, if the correct class label belongs to the top-5 labels predicted by the model (in a descending order of sorted cosine similarities), then the prediction is considered correct.

For this work, the Top-5 accuracy is more important to study as it helps us understand the nature of the mistakes made by the model when it predicts a class label for a video. In all the experiments, the idea is to explore the predictions of the model where the actual label is present in the top 5 predictions of the model, but the first predicted label does not match the actual label. This helps to get an insight into the nature of mistakes made by the model, and ask questions that can understand the data and the labels better.

4.3.2 Results

In this section, we look at the predictions of the forced choice and the hybrid models for a few samples from the validation set and try to understand the nature of mistakes. We choose the top 4 predictions by the networks for analysis. The findings of every sample are tabulated with the Video ID of the sample as the first column followed by the actual label of that video in the dataset. The next four columns in the table explain the top 4 predictions of the model. In each sample, we compare the actual label with the first choice of the models, and then give a gist of the other prediction results.

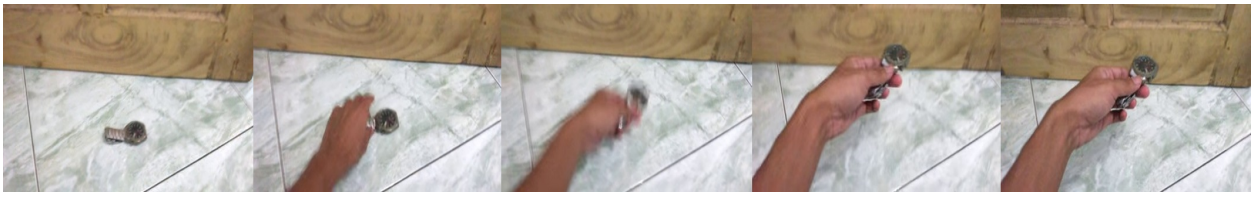


Figure 4.5: A video with the label 'Moving something up'. All three models get the first prediction wrong, but the language embedding and hybrid models have more semantically relevant first predictions of picking up than the forced choice prediction of putting an object underneath.

In Figure 4.5 we see a few of the video frames from the action *moving something up* and in Table 4.1 the predictions of the two approaches. This sample shows how the hybrid model confuses the first prediction with a label(s) that is semantically more relevant to the actual label than the

Table 4.1: Predictions of a video labeled *moving something up* by the two models. The predictions of the hybrid model are semantically closer to the action of moving up than the forced choice predictions.

Video Id	Actual Label	Model Type	Prediction 1	Prediction 2	Prediction 3	Prediction 4
46296	Moving something up	Forced Choice	Putting something underneath something	Moving something up	Pretending to put something underneath something	Lifting something up completely without letting it drop down
	Moving something up	Hybrid Model	Lifting something up completely without letting it drop down	Picking something up	Lifting something with something on it	Holding something

forced choice approach. *moving something up* and *picking something up* are very similar to each other, in both the linguistic as well as the visual sense. Similarly, *lifting something up without letting it drop down* can be viewed as a more expressive form of *moving something up*, but still maintains the semantic relevance to the action. In the forced choice model however, the predicted label is semantically very far away from the actual label, even though the very next prediction is the correct one. It is an interesting question to try and understand the internal details of the model when it predicts an almost disjoint category of class label only from the visual features.

Figure 4.6: A video labeled 'Moving something closer to something'. The language and hybrid model are able to make more semantically relevant first predictions that align with slightly moving an object, but the forced choice model's first prediction is completely irrelevant to the main label.

In Table 4.2 we see that the hybrid model does better at semantically relevant first predictions that the forced choice. As seen from Figure 4.6, *moving something and something closer to each other* is almost overlapping with *moving something closer to something*. Given the inherent over-

Table 4.2: Compared to an irrelevant first prediction of the forced choice model, language models are able to capture the semantic relevance in their first predictions.

Video Id	Actual Label	Model Type	Prediction 1	Prediction 2	Prediction 3	Prediction 4
206534	Moving something closer to something	Forced Choice	Pretending to open something without actually opening it	Moving something and something closer to each other	Moving something closer to something	Pretending to close something without actually closing it
	Moving something closer to something	Hybrid Model	Moving something and something closer to each other	Moving something closer to something	Hitting something with something	Moving something and something so they collide with each other

lap between pairs of labels in the dataset, the language information is also a compelling source of information to successfully disambiguate between actions. The degradation of the model due to lack of this additional information can be seen in the forced choice which is very far from any related activity to *moving something closer to something*.

Although this network is not used in the rest of the experiments, the insights obtained from it laid the foundations for the bulk of my work. Incorporating the language branch does not improve classification accuracy, but the nature of predictions in the top-5 category were more graceful and semantically more relevant to the actual label. This demonstrates that when networks are exposed to the language of the labels rather than just a one hot vector, they can learn to associate more nuanced characteristics of the actions in their predictions.

4.4 Introduction to Video Action Recognition Networks

This section gives an understanding of the working of two video action recognition networks referenced in this dissertation: SlowFast [4.4.1] and Temporal Shift Module (TSM) [4.4.2]. Both the networks rely on a Resnet architecture as their backbone, hence it was covered as background before an overview of the models. The readers can skip to the respective sections pertaining to their interests.

4.4.1 SlowFast

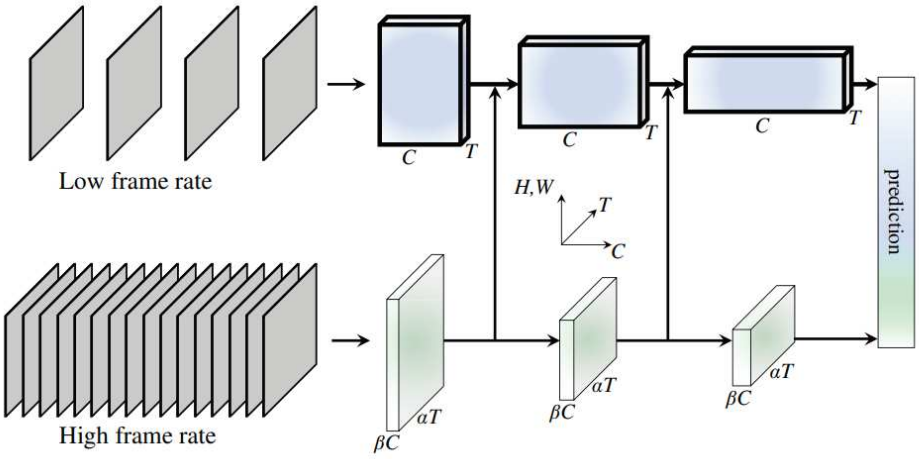


Figure 4.7: Block diagram of SlowFast network. There are two pathways: Slow pathway processes frames at a lower frame rate, focusing on spatial information. The Fast pathway processes frames at a higher frame rate, focusing on the temporal information in a video. The pathways communicate with each other using lateral connections.

In 2019, Feichtenhofer *et al.* proposed the SlowFast network [18] for video action recognition. This architecture was influenced by the design of the retinal ganglion cells in the primate visual system. In those cells, there were 80% Parvocellular (P cells) and 15-20% Magnocellular (M cells). The P cells focused on the spatial stimuli, focusing on colors, textures and lighting while the M cells focused on the motion. Thus, the M cells respond to fast temporal changes and have a high frame refresh rate. The P cells respond slowly to temporal changes, thus having a slower frame refresh rate.

Thus, the authors propose a similar design of the network, mimicking the two cells as individual pathways processing the video frames. The Slow pathway follows the P cells while the Fast pathway follows the M cells. The Slow pathway processes the video at low frame rates focusing on the spatial information in the video frames. Compared to that, the Fast pathway processes video frames at high frame rates focusing on the temporal information or motion. In order for the

two pathways to communicate with each other, the authors propose a lateral connection that fuses information from both pathways. Figure 4.7 shows the architecture of the SlowFast network.

The SlowFast architecture bears resemblance to the Two Stream Network [12] in many ways. The network also had two branches, one that processed spatial information with the video frames and the other modeled the motion in the video. However, the local motion between frames was modeled using optical flow [4]. Compared to that, SlowFast only uses raw frames for processing and experiments with different temporal speeds for efficient spatio-temporal information extraction.

stage	*Slow* pathway	*Fast* pathway	output sizes $T \times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2
res_2	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	*Slow* : 4×56^2 *Fast* : 32×56^2
res_3	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	*Slow* : 4×28^2 *Fast* : 32×28^2
res_4	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	*Slow* : 4×14^2 *Fast* : 32×14^2
res_5	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	*Slow* : 4×7^2 *Fast* : 32×7^2
global average pool, concate, fc			# classes

Figure 4.8: The instantiation of the SlowFast network. The input dimensions to a pathway are shown as (T $\times S^2, C) where T is the temporal dimension or number of frames$,S$\times S$ is the kernel size and C is the number of output channels. Strides are denoted as temporal stride, spatial stride^2. There is no temporal down-sampling in the network until the global average pooling. The number of frames in the Fast pathway is α times that of the Slow pathway, where $\alpha = 8$. The lateral connections used to fuse the Slow and Fast pathway are not shown in the design.

Table 4.8 shows the instantiation of SlowFast network. The input dimensions to a pathway are shown as (T $\times S^2, C) where T is the temporal dimension or number of frames$,S $\times S$ is the kernel size and C is the number of output channels. Strides are denoted as temporal stride, spatial stride[2]. There is no temporal down-sampling in the network until the global average pooling. The number of frames in the Fast pathway is α times that of the Slow pathway, where $\alpha = 8$. The lateral connections used to fuse the Slow and Fast pathway are not shown in the design.

Slow Pathway

The Slow pathway operates on a large temporal stride τ meaning one out of every τ frames in a video is fed as input to the Slow pathway. Thus, for a video with 64 frames and a τ of 16, the Slow pathway processes 1 out of every 16 frames, and thus takes an input of 4 video frames. For generalizability, if the number of frames in the Slow pathway is T, then the total number of frames in the video is $T\tau$.

Fast Pathway

Compared to the Slow pathway, the Fast pathway operates on a smaller temporal stride, sampling more frames than the former. The temporal stride of the Fast pathway is given by τ/α where $\alpha > 1$. Thus for a video with 64 frames and τ of 16 and α of 4, the Fast pathway processes 4 out of every 16 frames, and thus takes 16 video frames as input. Compared to other prominent architectures, there is no temporal downsampling of the input tensor over the layers in the Fast pathway until the final global average pooling layer before final classification. Additionally, because the Fast pathway is not intended for spatial processing, the number of channels in the layers is significantly lesser compared to other networks (or even the Slow pathway). Thus, at every stage the number of channels in the convolution layer are $1/\beta$ times the channels in the Slow pathway. The value of β used in the paper is 8.

Lateral Connection

In order for the two independently processing pathways to be able to share information throughout the network, the authors added lateral connections to fuse the two representations. These connections are added between every stage as seen from the table between *pool1*, *res2*, *res3*, and *res4*.

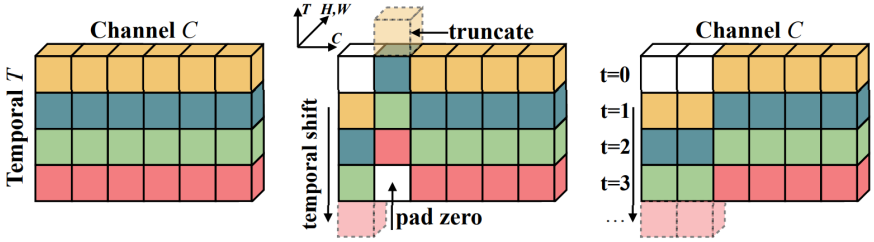
Because the two pathways differ in their temporal dimensions, a transformation is performed to be able to fuse them together. Given the feature shape of Slow pathway as (T, S^2, C) and Fast pathway as (αT, S^2, βC), where *T* is the temporal dimension, *S* is the convolution kernel and *C* is the number of channels, the authors propose three ways to equalize the dimensions and finalize on using a time-strided convolution operation. This is a 3D convolution of a 5×1^2 kernel with 2βC output channels and stride of α.

Classification Layer

The classification layer consists of a global average pooling operation that is performed on each pathway's output. The respective pooled features, 2048 dimensions for Slow pathway and 256 for Fast pathway as seen in Table 4.8, are then concatenated and used as the final video level representation. Thus, the video level representation from SlowFast is 2048+256 = 2304 dimensions. This feature vector is then passed to a fully-connected layer for classification with output as number of classes in the dataset.

4.4.2 Temporal Shift Module (TSM)

Up until 2018, the CNN architectures used for video action recognition had two prominent concepts; using 2D CNN for spatial representation and modeling temporal information using various aggregation techniques [1, 10, 12] or incorporating the temporal channel with the convolutions using 3D CNNs [14, 15, 38]. However, while these approaches were efficient they suffered two major shortcomings which were parameter explosion and heavy computation cost. Lin *et al.* proposed the Temporal Shift Module (TSM) for a high accuracy low computation cost video action recognition network. The authors focused on getting the high efficiency and accuracy of 3D CNNs while maintaining the complexity and costs of 2D CNNs. The resulting design achieved low latency in a real-time online setting and quickly became the state of the art for video action recognition on numerous popular datasets.

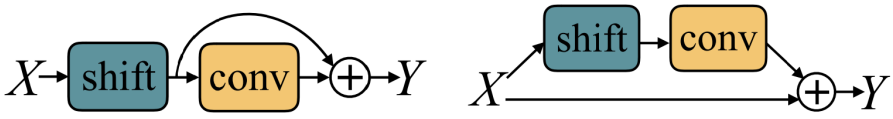


(a) The original tensor without shift. **(b)** Offline temporal shift (bi-direction). **(c)** Online temporal shift (uni-direction).

Figure 4.9: Illustration of the Temporal Shift Module (TSM). At any given instance T_i, the network is aware of channel information from the past frame T_{i-1} as well as the future frame T_{i+1}. In an online setting where future frames are not available (c), the network only relies on the past frames for sequential information.

The core idea of TSM is to shift part of the channels along temporal dimension to facilitate the exchange among neighboring frames. Thus information processed from any current frame also has the past and the future frames mingling within itself (bi-directional TSM). The channels from those frames are partly shifted into the current frame. Similarly, for an online setting where the future frames are not available, the current frame mingles with the past frames (uni-directional TSM). Figure 4.9 illustrates the idea of shifting channels in both offline as well as online settings.

The TSM module consists of two operations: *shift* and *multiply-accumulate*. The time dimension undergoes a +1 and -1 shift and multiplied and accumulated into the channel dimension. This is done using two approaches *partial shift* and *residual shift*. In partial shift, a certain fraction of channels are shifted from the past and future frames into the current frame. The authors shift 1/8 channels for this type of shift in each direction. This brings down the cost of memory movement.



(a) In-place TSM. **(b)** Residual TSM.

Figure 4.10: TSM consists of two operations: shift and multiply accumulate. In in-place shift(a), the specific network block is replaced by TSM block. In residual shift(b), the TSM block is inserted in the residual branch of the network. The latter approach gave better results.

When using a Resnet-50 backbone, replacing any specific block of the network with a TSM block can harm the spatial learning feature of the model. This type of replacement is known as in-place TSM as shown in Figure 4.10 (a). Thus, the authors insert the TSM module in the residual branch of the network as shown in Figure 4.10 (b). Another factor to be considered in the residual block is the proportion of shifted channels. Too small a fraction does not allow the network to understand the temporal reasoning for large temporal relationships (where the actions are slightly delayed rather than instantenous). As stated earlier, shifting 1/4 channels (1/8 in each direction) gives the best performance. This architecture gives 61.38% validation accuracy on the Something-Something v2 dataset. Additionally, this network is light enough to be run in real time on any smartphone as well [1].

4.5 Relabeling the Something-Something dataset

This section describes the core idea of our work. Starting with the experimental setup to run the networks in inference mode, the section moves to explain the protocol for a multi-person assessment of the legacy labels. This step involves details of an initial experiment conducted to verify the hypothesis that there are multiple valid interpretations to describe a video. The scaled up version of crowd sourcing the top-1 misclassification is explained in section 4.5.3. Finally, this section describes how the results obtained are consolidated and the rules employed to relabel a particular video sample.

4.5.1 Experimental setup

This section investigates the nature of misclassifications of the two networks discussed above: SlowFast [18] and TSM [16]. The Top-1 and Top-5 evaluation metrics are considered and the pattern of labels as predicted by both networks is inferred to be an indicator of what the networks possibly learn from the videos. The checkpoints of the best performance of the networks are avail-

[1] In this case however, the backbone network used for TSM changes from Resnet-50 to MobileNet V2 [67]

able online[23]. The networks are run in inference mode. Every video undergoes a preprocessing step as specified for video action recognition. A fixed number of frames (T) are sampled from the video. The video is normalized with the mean and standard deviation values of ImageNet. The mean values for the RGB channels are (0.485, 0.456, 0.406) while the standard deviation is (0.229, 0.224, 0.225) respectively. The short side of the video is resized to 256 and a center crop of 224x224 is obtained. Thus for a video with N frames, the final dimensions of the tensor that is input to the model are TxCxHxW. I use 32 frames for SlowFast and 16 frames for TSM as those gave the best performance. The SlowFast network has a validation accuracy of 60.25% while TSM has a validation accuracy of 61.38%. Table 4.3 shows the performance of both networks in inference mode.

Focusing on the misclassification numbers helps get an insight into the nature of mistakes of the two models. The results are consolidated by finding the union of the two networks. In total there are 9481 samples obtained from the union of predictions in the top-1 category and 2106 samples in the top-5 category. Before understanding how the labels are reassigned to the videos for evaluation, I process the file to examine the patterns of classes that are most confused as well as the classes that are least confused. Please note that these classes belong to the actual label of samples in the dataset, and not the selections by the workers. The latter will be discussed in section 5.1. Table 4.4 summarizes in decreasing order of counts of classes that are most confused as well as classes that are least confused in the Top-1 category.

Table 4.3: Top-1 and top-5 validation accuracy of SlowFast and TSM. The union of misclassifications from both networks is used for reannotation and re-evaluation of the networks.

Approach	Top-1 accuracy	Top-5 accuracy	# misclassified samples
SlowFast	60.25%	86.67%	9963
TSM	61.38%	87.04%	9568

[2]SlowFast: https://github.com/facebookresearch/pytorchvideo/blob/main/docs/source/model_zoo.md

[3]TSM: https://github.com/mit-han-lab/temporal-shift-module

Table 4.4: Patterns of classes that are most confused as well as least confused. Please note that these are the counts of samples corresponding to the legacy labels and not the selections by the workers. The most confused classes have multiple valid interpretations with other labels in the set causing confusions in model predictions.

Class Name	Count
Most confused	
Moving something up	168
Showing something to the camera	155
Putting something on a surface	150
Dropping something onto something	147
Stuffing something into something	146
Least confused	
Turning the camera left while filming something	7
Lifting a surface with something on it but not enough for it to slide down	6
Pushing something onto something	6
Spilling something behind something	4
Turning the camera downwards while filming something	4

4.5.2 Baseline experiment for multi-label interpretation of video samples

It is my hypothesis that humans perceive the same video differently based on certain characteristics like specific series of events or subsegments in a video, or the focus on specific objects in a frame or sequence of frames. To verify this hypothesis, a pilot study is conducted wherein the users are shown a video and asked to select all relevant descriptions of the video from the available choices. This initial experiment is carried out with a small sample size of 1000 samples belonging to the top-5 misclassifications category. The aim is to understand how users interpret a video when they are presented with options describing it. This tool is named as AnnotateMe. Every video was annotated by two participants to mitigate any annotation bias. The goal of the pilot experiment is to seek answers to the following questions:

- Do the network predictions align with the way humans perceive the videos?
- Are there multiple different interpretations of the same video as selected by the users?
- How many times is the actual label selected from the available choices?
- What are the patterns of selections for a particular class of videos?

Figure 4.11 shows the user interface of AnnotateMe tool. The video is seen on the left hand side while the video descriptions (or video labels) can be seen on the right. The label options presented to the users are obtained from the union set of predictions from the two models. The actual video label as associated with the video in the dataset also belongs to the choices presented to the users, but is randomized in occurrence to prevent users from tricking the system and selecting it as the only choice. Thus, if both networks agree on all the 5 predictions, then the intersection set contains 5 entries. Similarly, if neither networks agree on the predictions, then there are 10 entries for that video.

The users/annotators are shown the video and the corresponding labels for it and asked to select all the video labels that best describe the video. Every video has two confidence values: **Yes** and **Maybe**. If the user is confident of the selection, then Yes is selected, Maybe if the selection seems fitting but not completely. As in the given example in figure 4.11, the user is confident of the selection *showing something on top of something*, so it is marked as Yes. However, the selection *poking something so lightly that it doesn't or almost doesn't move* fits the video description but not completely. Thus, the confidence score for it is Maybe. The selected labels will be seen below the video below the Final Label heading.

In addition, they are also given the choice to add a video label describing the video if it is not present in the labels pool. This can be done by using two drop down menus seen below the label choices. To avoid overwhelming the user with all the 174 labels in the dataset, a two step process is set up. The user first selects the coarse group of the description from the first drop down menu. This menu contains 50 entries, *i.e.* the 50 coarse grained labels made available by the authors. Based on the coarse group selected, the second drop down menu updates its entries to all the fine-grained labels belonging to that group. Any selection made from the second drop down menu will be updated in label set for the video. Any ambiguous video sample goes for a second review by selecting the 'Mark for Review' checkbox on the left corner.

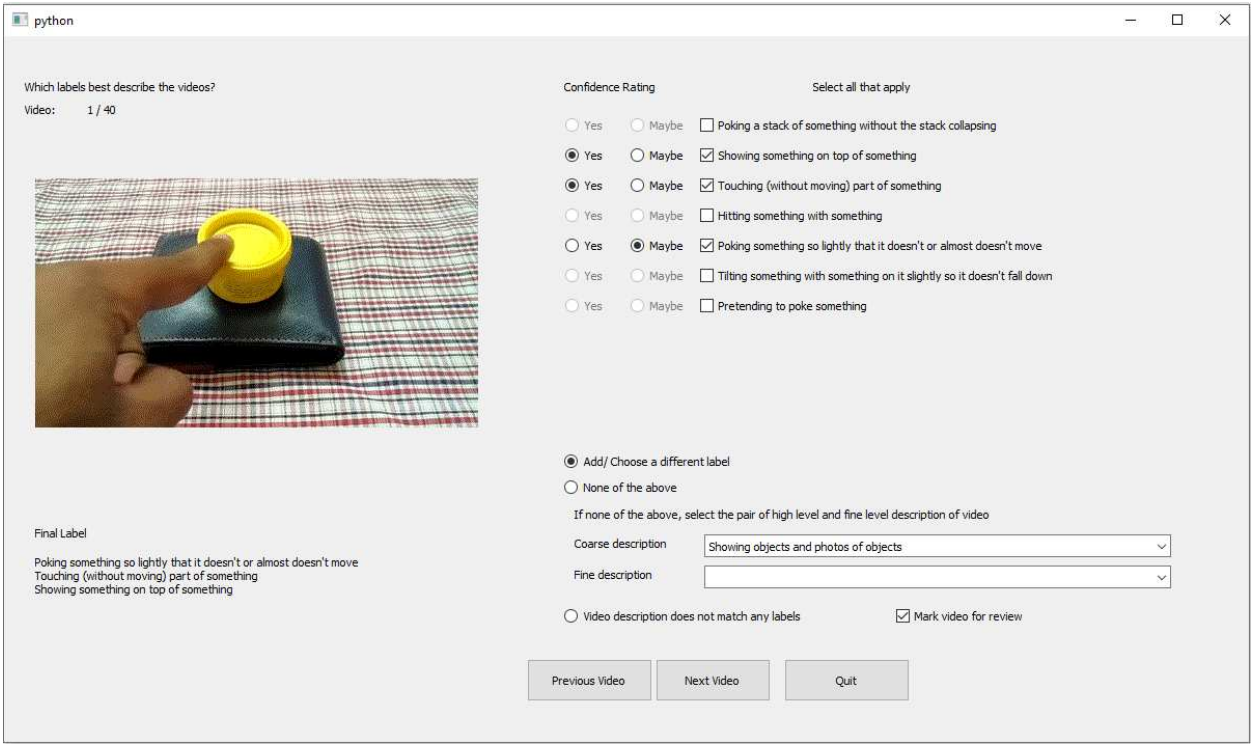


Figure 4.11: Illustration of the AnnotateMe tool used a pilot experiment to get annotations from humans. Users are tasked to select all relevant labels in the options that describe the video on the left side. Additionally, users can also select labels from the drop down menus that are not seen in the selections. Any ambiguous sample can be marked for review for a second pass.

4.5.2.1 Inferences

The goal of the pilot study was to answer the question whether there indeed are multiple ways of interpreting a videos and if the labels in the given dataset are relevant. For this, I got annotations for videos from multiple participants and analysed the patterns of selections. The actual label was included in the selections, and randomized to account for accidental bias. Table 4.5 summarizes the count of the number of annotators selecting the actual label. It is observed that out of 1004 videos where each video was annotated by 2 participants, in 527 samples of 52.49% of the total samples, both participants were in agreement in selecting the actual label in the descriptions of videos. One of the annotators selected the actual label for 32% of the total videos, while none of the annotators selected the actual label for 156 or 15.53% total samples.

Table 4.5: Summary of the results of the pilot experiment. 84.76% of the total samples had multiple labels selected. 52.5% total samples had the actual label that was agreed upon by both annotators. This result asserts the claim that there are multiple interpretations of an action in a video.

# annotators agreeing on the actual label	Count	Percentage
2	527	52.49%
1	321	32%
0	156	15.53%
Total number of samples	1004	

Similarly, independent of agreement, 851 samples or 84.76% of the total samples had multiple labels selected. There were 310 samples that had multiple labels out of 527 samples where both annotators agreed. This accounts for 58.82% of the samples. Out of the 321 samples, there were 125 samples that had multiple labels where either of the annotator selected the actual label. And 63 samples had multiple label selections where neither of the annotators selected the actual label in their selections. This result asserts my claim that there are more than one way of interpreting a video based on how language is used to describe the action in the video. As described earlier,

the labels presented to the annotators were obtained from the union set of the predictions of the SlowFast and TSM networks. Almost all annotators gave the feedback that the selections were relevant and positively aligned with the video shown. This helps to understand that the model are learning important characteristics of the videos when making predictions of the actions, and those that align with how a human, untrained for the specific task of classifying videos would perceive them as well.

While this pilot study focused on a very small sample set of 1000 videos belonging to the Top-5 category of misclassifications, the conclusion that there are multiple interpretations of a video and the action in it as perceived by humans is reassuring. Relabeling the Top-5 category can help to bridge the difference in the Top-5 validation accuracy, almost the entire community focuses and compares network performances based on the Top-1 accuracy. Thus, in order to produce a more coherent and consistent result of my hypothesis, I replicated the experiment to include samples from the Top-1 category as well. The following section describes the process to scale up the pilot experiment for a more thorough analysis.

4.5.3 Crowd-sourced annotations

The pilot experiment described above laid the foundations to include more samples belonging to the Top-1 category as well as some balance samples from the Top-5 category. A similar union of results from the two networks provided approximately 11,857 video samples. Of these, there are 9481 samples in Top-1 category and 1106 samples in the Top-5 category. These videos were crowd-sourced to Amazon MTurk[4] for annotations and designed as a video classification experiment. I (requester) submit the task for the annotators (workers) in return for monetary compensation upon successful completion of the task. The terms requester and worker are used interchangeably for me and the users respectively. Each task is known as HIT (Human Intelligence Task) and the worker is paid 3 cents ($0.03) upon successful completion of the task when it is approved. If the task is rejected, the worker is not paid.

[4]https://www.mturk.com

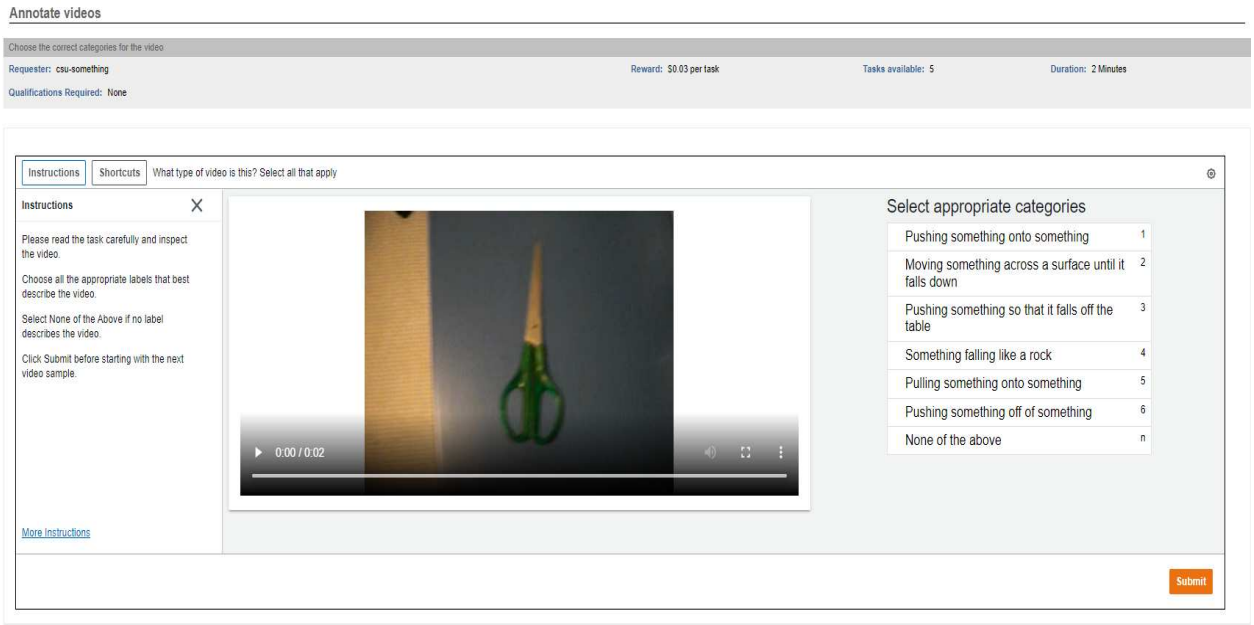


Figure 4.12: An illustration of the video reannotation tool as presented to the workers on Amazon MTurk, a crowd-sourcing platform. Each worker is shown a video and the corresponding labels describing the video. Workers are tasked with selecting all the relevant labels that describe the video. The options available are obtained from the union of the predictions from SlowFast and TSM.

Figure 4.12 shows an illustration of the MTurk experiment as presented to the users. The workers can select all appropriate descriptions of the video seen on the left. The workers volunteer to complete as many HITs as they like. No personal information of the workers is accessed in any way. We can apply certain filters to limit the number of workers for our tasks, but we did not apply such filters so as to maximize the responses for a more diverse data. A total of 1663 workers participated in this study. The maximum number of approvals received by a worker were 494 while the average number of approvals were 16.

Because we can anticipate erroneous selections from workers trying to game the system or in an attempt to mitigate annotation bias, every video is intended to be reviewed by three workers. It is inconvenient and inefficient to have to review 36,000 videos one by one. To make this process streamlined and efficient, another tool is designed to review the annotations obtained from workers on the MTurk platform. Figure 4.13 shows the interface of the tool used to review the videos, aptly named ReviewMe. Three annotations are collected for every video. These have to be reviewed

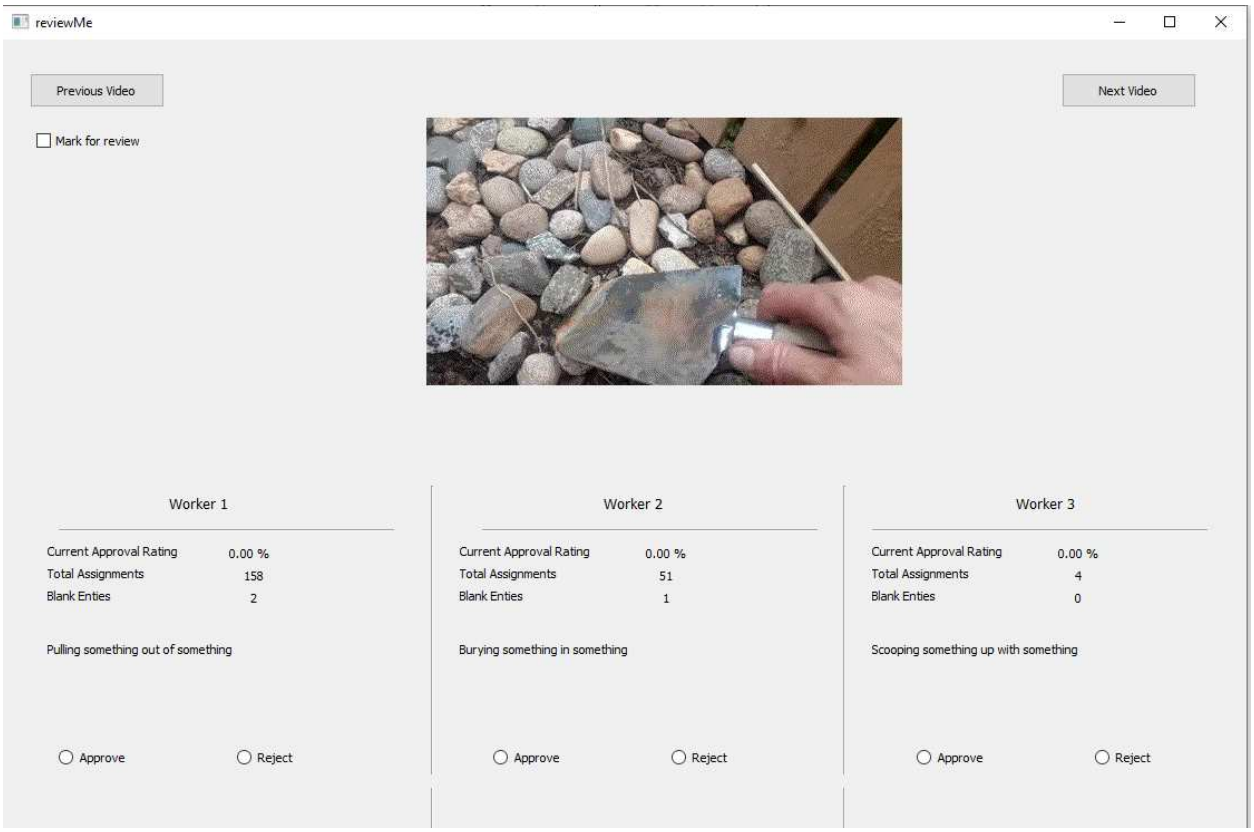


Figure 4.13: The ReviewMe tool is designed to make the assessment of worker annotations obtaine from MTurk more efficient and streamlined. Each worker's selections are seen below the video. Each worker column has a radiobutton to approve or reject a task based on how much it matches the video shown above.

individually before being approved or rejected. In order to save time and redundant efforts, the tool shows a video and the corresponding answers from the workers. No personal information of the workers is made available, and neither is any identifier seen in the interface. All workers are named as Worker 1, Worker 2 and Worker 3. Each worker's answers are seen below, along with the number of assignments completed, the number of blank entries by the worker as well as the current approval rating. Every HIT has two options associated with it: Approve and Reject.

If the worker does not select any option and leaves the task blank then the task is rejected and the worker is not paid for it. Similarly, if the selections are done in such a way that they are not close to the expected answer then the assignment is rejected. If the selection is correct or is close to the actual label, then the HIT is approved, and rejected otherwise. For example, if the relevant selection is *picking something up* and the subject selects *lifting something*, the task is approved.

However, if the worker selects *throwing something in the air* then the assignment will be rejected. At the end of every batch, all the results are compiled together, and a worker's HIT is approved or rejected using MTurk's API. The decision is then communicated to the workers through MTurk API.

4.5.4 Evaluation of human annotators

In order to quantify the user agreement of labels, we would have to apply agreement rating algorithms like Fleiss Kappa score [68, 69] and Cohen Kappa score [70]. Yet, this setup differs from the conditions required for the calculation of the scores thus preventing us from applying them to our data. Both algorithms assume that there is a single selection for a particular sample. But in this experimental conditions of this research, there is a particular interest in getting all relevant selections for the videos. The Fleiss Kappa score is inappropriate because while every annotator can make multiple choices, not every annotator must make the same number of choices. To address this issue, I rely on the evaluation judgement of worker submissions using the ReviewMe tool.

The underlying assumption in using the evaluations from the tool is that the worker selections have been meticulously verified with the corresponding video before approval or rejection. To ensure the integrity of the selections, for cases where there is doubt with the worker selections, the videos are marked for a second review thus eliminating any false approvals. Although these steps are taken to obtain as revised and clean data as possible, there are always some samples which slip through the cracks. Across all the batches, there are a total of 560 samples in the Top-1 category of videos wherein there were no approvals for either of the workers, or cases where there was only one approval for one worker, but the selection was not matching the actual label. To account for these incomplete or missing samples, I pass the samples through MTurk once again and use the best of both results. After the second round of curation, there were 506 videos that were salvaged and 54 videos that were not added into the final data to be used for evaluation. Removing the 54 entries resulted in a data that contains 9427 samples belonging to the Top-1 category.

Once the batches have been processed and reviewed, all results from MTurk are consolidated to analyse patterns of human evaluations of video labels. The Majority Voting strategy is applied by taking advantage of the reviews of worker HITs obtained using ReviewMe. The decision to assign a label or a set of labels to the video relies on three rules. These rules focus on the total number of approvals or rejections for a particular sample as well as the agreement of selections between the workers. The labels are assigned if one of the following conditions are met:

1. All HITs are approved and all workers agree on the labels. This is usually the condition where the actual label is present in the selection list, but also when additional labels are to be incorporated.

2. Two HITs are approved and both workers agree on the labels. This condition ensures a majority approval of workers to allow alteration of labels.

3. All the HITs are approved, but none of the workers agree on the labels. This case is particularly interesting because it leads to all the possible ways of interpreting a video correctly.

The results of the data are tabulated to understand how many times the actual label was selected by the annotators corresponding to the number of approvals. It can be observed that the upper bound of the label set cannot be determined with certainty as each worker can potentially select valid multiple interpretations for the video. In a condition where all workers are approved and each worker has selected one label that is different from the legacy label, the number of labels in the reassessed set for the video sample will be 4. However, there can be more than 4 valid labels in this set as well. The lower bound for the reassessed label will be 1 if all workers agree on the label and it is the legacy label for the video sample. In my experiments, the maximum number of reassessed labels was seen to be 8.

Table 4.6 illustrates the distribution of samples based on the number of worker approvals and the condition whether the actual label is present in the selections. This helps to get an insight into the numerous ways in which the workers interpret a video and how many times it aligns with the actual label in the dataset. Once all samples are reassigned following the rules mentioned above,

Table 4.6: The distribution of worker assessment verdicts with existence of labels in the selections. Out of 9427 samples, 4149 samples are where all 3 workers were approved, and selected the legacy label in their selections. We consider a Majority Voting using approval verdict for adding a new label to the label set.

Number of workers approved	Label in selection	Count
3	Yes	4149
3	No	1282
2	Yes	2359
2	No	871
1	Yes	753
1	No	13
Total samples handled		9427

the two networks are re-evaluated. The scope of this dissertation does not account for the relabeling of the training data and only examines samples in the validation set. This is because the training step involves random cropping and resizing, where the actual object in consideration might be lost particularly if it is filmed off center and more towards the edge of the video frame. Chapter 5 Section 5.1 discusses the change in accuracy, also denoted as ReaL accuracy and the inferences about the networks from it.

4.6 Applying Grad-CAM visualizations to videos

Chapter 3 Section 3.3 explained how Grad-CAM algorithm is applied to visualize the convolutional layers of a CNN model when it classifies an image input. As a quick recap, Grad-CAM is an approach to investigate the internal layers of a CNN model by highlighting the regions in an image that have a positive influence on the prediction of a particular class. It is used to study and answer the question *why a network predicts what it predicts* as a way to establish trust in the learning and predictive capabilities of the networks.

The two fundamentals of Grad-CAM are activation maps and gradients. Activation maps are obtained when an image is processed in the forward pass, while gradients are obtained in the backward pass. Heatmaps are generated based on the weighted interaction of activations and gradients and overlaid on the input image to highlight salient regions of the image that have a positive influence on the prediction of the class. While this technique was developed primarily for a static

image, it can be easily extended for videos. The focus is on how the concept adapts to the networks referenced in this thesis, *i.e.* SlowFast and TSM, and what insights are obtained from the visualizations of certain videos processed by these networks.

4.6.1 Grad-CAM for SlowFast

In SlowFast network, there is no down-sampling of the temporal information in videos. Thus, the number of frames input to each pathway remain consistent throughout the extent of the network. Let us consider a SlowFast network that takes as input a video segment consisting of D frames. For ease of understanding, we discuss the fast pathway first and how Grad-CAM is applied on it. The input to the fast pathway is a tensor of dimensions $D \times C \times H \times W$. In the last convolutional layer, the output dimensions of activation maps in the fast pathway are $[256 \times D \times 7 \times 7]$(4.4.1). The gradients also have the same dimensions as the activation maps. Following the tutorial on Grad-CAM, the pooled gradients or weights of every activation maps is a 1-D tensor of size [256]. These weighted kernels of size 7×7 are used to generate the heatmaps. The heatmaps obtained for each of the D frames are overlaid on the original input image and visualized. The Slow pathway is almost similar but with two minor changes. First, the number of frames input to it are D/4 instead of D. Second, the number of activation maps in the last convolutional layer are 2048 instead of 256.

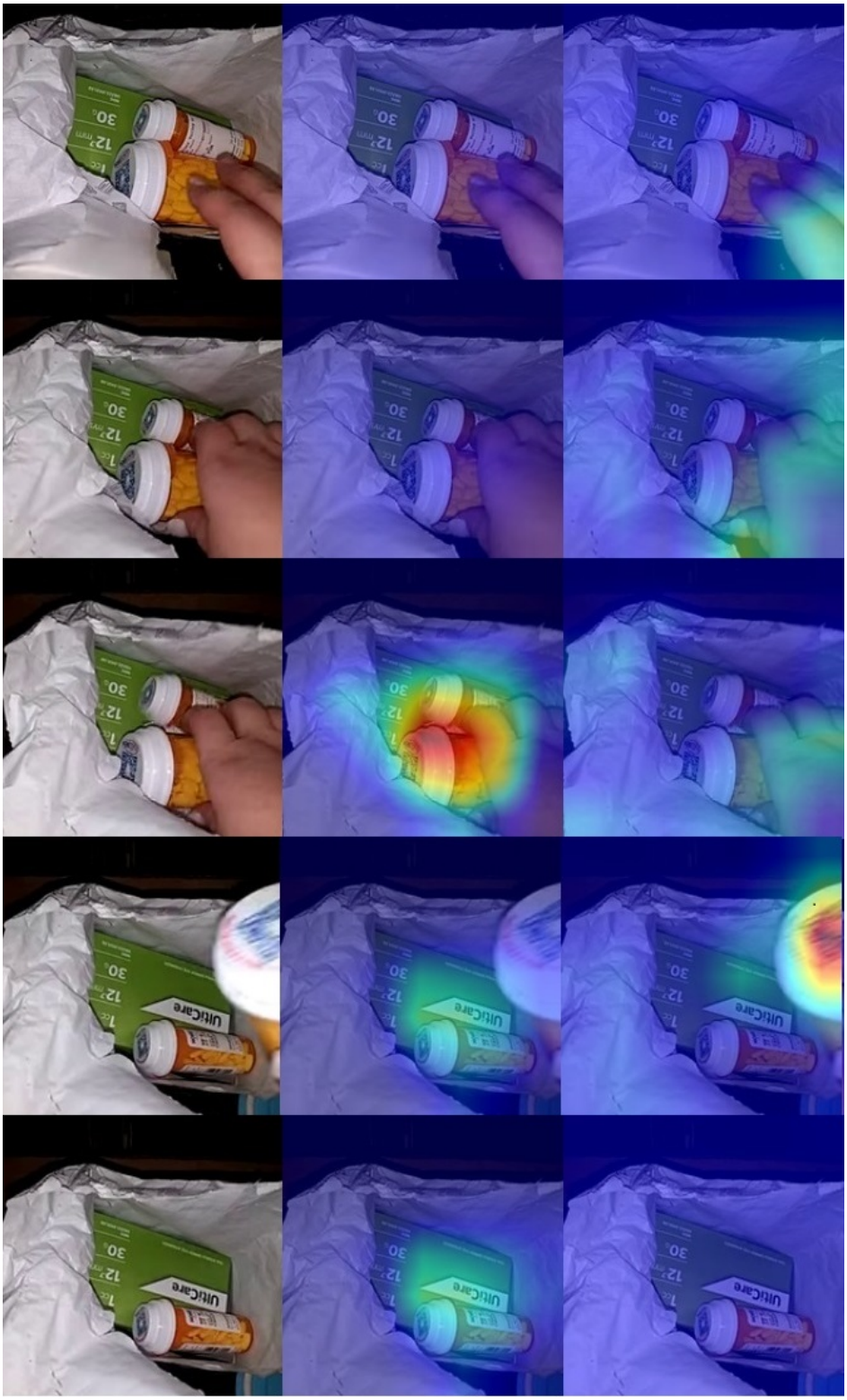


Figure 4.14: Grad-CAM visualization for *taking something out of something* using SlowFast. While the Slow pathway (middle column) focuses on the object interacted with, the Fast pathway (last column) focuses on the movement of the hand during the length of the video.

Figure 4.14 shows the Grad-CAM visualization for selected frames from a video with action label *taking something out of something*. The first column shows the original video frames, the second shows the heatmap of the Slow pathway overlaid on the video frame while the third column shows the heatmaps of the fast pathway. The Slow pathway focuses more on the object that is being interacted with in a video and its change as the video progresses. In comparison, the fast pathway focuses more on the motion of the hand as it performs the necessary action through the video.

It is observed that as the hand starts to reach the medicine bottles in the first frame, the region around the hand in the fast pathway gets highlighted. In the third frame, the Slow pathway highlights the medicine bottle that is being picked up. The Fast pathway continues to focus on the hands, but the activation is not as high. In the following frame however, the Fast pathway again highlights the hand as it moves away from the bottle and out of the frame boundary. The last frame shows that the medicine bottle left on the surface is highlighted thus implying that the Slow pathway focuses on the objects being interacted with.

4.6.2 Grad-CAM for TSM

Similar to SlowFast, there is no temporal down-sampling of video frames in TSM. Thus the number of frames remains consistent throughout the depth of the network. The frames mingle with each other due to the temporal shift function in the residual layer of the network. In the penultimate layer before the softmax function is applied, the results from the frames are aggregated using a consensus function, usually Average Pooling over the video frames. Comparing with the example of how SlowFast processes the video, let us consider a video consisting of D frames as input to the network. Thus, the input dimensions are $D \times C \times H \times W$. In the last convolutional layer, the dimensions of the video as fed to the consensus function are $[D \times 2048 \times 8 \times 8]$. This is the input tensor used to generate the activation maps, and the kernel size of 8x8 is used for overlaying on the frames for visualization.

The dimensions of the gradients are same as the activation maps in the forward fuinction *i.e.* $[D \times 2048 \times 8 \times 8]$. The weighted gradient is a tensor of size [204] indicating one weight value

for every activation map present in the last layer. Thus, the consolidation of all the activation maps results in a tensor of size $[D \times 8 \times 8]$ which is used to overlay onto the video frames, one for each frame.

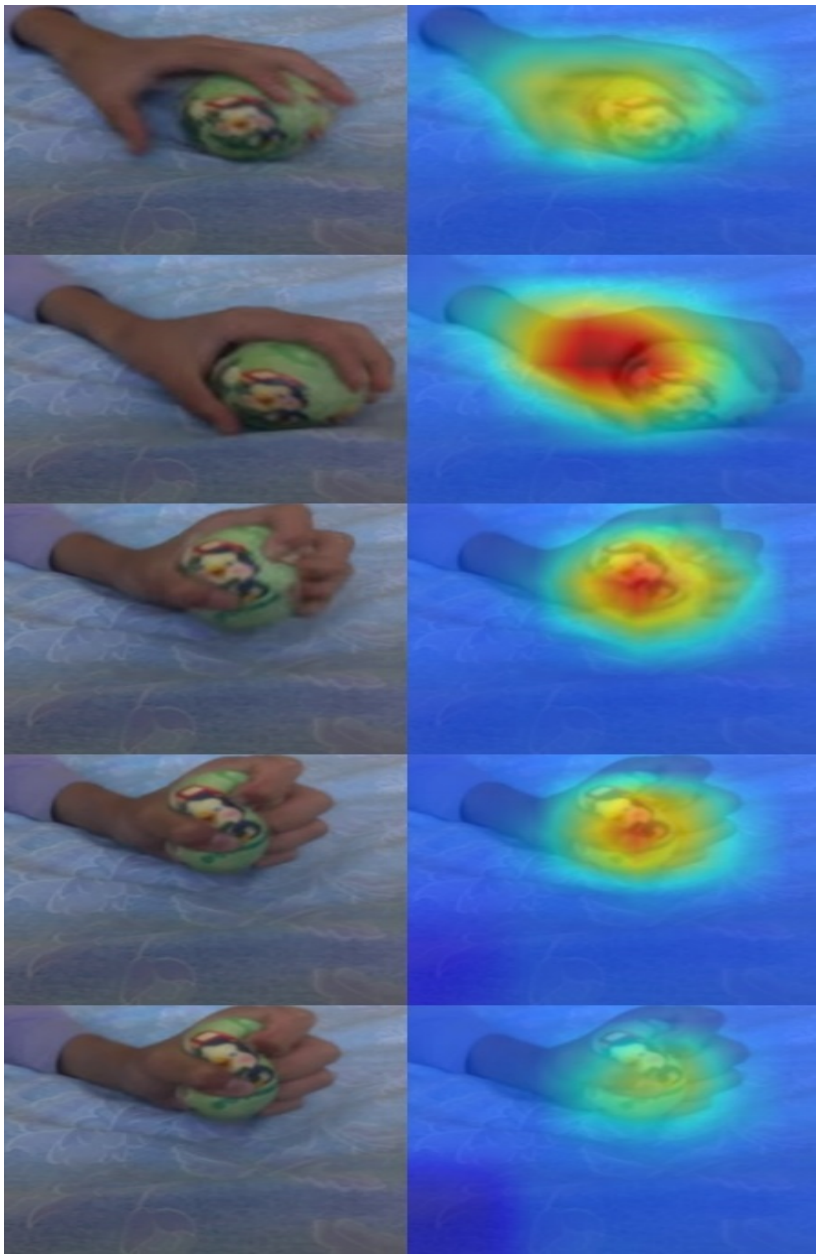


Figure 4.15: Grad-CAM for *Squeezing something* using TSM. The area around the ball reduces in size as the hand moves forward to grab and squeeze it. TSM focuses more on the target object rather than the hand, much like the Slow pathway in SlowFast network.

Figure 4.15 illustrates selected frames from a video with class label *Squeezing something*. The first column shows the original video frames, while the second column shows the heatmaps overlaid onto the frames. Contrary to the SlowFast illustration where is one visualization for the Slow pathway and one for the Fast pathway, TSM only has a single pathway as the video is processed. While there are subtle differences between the two networks, the TSM visualization can be considered to be more aligned with the Slow pathway processing spatial information over the duration of the video.

It is observed that as the hand moves closer to the ball, the network highlights the ball as the target object interacted with. As soon as the ball is grabbed, the region immediately locks in on the object as indicated by a red region around the boundary of the ball and the hand. When the ball is pressed, the region around the ball not only coincides, but also reduces in size. This indicates that the model can learn the reduction in size of the target object. This pattern continues as the ball is further pressed in the fourth frame. The progression of the five frames shows the complete nature of how TSM understands or focuses on when it predicts an action of *squeezing something*.

Chapter 5 section 5.3 presents some case studies of agreements and disagreements of SlowFast and TSM. It paves a way for the reader to establish trust in the networks as they process video input to predict a particular action. The discussion highlights how two similar actions can be confused with each other, what relevant commonalities are seen between the two actions that compel the confusions for the networks. It also implies the core inconsistencies of the data collection and labeling step as the main source of the reduced model performance.

Chapter 5

Results

This chapter provides the effect of Reassessed Labels of the validation set of Something Something dataset, and the insights from ReaL accuracy on video action recognition models. An analysis and evaluation of the labels obtained from crowd-sourced experiments is presented, along with the patterns observed by the human-in-the-loop approach.

The goal of the experiments is to understand the effects of ambiguous language in the labels of the Something Something dataset on the performance of video action recognition networks. A direct cause of the ambiguity can be traced to the inconsistency in data collection and curation step of the dataset and how it translates to model performance. A video involving human object interaction can be interpreted in more than one ways. For example, *dropping something into something* can also be interpreted as *putting something into something* based on how similar the videos look visually. Similarly, when the class labels describing the videos are linguistically close to each other, the possibility of confusions in classifying them is higher. My argument is that the networks are learning very relevant properties of actions from videos and can predict the classes with a good confidence. However, the validation accuracy of these network is hurt when the predicted label does not match the actual label due to an error in the data itself. To explain, I took two different approaches and arrived at the same conclusion.

First, I introduced a human-in-the-loop approach and review each video in the validation set through a crowd-sourced platform. A user is shown a video and a corresponding set of labels associated with it. The task is to select all relevant choices that describe the video. To remove user bias due to attempts to game the system, annotations from three users are collected. User selections are compiled and the labels are reassigned based on three rules using approvals of worker annotations. The new dataset thus obtained is referred to as Reassessed Labels. the networks are then re-evaluated using the new labels and refer to it as ReaL accuracy. Second, the internal workings of the networks are probed as a way to corroborate what networks learn when predicting

a particular class label. These internal workings of the networks are visualized using the Grad-CAM [25] approach. It is observed that the heatmaps for similar classes are similar *e.g. Pushing something slightly so that it moves* and *Poking something slightly so that it moves*. There are many videos in the *Stuffing something with something* class that are confused with *Putting something into something* or *Dropping something into something*. Surprisingly, upon observing the videos manually, there is more agreement with the network's first prediction that is with a high confidence than what the actual label is for the video. This infers that the model can predict a class correctly, but still gets it wrong because the actual label itself is wrong. This is explained in more detail in section 5.3.2.

The rest of the section is as follows: Section 5.1 describes the re-evaluation of the networks using the reassessed labels. This section also discusses the patterns of co-occurring classes where the workers select one class when they select the legacy label corresponding to the video. A change in the original and ReaL accuracy for SlowFast and TSM is examined. Section 5.3 handles case studies of Grad-CAM visualization for a varied configurations of videos that are correctly classified by both networks as well as the confusions in model predictions. This section also gives the reader to understand the similarities in the internal learning when the network predicts a label that makes sense for the action being performed. Lastly, Grad-CAM is extended from the last layer to all other layers of the network to study the progressions of the network's learning as it gets better at predicting an action.

5.1 Re-evaluating the models

In this section, I discuss the patterns of reassessed labels obtained from the human-in-the-loop step using Amazon MTurk, a crowd-sourced platform. Before studying the impact of the new annotations on the model performance, we will briefly look at the co-occurring classes obtained from the new annotations in section 5.1.1. In particular, the focus is to see what percent of the total samples of one class were attributed to the other class. Table 5.1 lists the top-5 classes with the highest percentage of co-occurrence of classes.

Table 5.1: Top-5 classes co-occurring with the legacy label as selected by crowd-sourced workers. We observe that based on what the action label changes based on what target object is considered. Similarly, a label is confused with its parent class. The influence of ambiguous language of labels and visual overlap of actions results in multi-label selections by the workers.

Class Label	**Count**	**Co-occurrence class**	**Count**	**Percent**
Pouring something out of something	25	Pouring something into something	21	84%
Poking a hole into some substance	11	Poking a hole into something soft	9	81.82%
Letting something roll along a flat surface	113	Rolling something on a flat surface	91	80.53%
Throwing something onto a surface	71	Throwing something	56	78.87%
Lifting something up completely without letting it drop down	46	Picking something up	36	78.26%

5.1.1 Patterns in co-occurring classes

It is observed that the co-occurring classes semantically align with the actual class labels. Based on which object is considered as the focus object in the video, the label selection is changed accordingly. This is evident from *pouring something out of something* where the focus is on the object pouring the liquid but is confused with *pouring something into something* where the focus is on the object being poured into. While the main action is pouring, both actions are correct for the given video. Similarly, an action is confused with its parent class. This is seen for *throwing something onto a surface* and *throwing something*. Such instances where the parent parent label and the child label are considered as distinct labels is one of the core sources of confusions of model predictions for this dataset. We also observe that *throwing something in the air and letting*

it fall is often confused with *something falling like a rock*. The annotators consider the main action as the object that is falling down rather than throwing it in the air. Similarly, when an object is dropped but the surface on which it lands or might land is not shown, annotators tend to anticipate a surface and thus select the label *dropping something onto something* in place of *something falling like a rock*.

Based on how a person perceives the positions of two objects relative to each other influences their selection of labels. For example, depending on which object is considered in focus at the time of selecting a label, a change in perception when two objects are shown next to each other can lead the worker to select either or all of the three labels: *showing something next to something* as well as *showing something behind something* or even *showing something in front of something*. A parallel trend is observed when workers confuse the prepositions front, behind, into, onto, etc. For 75% of the samples of *twisting (wringing) something wet until water comes out*, workers also selected *twisting something*. This tells us that while an action is a unit of cause and effect, humans often consider a hierarchy of the abstraction that describes the video, and then make it more verbose.

When no action is performed in the video, the most common label that is selected is *showing something to the camera*. On a fine level the position and placement of the hands can help discern between poking and pushing an object. However, when the main cause distinguishing one action from another involves the application of a slight force to move an object, a poking action is confused with a pushing action for 75% of the samples. There are also instances where number of objects are stacked together. Workers tend to select *putting number of something onto something* 70% of the time in place of the stacking label. Overall, it is observed that there are myriad reasons considered by the workers when they select a particular to describe a video. While access to any personal information like knowing whether English was the person's first language was unavailable, I observed that workers with a good grasp of the language often selected more verbose options and had multiple selections rather than a single one.

The influence of Reassessed Labels and how ReaL accuracy compares to the original validation accuracy will now be studied.

5.1.2 Real accuracy for SlowFast and TSM

Using the proposed ReaL accuracy, I re-evaluate SlowFast and TSM. It would be expected that the accuracy would be higher if the human annotators agreed with the first predictions of the networks. However, this was hidden from the annotators when the order of the labels was randomized. The predictions for each sample are available when the two networks were run in inference mode after training. To evaluate the ReaL accuracy, I check if the Top-1 prediction of the network belongs to the set of labels now available for the sample. If the prediction does not belong to the set of labels available, then it is deemed incorrect. The ReaL accuracy corresponds to the number of correct predictions out of the total number of samples present.

Table 5.2: The performance change from the original accuracy to ReaL accuracy when SlowFast and TSM are re-evaluated using the reassessed label set. We observe an increase of almost 12% top-1 and 3% top-5 for both networks indicating that both networks respond positively to the inclusion of multiple interpretations of actions in a video.

Network	Category	Original accuracy	ReaL accuracy
SlowFast	Top-1	60.25%	**72.21%**
	Top-5	86.66%	**89.31%**
TSM	Top-1	61.38%	**73.34%**
	Top-5	87.04%	**89.83%**

Table 5.2 summarizes the difference in performance of the two networks between the original validation accuracy and ReaL accuracy. The ReaL top-1 accuracy implies that the first prediction falls in the set of Reassessed labels available for the sample. Both networks have an increase of almost 12% in their top-1 accuracies. The same change is also reflected in the top-5 accuracy, where SlowFast sees a change of 2.65% and TSM sees an increase 2.79% in the top-5 accuracy.

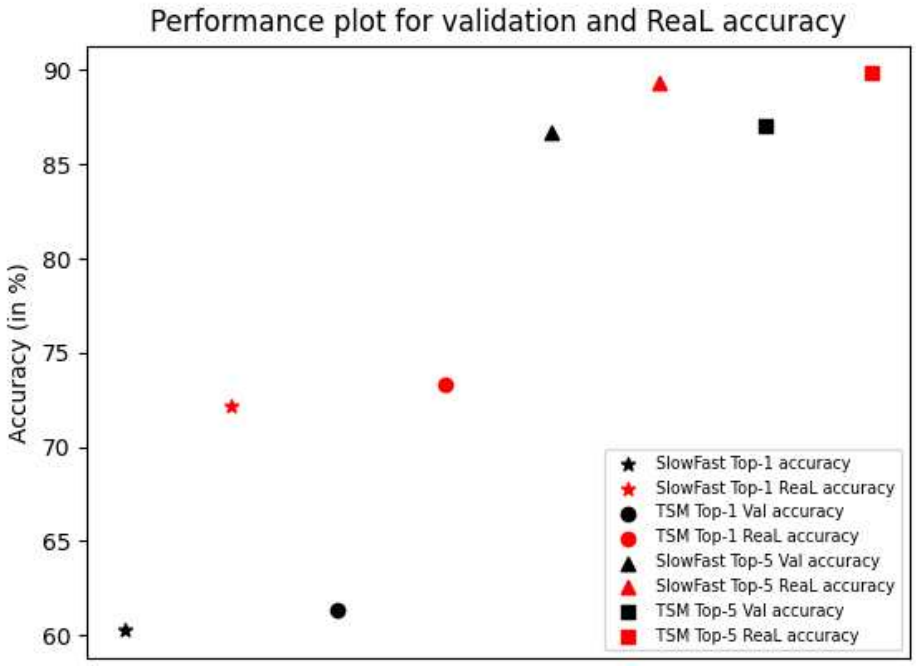


Figure 5.1: Scatter plot of comparative performances of SlowFast and TSM based on original and ReaL accuracy. The Y axis shows the accuracy in %, but the X-axis is not shown to have a greater spread of values across the plot for ease of interpretation. Black and red correspond to SlowFast and TSM respectively.

Figure 5.1 illustrates the change in performance of the two networks, *i.e.* the plot of original and ReaL top-1 and top-5 validation accuracies for TSM and SlowFast networks. Each metric is marked with a different marker. The black markers represent the original validation accuracy, while red markers represent the ReaL accuracy for the metric under consideration. The Y-axis shows the accuracy values in %, but the X-axis is not shown to have a greater spread of the values across the graph for ease of interpretation.

5.1.3 Extending ReaL accuracy to training data

In the scope of my research, I have focused on the re-annotation and reassessment of the samples in the validation data. In order to apply the re-annotation protocol to the training pipeline requires some changes in the way a forced choice training is carried out. Conventionally, the label

for a sample is represented as a one-hot encoded vector for training. In keeping with the same design, the model will predict a probability distribution after the softmax layer. The index of the maximum probability value in the output (also denoted as argmax) will be used to calculate the softmax loss with the ReaL set. Thus, if the argmax of the model prediction belongs to the ReaL set, then it will be treated as a correct prediction. However, the possibility of multiple labels for a single video sample results in moving to a k-hot encoding vector rather than a one-hot encoding. Thereby, the cross entropy loss will have to be changed appropriately to account for all of the labels now available for the sample. Instead of treating the problem as a multi-class classification problem, it is considered as a binary classification problem for every independent label in the ReaL set. A sigmoid cross entropy loss which does not require mutually exclusive predictions will be used to train the network [24].

As a result, the concept of reassessed labels can be incorporated into the training pipeline to study how the internal network learning is altered if it is no longer restricted to a single label and allowed multiple valid interpretations of the video under consideration.

5.2 Comment on the dataset

This section is not required, but I was compelled to note it for the larger audience. There are some vile and inappropriate videos in the Something Something dataset. For example, one video shows ear-wax being taken out of a person's ear and another video shows a person spreading a green oil on another person's back, just because the label corresponding to the video in question was for *Spreading something onto something*. This was taken even further when a person recorded a video of themselves unzipping their pants. The video has a label *Moving part of something*, but the video is demonstrably inappropriate. People have recorded videos without proper clothing, in a semi-naked state. While the focus of the dataset is on the action and not the background, the videos are extremely disturbing and must strongly be curated and removed from the dataset. In bringing a massive dataset with number of challenges to tackle in videos, the authors did not focus on censoring such inappropriate materials within the videos. the only attributes the videos were

checked for were the consistency of the length of the videos and the uniqueness of the videos, and probably violence or pornography. There was very minimal curation about whether the videos were appropriate for the labels, or even appropriate for viewing. This however, might be a personal remark and does not affect the conditions or results intended in getting real world videos from this dataset for the purpose of this dissertation.

5.3 Peel the onion: What do the networks learn internally?

This section presents some case studies of network predictions for certain video samples using Grad-CAM visualizations. I extend the same principle of visualization for a single image to videos. A heatmap is generated for every frame in the video, or for an image at a fixed stride (usually 4 for SlowFast).

5.3.1 When both networks agree on the actual label

5.3.1.1 Pretending to squeeze something

The video seen in Figure 5.2 is of a person squeezing something. Both networks, SlowFast and TSM get the prediction correct for this video. As seen for the SlowFast Grad-CAM visualization for *Squeezing something* in Figure 4.14, we observe that as the object gets squeezed, the area of the region of heatmap starts reducing gradually. Similarly, in this example as well, when the tube is being lightly squeezed, the are of the region where it is squeezed gradually reduces as seen in column 2 (Slow Pathway). Additionally, in the last frame, the region around the fingers is highlighted in column 3 (Fast Pathway) indicating that the hand will be released in the next instance.

This order of rise and fall of activated regions, combined with the slow press and release of the tube by the hand in the Fast Pathway makes the networks predict the action as happening but not enough to conclude as complete. Thus, both networks accurately predict the example video as *Pretending to squeeze something*.

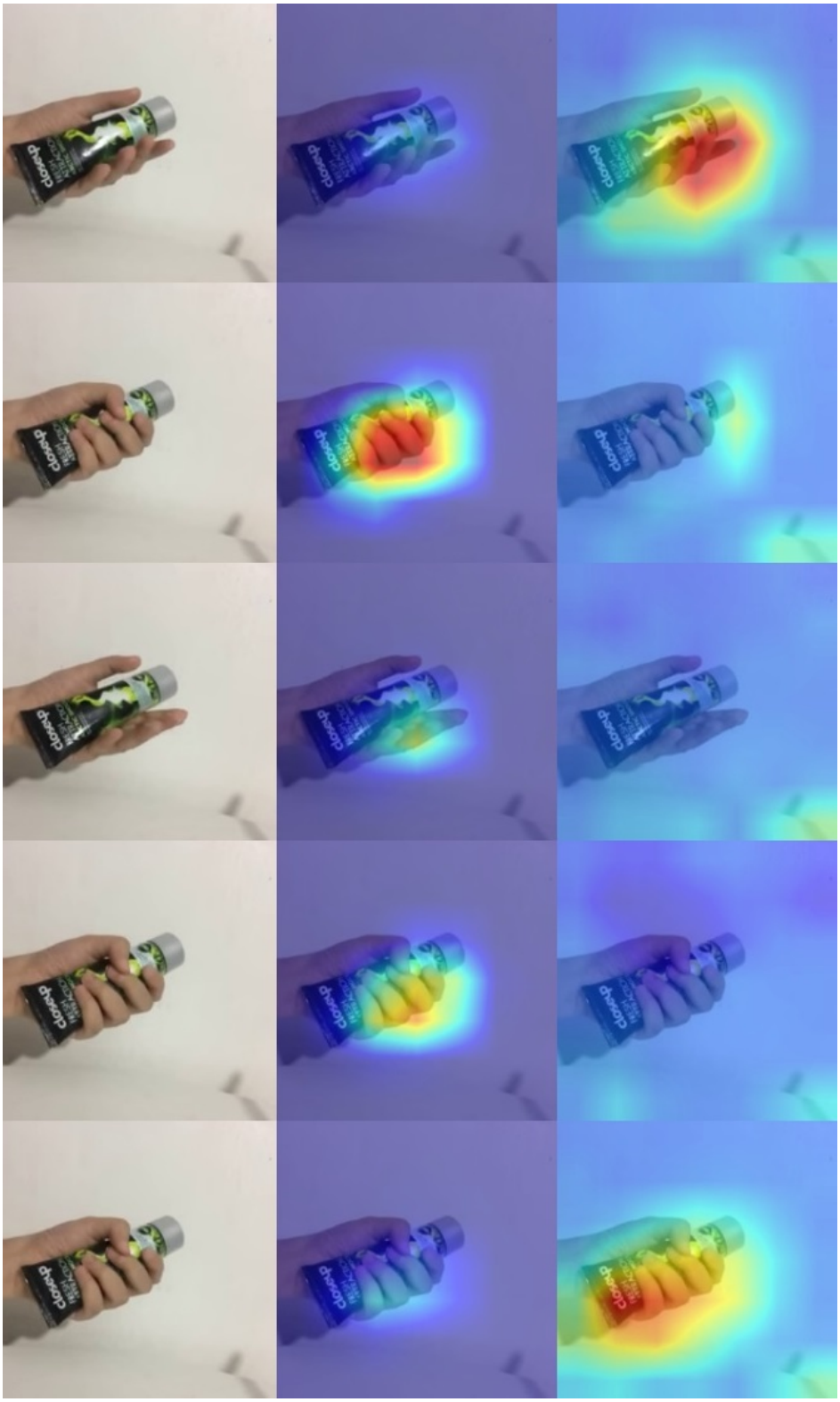


Figure 5.2: SlowFast Grad-CAM visualization for an action *Pretending to squeeze something*. The rise and fall of activated regions, combined with the slow press of the tube by the hand in the rightmost column makes the network predict the action label correctly.

5.3.1.2 Pushing something from left to right

The video seen in Figure 5.3 is of a bottle being pushed from left to right. The TSM Grad-CAM visualization for this video shows that a number of aspects about the video. Starting with the attention region highlighted around the bottle. Since this is the start of the action, the attention is not as evident as in subsequent frames, but indicative of the object is focus that is interacted with. In the next frame, we surprisingly observe that the network not only tracks the object, but also is slightly ahead of the position of the bottle. This indicates that the temporal information encoded within the network looks at the future frame, or prediction of what might happen next. I hypothesize that this is indicative of the network's ability to learn direction. This information is vital as a slight change in its anticipated trajectory or position can lead to a misprediction.

Again in the third frame, the attention is at its max as the bottle is in the middle of the action *viz.* being pushed. Note that since TSM has its backbone as ResNet-50, and is only modeling temporal information within but keeping the overall structure intact, the Grad-CAM visualization is more spatially focused in its attention regions, and implies temporal properties subtly within itself as is evident in this case study.

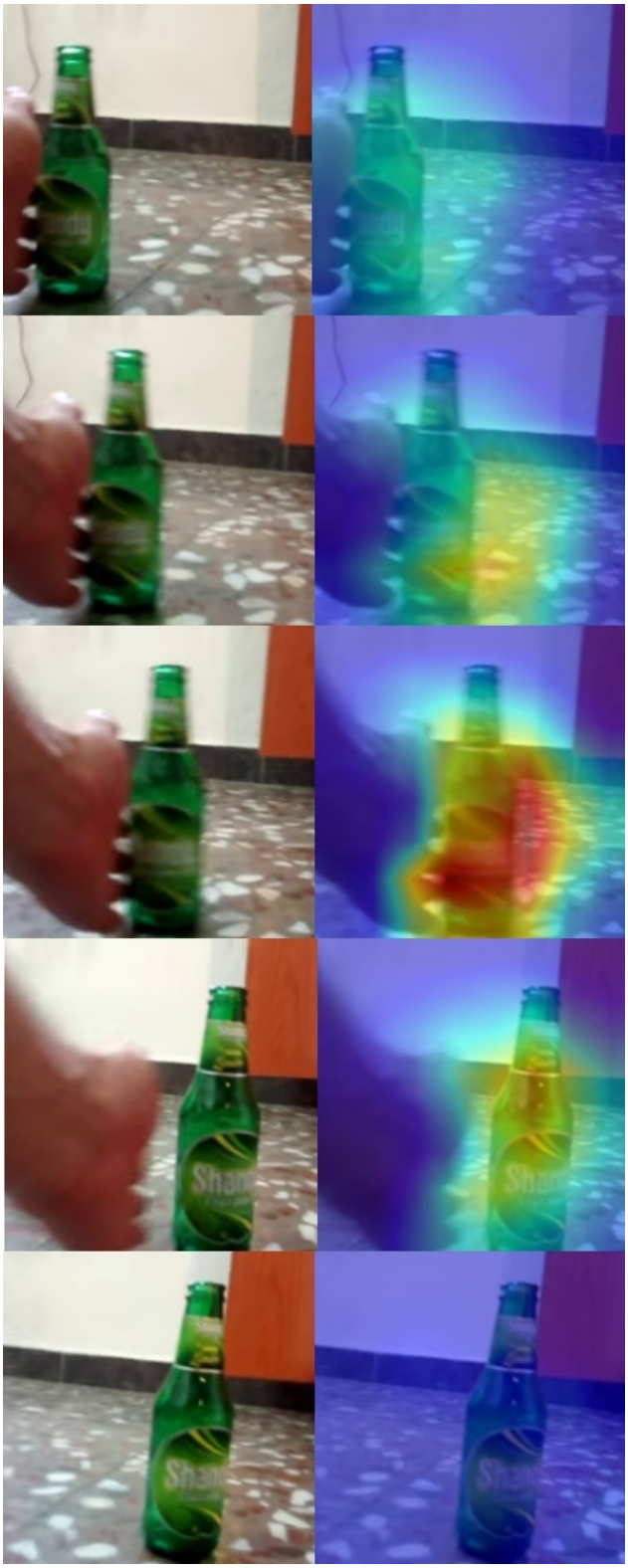


Figure 5.3: TSM Grad-CAM visualization for *Pushing something from left to right*. The network not only focuses on the object, but is slightly ahead of it. The future frame included in the TSM design leads to predict the direction of object motion (left to right).

5.3.2 Pattern of confusions of classes

This section studies the pattern of confusions commonly seen between classes. For ease of understanding, I present two classes and their common confusions predicted by the networks. These confusions will be studied based on how the Grad-CAM visualizations differ for each. In each case, it is observed that when the network confuses between two classes in question, they are usually very similar visually. Additionally, their Grad-CAM maps are also very similar. The network is focusing on subtle differences both in the spatial as well as temporal information when it makes a certain prediction. In all the cases explained, we observe that the network prediction is actually more relevant and corroborates with what a human would perceive as important signal when identifying an action. However, these are also the videos where the networks supposedly make a mistake. This mistake is not at the network end, but more at the source where the video was collected and/or labeled.

5.3.2.1 Folding something confused as Closing something

Both *folding something* and *closing something* can be seen as being visually very similar in a number of overlapping contexts. While the action of folding something is particularly associated with paper-like objects, closing something has a much broader scope based on what objects can be interacted with. I consider the two classes and filter out the samples which have been confused for each other by TSM model.

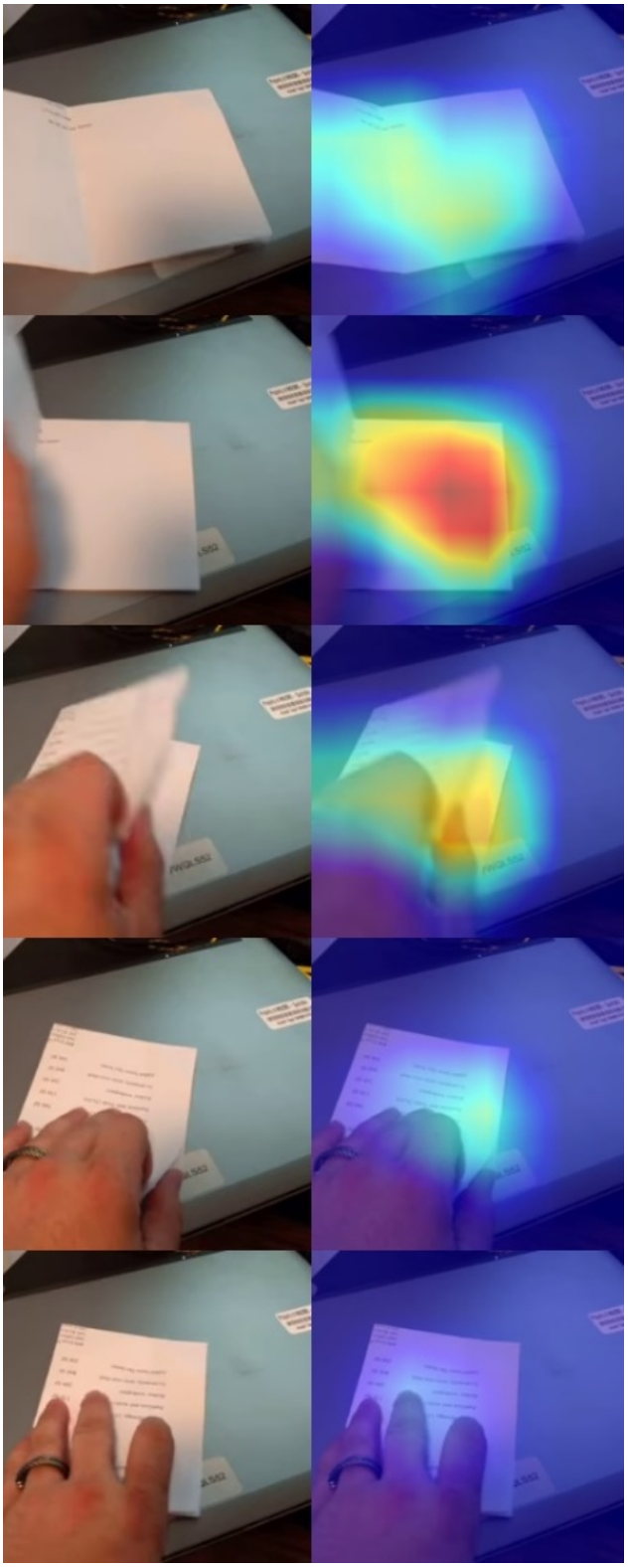


Figure 5.4: The legacy label corresponding to this video is *Folding something*. However, the model predicts it as *Closing something* with 95.95% confidence. The target region where the half of the paper is folded onto has the most saliency, which prompts the prediction of closing something.

In figure 5.4, the video is of a person folding a piece of paper. The crease of the paper along which it is folded is seen clearly in the original frames on the left column. As the action progresses, it can be observed that the Grad-CAM visualization has highlighted very relevant regions as seen by the model. This includes the region of both halves that are interacting with each other in the first picture, the paper onto which the other half is folded as well as the transition from partial covering to complete covering onto the paper. While the network predicts this action as *Closing something* with 95.95% confidence, the actual label is *Folding something*. This is a rather tricky sample because both interpretations makes sense for the model.

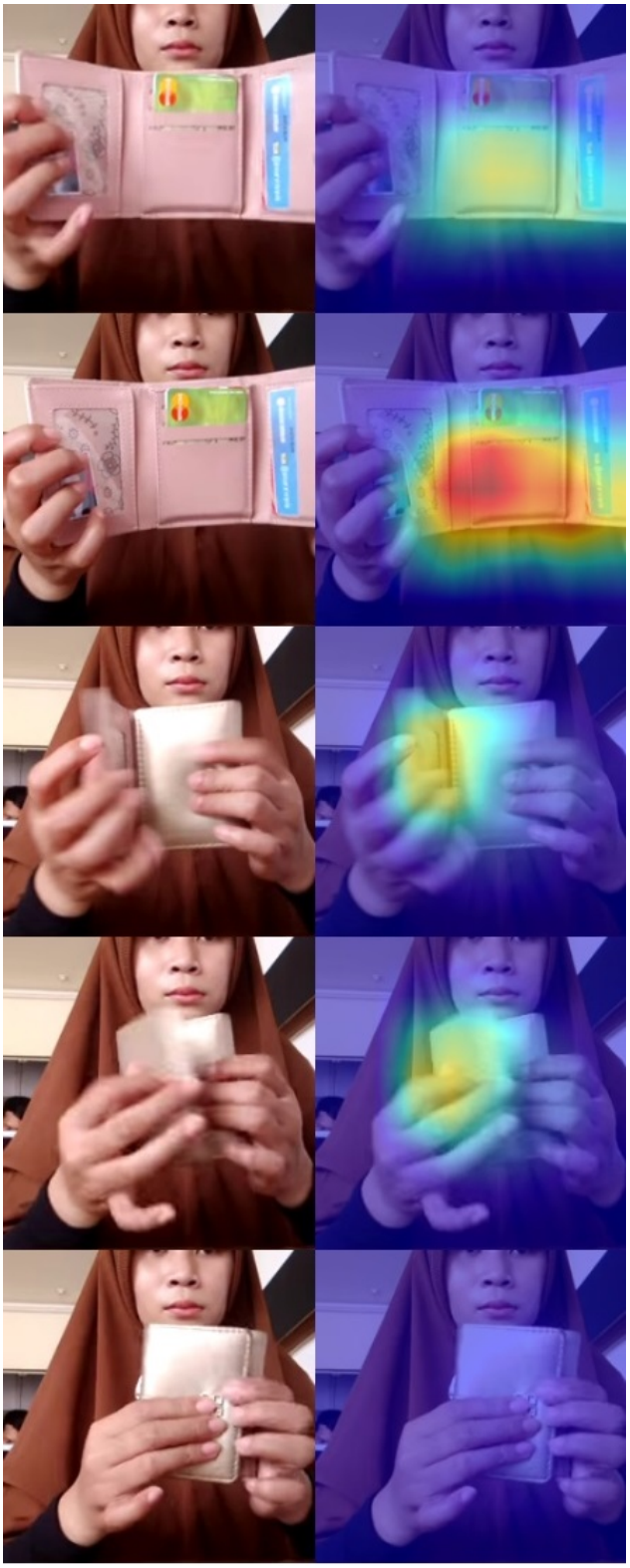


Figure 5.5: The video is of a person closing an object. However, in this case the model predicts it as *Folding something* and pays attention to the crease of the flap of the object that is being folded. Visually, it aligns with how a human would interpret this action, yet the model fails in its prediction.

In figure 5.5, the person closes the two flaps of the wallet. The model predicts the action *Folding something* for this video with a confidence of 98.61%. When we compare the visualizations on the right to the action predicted, the third frame reveals the main reason for the prediction. Folding an object can be understood when a crease along which the flap is folded can be seen. Similarly the first frame implies the crease, although it also considers the surface on which the flap folds. The fourth frame focuses on the hand as it completes the action.

5.3.2.2 Stuffing something into something confused Putting something into something

Visually, stuffing something involves a repetition of the action when the container is smaller than the object, or the object is longer in dimensions than the container. There is a slight struggle that can be seen as the actor carries out the action. Putting something into something, on the other hand, involves no such struggle. There is no repetition of the action and action is complete within the first try.

When we visualize case study videos of the two classes when they are confused with each other, we observe that when the network predicts the other class, it usually does so with a high confidence over 85-90%. Additionally, when verified visually, the network prediction wins over the actual label. To illustrate this point, we present the case study of two videos, where the network predicts the other class and wins. The frames do not display the temporal aspect of the video that helps determine the classes, but this can be better understood when the complete video is played.

In Figure 5.6, the video is of a person putting a spoon into a cup. The model predicts the label as *putting something into something* with 99.38% confidence. However, the actual label for this video is *stuffing something into something*. Visually, there is no struggle to put the spoon into the cup, nor is there any repetition. Thus, this is a case where the actual label is incorrect, while the model is able to predict the right label and therefore wins.

Similarly, in Figure 5.7, the video shows a person stuffing a tissue paper into its box. There is non negligible struggle as well as repetition involved in completing the action. The network predicts the label for this video as *stuffing something into something* with 94.95% confidence.

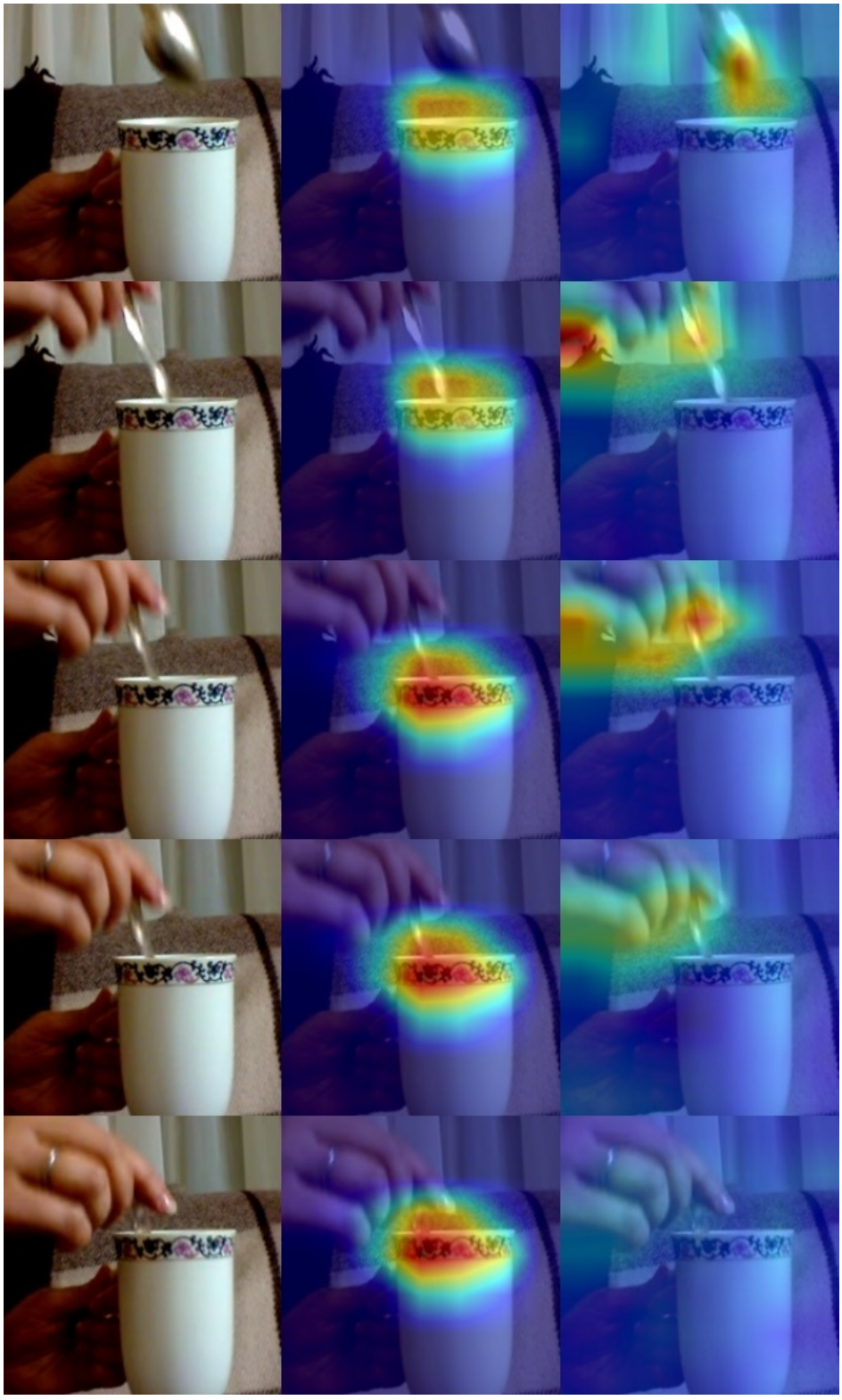


Figure 5.6: The video is of a person putting a spoon into a cup. The model predicts the label for this video as *Putting something into something* with 99.38% confidence. However, the actual label for this video is *Stuffing something into something*. Visually, there is no struggle to put the object into the cup, and thus the model wins over the actual label.

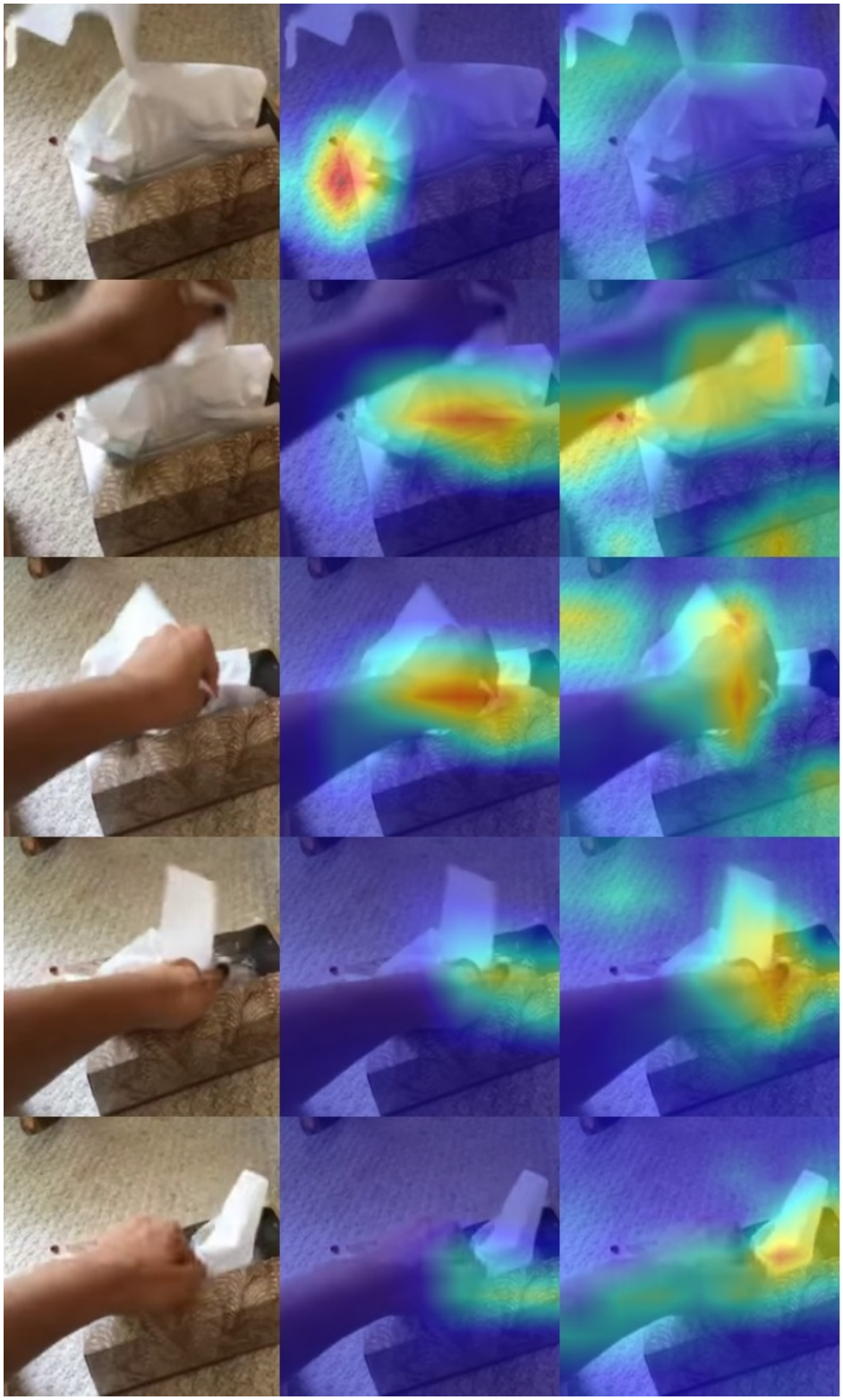


Figure 5.7: The videos shows a person stuffing tissue paper into the box. The model predicts the label for this video as *Stuffing something into something* with 94.95% confidence. However, the actual label for this video is *Putting something into something*. Visually, there is repetition when the paper is put into the box, and thus the model wins over the actual label.

However, the actual label is *putting something into something*. This is another case where the actual label is incorrect and the model is able to capture the true action and thus wins.

While the above two cases were only between the two classes, the next example is a little different. The video frames in figure 5.8 show a person putting in a number of objects one by one into a container. Each time the person puts an object into the container, the region around the container is activated and shown by the red heatmap values in the middle column (Slow Pathway). Additionally, when the hand moves towards the container, it is highlighted as seen in the third column (Fast Pathway).

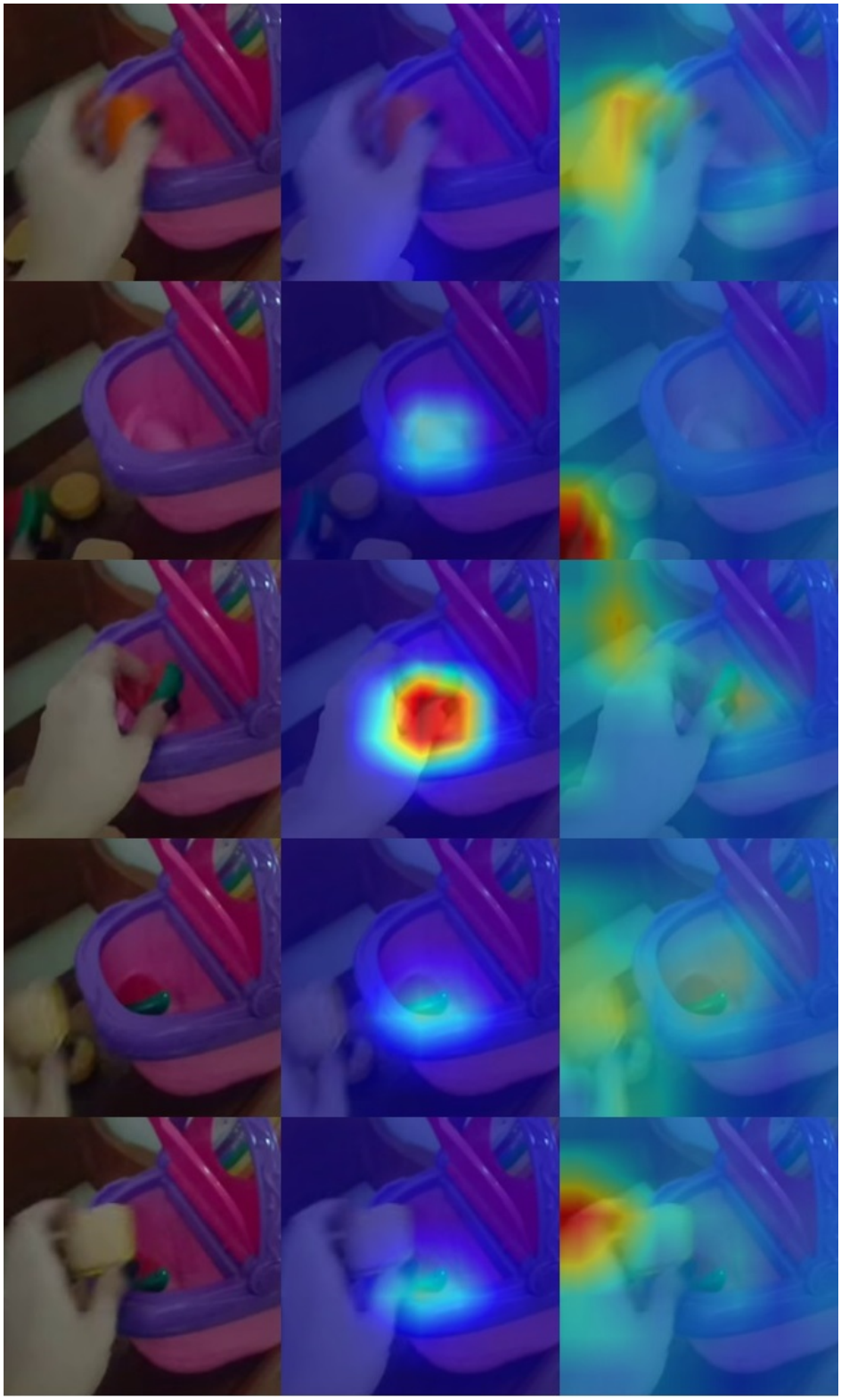


Figure 5.8: The model predicts the label for this video as *Putting number of something into something* with 74.14% confidence. However, the actual label for this video is *Putting something into something*. Since there is another more descriptive label in the label set, the model that label picks up. This also indicates the high overlap between labels in the dataset which also creates unnecessary errors for the network.

The top two predictions of the model are *Putting number of something onto something* with 74.14% confidence and *Stuffing something into something* with 24.79% confidence. These predictions can be corroborated by number of distinct objects being put into the container, as well as the repetition involved in performing the action, even when no obvious signs of struggle are seen during. The actual label in this case, however, is *Putting something into something*. On a high level, this label makes sense; however, even when the model picks up a more descriptive label from the label set, it is still deemed incorrect. In this case, the high overlap between between labels in the dataset is the reason for the unnecessary errors for the network.

5.3.3 Going deeper into the networks

In the previous section, I applied Grad-CAM on the final convolutional layer to get an insight into the salient regions of video frames as the action evolves. This technique can be extended to dive deeper into the network to see the gradual progression of what the network learns to focus on as it reaches the final layer. I present a layer wise Grad-CAM visualization of two videos, one for both SlowFast and TSM to study the change in their learning.

5.3.3.1 Visualizing TSM

Figure 5.9 illustrates the layer-wise visualization of a video labeled *Plugging something into something but removing it right as you remove your hand*. Each row represents the progression of frames into the action, while each column represents layer wise progression from left to right starting with the original image, layer 1, layer 2, layer 3 and then the final layer. We observe that as the layers get deeper, the saliency of the network narrows on the object that is interacted with. While in layer 3 the object saliency also includes the wire, it quickly disappears in the final layer.

In the first layer, almost everything except for the boundary around the arm is having positive values, but that reduces as the depth increases. It can be observed that in the middle row, the action being performed is at its peak, and thus the attention is very highly and positively localized around the plug point and the object. Unlike the previous case studies using SlowFast where one

pathway focused on the hand while the other focused on the object, TSM behaves more like the Slow pathway focusing on the object(s) in action.

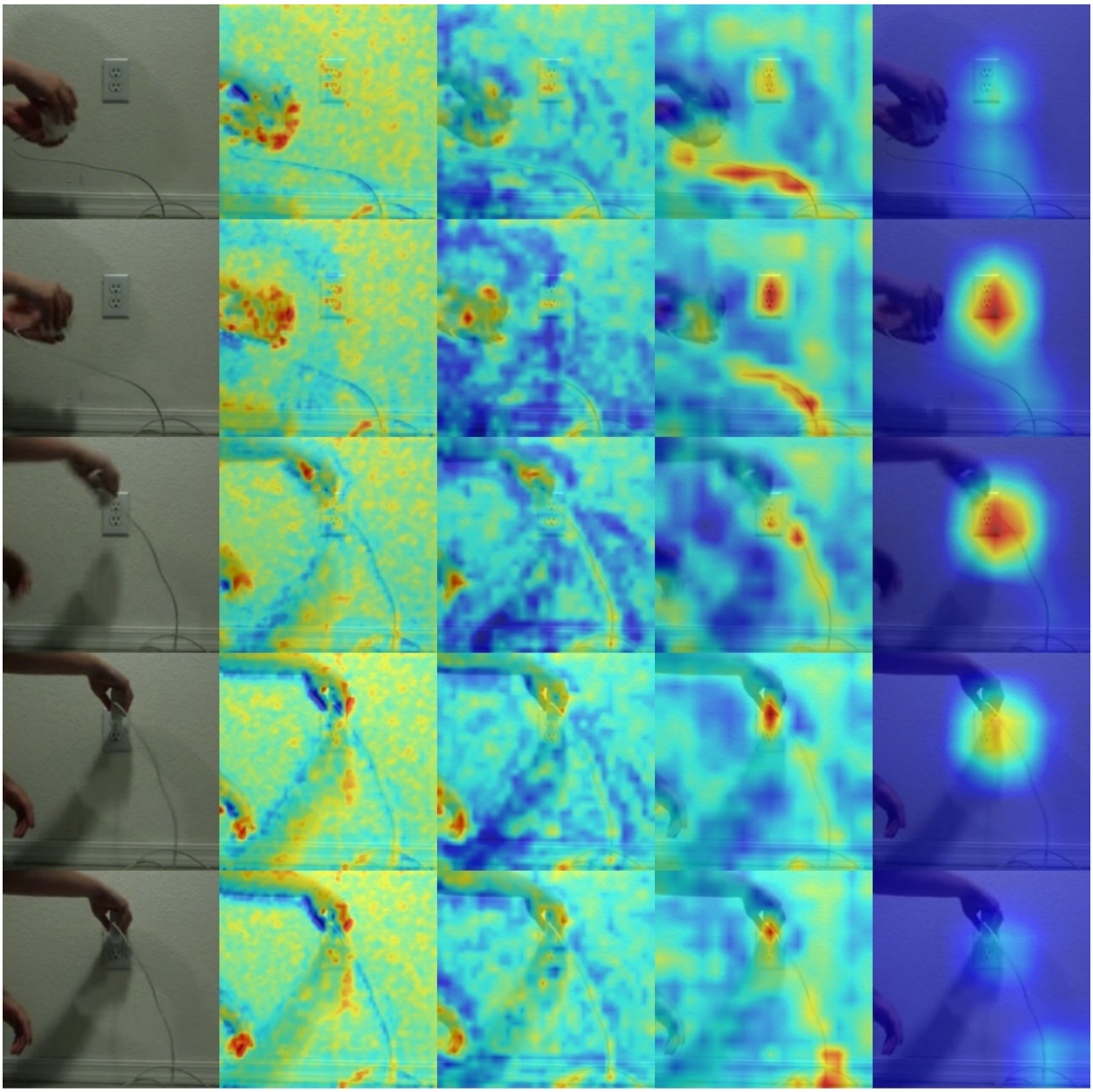


Figure 5.9: Visualizing the earlier layers of TSM. For an action label *Plugging something into something but removing it right as you remove your hand*, the network intially looks at the hand, but quickly changes the focus to the object as it goes deeper into the layers. The last layer shows highest saliency source as the target object.

5.3.3.2 Visualizing SlowFast

Figure 5.10 illustrates selected frames for the layer-wise visualization of Slow pathway for a video labeled as *moving something down*. Similar to the previous network, the Slow pathway focuses on the object being interacted with and activates the salient regions around it. As the video progresses going down the rows, we can see that the attention region around the bottle being moved down increases. It peaks around the middle of the duration of the video (roughly around 16-18th frame in a video with 32 frames). One very curious point to note is the blue blob that can be seen in layer 4 or the one before the last layer. It is unknown why the network has a strong blue color which disappears immediately in the next layer.

Figure 5.11 shows the visualization of the same action, but with the Fast pathway. Compared to the previous figure, this figure can be considered as a complement based on how the salient regions are highlighted. As can be seen in deeper layers, the focus is on the arm starting with the sleeve of the actor and continuing till the last layer only reducing in magnitude. A curious observation comparing the last layers of the Slow and Fast pathway show that the latter is able to predict the future position of the bottle immediately seen in the next frame for the former (Slow) pathway. This is because the Fast pathway sees 4 times more number of frames than the Slow pathway, and thus models this information in the activations. In the middle layer, we can see that there is a history of the previous position of the hand that is lingering in the current frame as the arm goes down. When the network is focusing on the arm, it starts with the boundary of the sleeve in the second frame and gradually ends up at the elbow in the last frame. Surprisingly, the blue 'ghost' blob continues to have a presence even in this pathway, and disappears in the last layer.

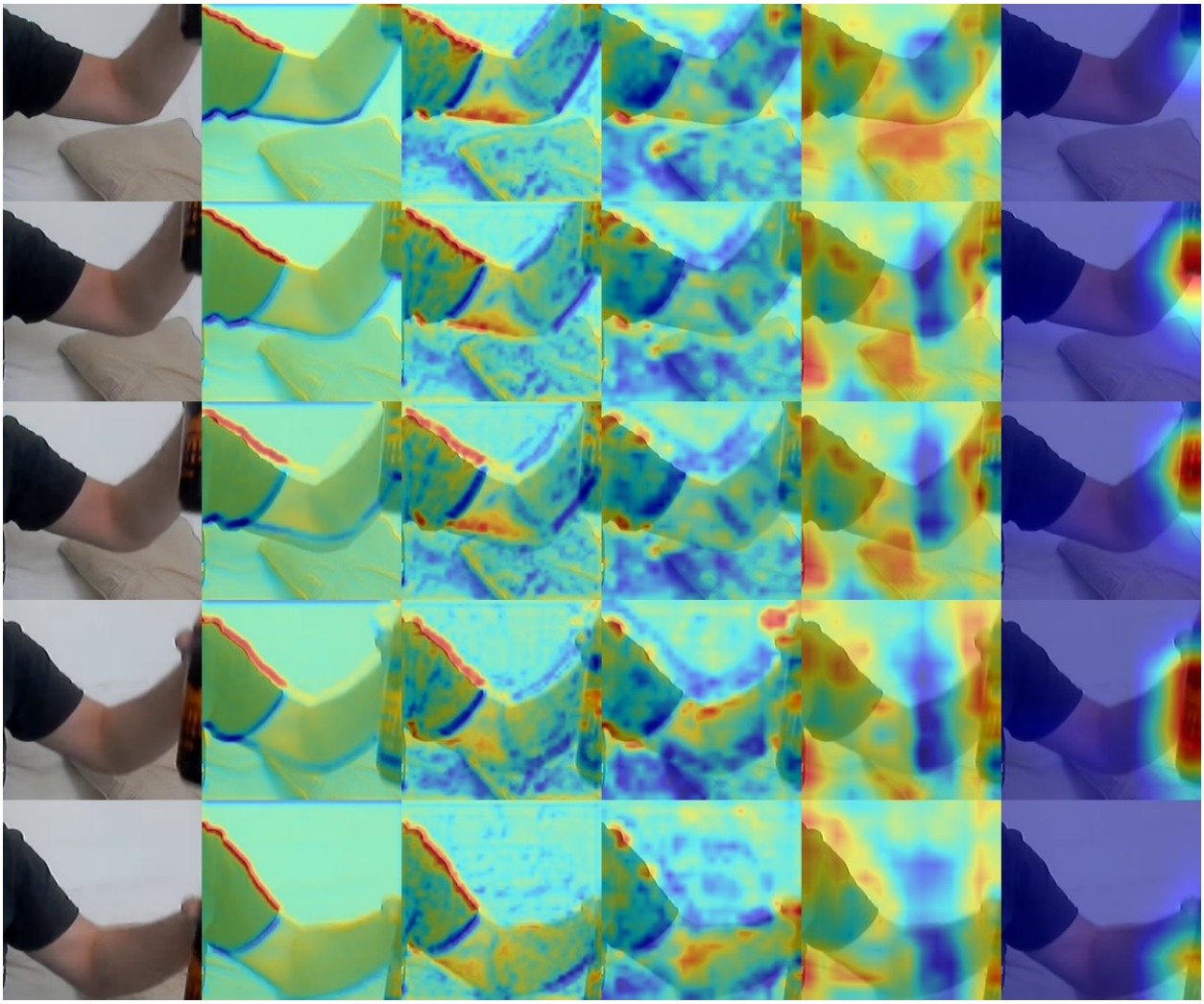


Figure 5.10: The layer visualization of the Slow pathway for *Moving something down*. The focus is on the object and the attention increases as the bottle moves downwards with the video progression.

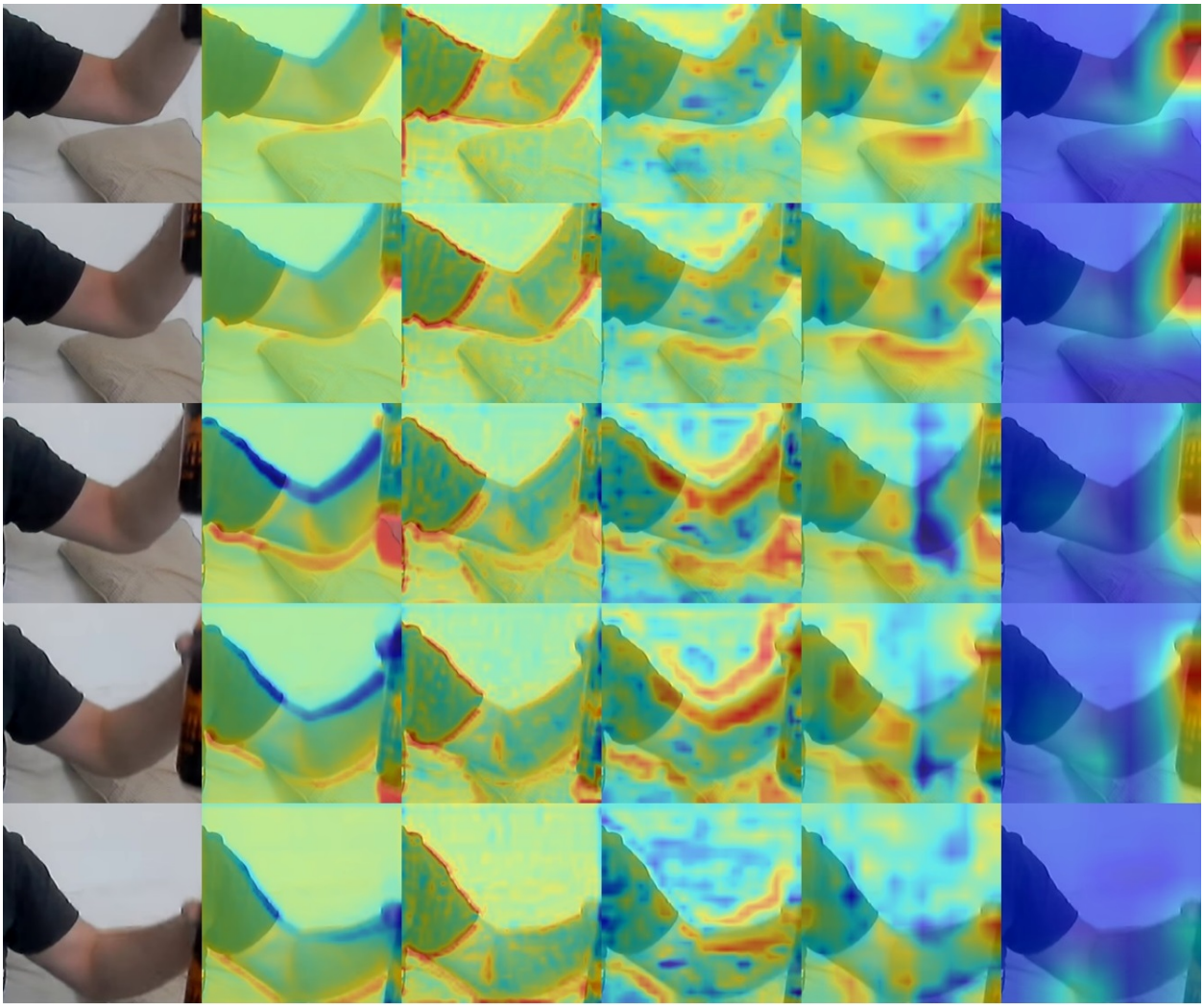


Figure 5.11: The layer visualization of the Fast pathway for *Moving something down*. The focus is on the arm starting with the sleeve of the actor and continuing till the last layer, only reducing in magnitude. The Fast pathway is able to predict the future position of the object, which makes sense given it has more expanse of the complete video fed to it as input.

Chapter 6

Conclusion and Future Work

6.1 Summary

Gestures or actions have a major influence in conveying or establishing context in a non-verbal communication. They differ from each other only in the range of their motions the former being more localized than the latter. These actions can be frozen in the moment into an image, or can progress more gracefully in a video. Classifying actions in a video is a topic that has long been studied in the domain of Computer Vision. Human Object Interaction, a sub-domain of video action recognition tasks, is challenged by the process of modeling how different objects interact with each other in the video in addition to modeling the temporal dependency between the video frames.

The current state of the art video action recognition networks are able to devise complex representations of videos and classify them with a high degree of confidence. These networks are trained to increase the confidence of the predicted class and to minimize the distance between the actual label and the predicted label. The manner of collection and curation of datasets in Computer Vision has followed a standarized protocol over the years. This involves collecting data from the internet or other sources and reviewing them using human labelers. The integrity of the data is maintained by posing the problem as a Yes or No question for annotators and consolidating the results. However, deviating from this protocol in pursuit of getting more real-life videos and a massive dataset in terms of samples and labels can result in inherent flaws in the data. This deviation can impact the networks in a negative way.

The labels in the Something Something dataset are of the form "Something action something" where *something* is a placeholder. There is a lot of overlap in the nature of the 174 class labels in the dataset with the use of different prepositions for the same action or a different label description of the same action. One way to address the loopholes in the performance of the existing models

is to reinspect the data and learn more perceptions about it. This work does not propose another state-of-the-art model to classify actions in videos. Instead, it attempts to address the shortcomings in the data and establish greater trust in the network's learning ability of different actions.

To investigate our hypothesis, we begin by examining the predictions and misclassifications of two popular video action recognition networks: SlowFast and TSM. We introduce a human-in-the-loop technique to review the videos given options describing the videos. The options available to the user for a video are obtained from the predictions of the networks. An initial pilot experiment is conducted using 1000 videos where each video is annotated by two users to study if our hypothesis holds true. As expected, 84% of samples selected contained multiple selections describing the videos. This validation permitted us to scale the experiment to include almost 11,857 videos in total. The reannotation of videos belonging to both top-1 and top-5 categories was crowd-sourced to Amazon Mturk. Each video is annotated by 3 workers to mitigate user bias originating from attempts to trick the system. An additional step of reviewing the workers' answers is facilitated by the ReviewMe tool. The tool is used to approve or reject the worker annotations which eventually pays them for their participation as well ensures an additional step of quality control check. A Majority Voting approach is applied to the answers thus obtained and approved from MTurk to decide if the labels can be included in the final reassessment. The resultant labels, called Reassessed Labels are used to re-evaluate the network performance in inference mode.

When the networks are re-evaluated in inference mode without any retraining, both networks see an increase of almost 12% top-1 accuracy and 3% top-5 accuracy. This staggering jump in the accuracy strongly implies that the models are already learning very relevant properties from the video needed to classify the actions in them. The correctness of the model predictions is further attested by the human-in-the-loop review system, where the options are the predictions from the two networks. However, the flaw in the dataset collection penalises the model predictions if the faulty expected label does not match the predicted label. Consequently to get higher validation accuracy, the newer architectures might be forced to learn erroneous properties of the data.

Additionally, in order to examine the internal workings of the models, we use Grad-CAM [25], a post-hoc attention approach to investigate the workings of an already trained neural network. This approach works by generating heat maps for the activations of different layers of the neural networks, and displays them onto the input image. As a result, the salient regions in the image that positively influence the prediction of a class can be visualized onto the image. Grad-CAM visualization is used as another route to validate my hypothesis that the models are learning relevant properties about the actions in videos. Some illustrations highlight the visualizations of the internal layers of the models based on what label is predicted as well as the actual label for the video. It can be inferred that the model can learn more verbose descriptions of the action as they are present in the label set. This in turn can be seen by the change in the salient regions with the evolution of the action. The network's learning abilities can be showcased as they go deeper to classify the video input.

6.2 Future Work

The research presented in this dissertation considers and proves the possibility that the networks are already doing a good job of classifying actions in the Something Something dataset. Although this dissertation focused on getting answers to bring the attention of the research community to the flaws in dataset curation steps, there are many extensions that can further this effort and establish some standardization of dataset quality.

As a proof of concept, this work considered only two networks for analysis. However, other more recent networks like Vision Transformers [28] were not in the scope of this research. There are key differences in how 3D CNNs and Transformers are able to model complexities in videos for classification. Investigating these differences can further help bolster the hypothesis through multiple ideas.

This work only handles re-annotating the validation data and re-evaluating the networks using the data. There are many videos where the main object interacted with is either off-center and closer to the edge of the video frame, or in such dark lighting conditions that it becomes impossible

to properly see what action is performed. The data preprocessing steps used for training include random cropping and resizing. These steps can completely eliminate the main object in the video frame; however, the training data in the Something Something dataset is massive with around 170,000 samples. The flaws that exist in the validation data can also be expected to be consistent in the training data. A continuation of this work can be extended to review the training data and compare the difference in performance reflected by the reviews.

This work uses Grad-CAM as a attempt to visualize the internal workings of the two networks. While Grad-CAM is effective at highlighting the salient regions in the frames, it can struggle when there are multiple occurrences of the same object in the video frame. Additionally, the approach struggles to highlight the complete object considered salient. There are other attempts like XGrad-CAM [31], Grad-CAM++ [30] that can render better visualizations.

A technique proposed in the original Grad-CAM paper called Guided Grad-CAM renders the exact features the networks focuses on when predicting a particular class in an image. Given time constraints, our work did not take the advantage of this approach to get better insights into the models. All Grad-CAM techniques mentioned above focus on getting the salient regions in an image, *i.e.* the spatial information. However, the temporal aspect of videos is rarely considered. The temporal Grad-CAM [29] focuses on understanding which frames in the video are most important for classification. This can help us get an insight into which sampling technique is most ideally suited for a network to maximize its learning. All the above ideas would be very compelling extensions of this work, and may inspire the community to trust their networks more than they do today.

Bibliography

[1] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li 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.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Alexei A Efros, Alexander C Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *null*, page 726. IEEE, 2003.

[4] Gioia Ballin, Matteo Munaro, and Emanuele Menegatti. Human action recognition from rgb-d frames based on real-time 3d optical flow estimation. In *Biologically Inspired Cognitive Architectures 2012*, pages 65–74. Springer, 2013.

[5] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR 2011*, pages 3169–3176. IEEE, 2011.

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

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

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[13] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[14] Joao Carreira and Andrew 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.

[15] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[16] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7083–7093, 2019.

[17] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2000–2009, 2019.

[18] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6202–6211, 2019.

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

[20] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[21] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[22] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[24] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

[25] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[26] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[27] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[28] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[29] Joonatan Manttari, Sofia Broomé, John Folkesson, and Hedvig Kjellstrom. Interpreting video features: A comparison of 3d convolutional networks and convolutional lstm networks. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[30] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[31] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. *arXiv preprint arXiv:2008.02312*, 2020.

[32] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005.

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

[34] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[35] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3376–3385, 2017.

[36] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018.

[38] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[39] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019.

[40] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. *arXiv preprint arXiv:1912.09930*, 2019.

[41] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 625–634, 2020.

[42] Victor Escorcia and Juan Niebles. Spatio-temporal human-object interactions for action recognition in videos. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 508–514, 2013.

[43] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 105–121, 2018.

[44] Yanli Ji, Yue Zhan, Yang Yang, Xing Xu, Fumin Shen, and Heng Tao Shen. A context knowledge map guided coarse-to-fine action recognition. *IEEE Transactions on Image Processing*, 2019.

[45] Brais Martinez, Davide Modolo, Yuanjun Xiong, and Joseph Tighe. Action recognition with spatial-temporal discriminative filter banks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[46] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[47] Tf-idf calculation. https://monkeylearn.com/blog/what-is-tf-idf.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[49] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[50] Glove blog calculation. https://www.mygreatlearning.com/blog/word-embedding.

[51] Zekun Yang, Noa Garcia, Chenhui Chu, Mayu Otani, Yuta Nakashima, and Haruo Takemura. Bert representations for video question answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1556–1565, 2020.

[52] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

[53] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*, 2020.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[55] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[56] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[57] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: From single to multi-labels, from global to localized labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2340–2350, June 2021.

[58] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[59] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly super-

vised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018.

[60] Kristof Meding, Luca M Schulze Buschoff, Robert Geirhos, and Felix A Wichmann. Imagenet suffers from dichotomous data difficulty. 2021.

[61] Raimonda Staniūtė and Dmitrij Šešok. A systematic literature review on image captioning. *Applied Sciences*, 9(10):2024, 2019.

[62] Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019.

[63] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[64] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[65] Manish Munikar, Sushil Shakya, and Aakash Shrestha. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5. IEEE, 2019.

[66] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps*, pages 323–350, 2018.

[67] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[68] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[69] JL Fleiss, B Levin, and MC Paik. Comparative studies: cross-sectional, naturalistic, or multinomial sampling. *Statistical Methods for Rates and Proportions. 3rd ed. Hoboken, NJ: J Wiley*, 2003.

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