VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

BTECH PROJECT

DEPARTMENT OF COMPUTER ENGINEERING & INFORMATION TECHNOLOGY

Dense Video Captioning

Ganadhish Acharekar
Akshat Shah
Arnav Shah
Saharsh Jain

Project Supervisor
Prof. Sandip T. Shingade

Abstract

The task of Dense Video Captioning (DVC), featured in the annual ActivityNet competition, aims to localize the different events in a given video and generate natural language descriptions for each of them. Most literature for this task involves utilizing CNN extracted video features and multiple stages of training. This project aims to utilize multiple modalities, namely audio and video, as opposed to previous single modality approaches, and we propose a model architecture that can be trained end-to-end. We utilize transformer architectures and efficient attention mechanisms for feature extraction, proposal generation as well as for captioning. We report xxxxx performance.

Contents

1 Introduction **9**
- 1.1 Problem Background 9
- 1.2 Problem Formulation 10
- 1.3 Objectives 11
- 1.4 Datasets 11
  - 1.4.1 Textually Annotated Cooking Scenes (TACoS) 11
  - 1.4.2 TACoS-MultiLevel 12
  - 1.4.3 Microsoft Video Description (MSVD) 12
  - 1.4.4 Montreal Video Annotation Dataset (M-VAD) 12
  - 1.4.5 MPII Movie Description Corpus (MPII-MD) 12
  - 1.4.6 MSR Video-to-Text (MSR-VTT) 12
  - 1.4.7 ActivityNet Captions 12
  - 1.4.8 ActivityNet Entities 13
  - 1.4.9 YouCook 13
  - 1.4.10 YouCook2 13
  - 1.4.11 MP-II Cooking 13
  - 1.4.12 VideoStory 13
  - 1.4.13 Charades 13
  - 1.4.14 Video Titles in the Wild (VTW) 14
  - 1.4.15 Kinetics 14
- 1.5 Evaluation Metrics 14
  - 1.5.1 BLEU: Bilingual Evaluation Understudy) 15
  - 1.5.2 ROUGE: Recall Oriented Understudy for Gisting Evaluation 15
  - 1.5.3 METEOR: Metric for Evaluation of Translation with Explicit Ordering 15
  - 1.5.4 CIDEr: Consensus based Image Description Evaluation 15
  - 1.5.5 WMD: Word Mover′s Distance 15
  - 1.5.6 SPICE: Semantic Propositional Image Captioning Evaluation 15

2 Literature Review **17**
- 2.1 Identified Themes 20
  - 2.1.1 Based on Training schemes 20
  - 2.1.2 Based on Modalities 20
  - 2.1.3 Based on Task ordering 20
- 2.2 Research Literature Gaps 20

3 Methodology **21**
  - 3.0.1 Multimodal Feature Extraction 21
    - Video Features 22
    - Audio Features 22
  - 3.0.2 Training 23
    - Knowledge Distillation 23

4 Feature Extraction **25**
4.1 Feature Extraction Methodology . . . 25
4.1.1 Video Features Extraction Using ViViT . . . 25
4.1.2 Audio Features Extraction using AST . . . 26
4.2 Temporal Sensitivity of Features . . . 27
4.2.1 Temporally Sensitive Pretraining of Proposed Feature Extractors . . . 28
Architecture . . . 28
Data Flow . . . 28
Loss functions . . . 29

A Supplementary Details **31**

B Paper Summaries **33**
B.1 Dense-Captioning Events in Videos . . . 33
B.1.1 Overview . . . 33
B.1.2 Datasets . . . 33
B.1.3 Performance . . . 33
B.1.4 Methodology . . . 33
Flow of pipeline . . . 33
B.1.5 Conclusion . . . 34
B.2 A Better Use of Audio-Visual Cues: Dense Video Captioning with Bi-modal Transformer . . . 35
B.2.1 Overview . . . 35
B.2.2 Datasets . . . 35
B.2.3 Performance . . . 35
B.2.4 Methodology . . . 35
Proposal Generation Module . . . 36
Captioning Module . . . 36
Training Procedure . . . 37
B.2.5 Conclusion . . . 37
B.3 Vision Transformer (ViT) - An image is worth 16x16 words: Transformers for image recognition at scale . . . 38
B.3.1 Overview . . . 38
B.3.2 Datasets . . . 38
B.3.3 Performance . . . 38
B.3.4 Methodology . . . 38
Input: Patch embeddings . . . 38
Transformer Encoder . . . 39
Training Procedure . . . 39
B.3.5 Conclusion . . . 40
B.4 AST: Audio Spectrogram Transformer . . . 41
B.4.1 Overview . . . 41
B.4.2 Datasets . . . 41
B.4.3 Methodology . . . 41
B.4.4 Training procedure . . . 42
B.4.5 Conclusion . . . 42
B.5 ViViT: A Video Vision Transformer . . . 43
B.5.1 Overview . . . 43
B.5.2 Datasets . . . 43
B.5.3 Performance . . . 43
B.5.4 Methodology . . . 43
Input: Token embeddings . . . 43
Transformer Encoder . . . 44
Training Procedure . . . 45
B.5.5 Conclusion . . . 45
B.6 Temporally Sensitive Pretraining of Video Encoders for Localization Tasks . . . 46
B.6.1 Overview . . . 46
B.6.2 Contributions and Findings . . . 46

B.6.3 Pre-training Methodology . . . 46
B.6.4 Local and Global Feature Encoding . . . 46
B.6.5 Performance . . . 47
B.6.6 Conclusion . . . 47
B.7 End-to-end Dense Video Captioning with Parallel Decoding . . . 48
B.7.1 Overview . . . 48
B.7.2 Limitations Identified . . . 48
Two-stage design . . . 48
Unreliable Event Count Estimation . . . 48
Design of Proposal Generators . . . 48
B.7.3 Proposed Solution . . . 48
Feature Encoding . . . 49
Parallel Decoding . . . 49
Localization Head . . . 49
Captioning Head . . . 49
Event Counter . . . 50
Set Prediction Loss . . . 50
B.8 End-to-End Object Detection with Transformers (DETR) . . . 51
B.8.1 Overview . . . 51
B.8.2 DETR Model . . . 51
Object detection set prediction loss . . . 51
DETR architecture . . . 51
Backbone . . . 51
Transformer encoder . . . 52
Transformer decoder . . . 52
Prediction feed-forward networks . . . 52
B.8.3 Conclusion . . . 52
B.9 Deformable DETR: Deformable Transformers for End-to-End Object Detection . . . 53
B.9.1 Overview . . . 53
B.9.2 Architecture . . . 53
Deformable Attention Module . . . 53
Multi-scale Deformable Attention Module . . . 54
Deformable Transformer Encoder . . . 54
Deformable Transformer Decoder . . . 54
B.9.3 Conclusion . . . 54
B.10 Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity . . . 55
B.10.1 Overview . . . 55
B.10.2 Methodology . . . 55
Encoder Token Sparsification . . . 55
Decoder Cross-Attention Map . . . 55
Encoder Auxiliary Loss . . . 56
B.10.3 Conclusion . . . 56

Chapter 1

Introduction

Machine learning is a subfield of *Artificial Intelligence*, which involves algorithms that enable automatic learning of tasks based on data. Unlike classical programming, which involves giving instructions and input data to a computer and producing output data, machine learning aims to produce the instructions or rules used for a task as the result, when the input and corresponding output data is provided to a computer.

Deep Learning, a subfield of Machine Learning has been growing popular in recent years, owing to their performance over traditional machine learning techniques without the need for intensive feature engineering by practitioners; deep learning models involve building up successive layers of increasingly meaningful representations automatically, allowing them to be rich and expressive with information. Different forms of deep learning models, including dense or fully connected layers, convolutional networks, recurrent networks and others have been applied to a variety of tasks which traditional algorithms have failed to address.

Convolutional networks (CNNs) have dominated the field of computer vision for quite some time. Convolutional networks consist of convolutional layers, pooling layers and fully connected layers. Out of these, convolutional layers are the core building blocks of CNNs, which apply a series of different *image filters* or *convolutional kernels* to an input image in a convolution operation and the outputs of different filters at a layer are stacked together. The filter weights are learned, so as to learn to extract features for the given task. CNNs were introduced mainly because of the problems with applying multi-layer perceptron networks to image data; MLPs ignore the spatial structure of the image, and fully connected layers have too many parameters. In contrast, CNNs take multiple dimensional inputs and utilize the spatial structures of an image, and have sparse layers with localized connections instead of global connections. Moreover, all the parameters for an individual filter are shared across all nodes.

Transformers were introduced by Vaswani *et al* in [1]. The transformer architecture is shown in Figure 1.2. It consists of encoder layers which use *multi-head self-attention* on input tokens followed by fully connected layers. The decoder layers use the output tokens and representations from encoder layers in a *multi-head cross-attention* block followed by fully connected layers. This architecture has been adapted by various other models to serve a variety of tasks. Transformers are gaining traction in a variety of tasks, including sequence to sequence tasks like machine translation, natural language generation and understanding, and recently, computer vision.

Key Terms: Dense Video Captioning, Transformers, Temporally Sensitive Pretraining, Sparse Deformable Attention, Context

1.1 Problem Background

There has been a significant progress in the fields of *Computer Vision* (CV) and *Natural Language Processing* (NLP) and other artificial intelligence problems in recent decades. In computer vision, tasks like object detection and classification are well-explored, while in natural language processing, sequence to sequence tasks like machine translation have been well worked upon. The most promising solutions to these tasks come from deep learning methodologies, outperforming techniques like rule-based systems and other traditional machine learning algorithms. Owing to the rich feature representation capabilities of different kinds of neural networks, tasks that are more natural to humans are becoming solvable by machines, especially perceptive and understanding tasks.

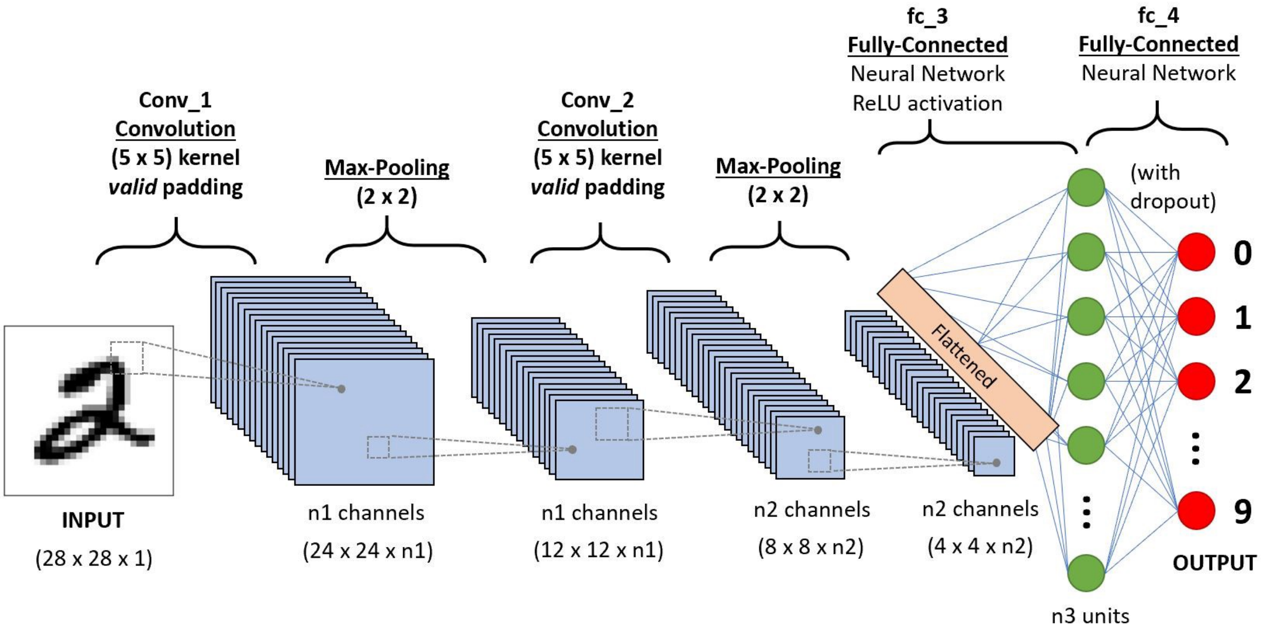

fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
OUTPUT
n3 units

Figure 1.1: A Convolutional Network

The task of image captioning combines the fields of CV and NLP. Video action recognition, which involves classifying the action in a short video clip, is a task that can be considered as the representative task for video understanding. Taking one step further from action recognition and image captioning, the task of video captioning involves labelling a video clip with a single natural language sentence. However, for long videos, a single sentence is not enough to describe all events that may occur in a video. Thus, the task of **Dense Video Captioning (DVC)** was introduced [2]. It involves generating natural language descriptions for each of the multiple events that may occur in a video.

Describing a short video in natural language is trivial for humans, but a challenging task for machines. Automatic video description involves understanding of many entities and the detection of their occurrences in a video employing computer vision techniques. These entities include background scene, humans, objects, human actions, human-object interactions, human-human interactions, other events, and the order in which events occur. All this information should then be articulated using comprehensible and grammatically correct text, employing NLP techniques[3]. Indeed, the task of dense video captioning can be thought of as the culmination of all the perceptive learning tasks of computer vision and the natural language generation task.

As we move towards a digital age, more and more content is generated in the form of multimedia, e.g., videos. The applications of dense video captioning are in tasks such as:

- Video summarization and highlights generation, e.g. in fields such as sports and entertainment
- Video retrieval via search and indexing on its dense captions. This can help in a multitude of areas such as experiments, surveillance and security, sports, education, et cetera.
- Video segment localization queries can be better handled by using dense event captions generated on a bulk of videos. Indeed, this can open the door to automatic interpretation of videos, a kind of data that is treated as unstructured binary data today.
- Increased accessibility to the visually impaired can be achieved with the combined use of DVC and the much more mature task of text-to-speech (TTS).

1.2 Problem Formulation

The task of dense video captioning involves two sub-tasks (1) temporal localization of events in a video and (2) describing the localized events in natural language. Given an input video $v = \{v_1, v_2, ..., v_T\}$ where v_i

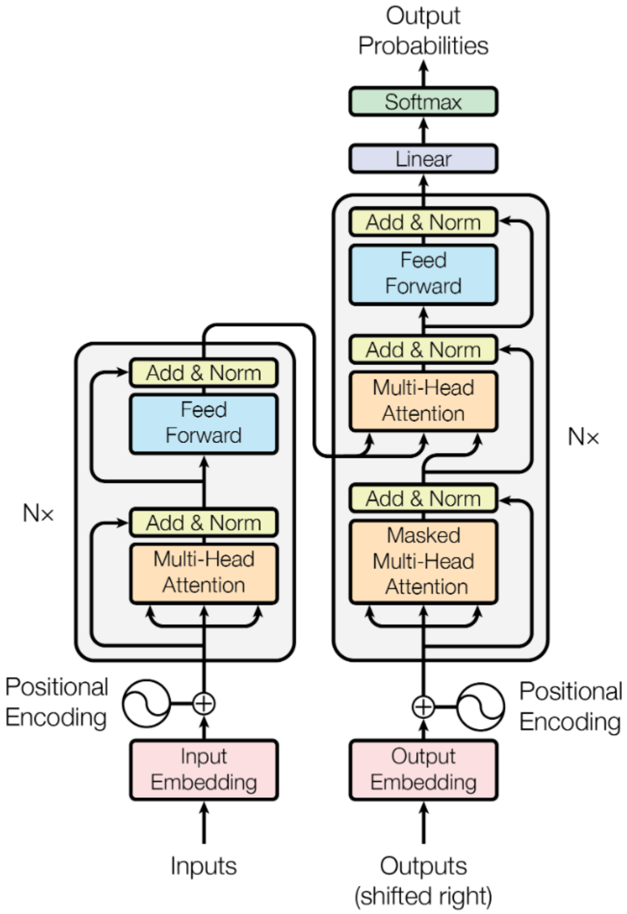

Output
Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figure 1.2: The Transformer architecture. Image courtesy [1]

represents i^{th} video frame in temporal order, the target of dense video captioning task is to output a set of sentences $S = \{s_1, s_2, ..., s_{N_s}\}$ where N_S is the number of sentences and $s_i = \{t_i^{start}, t_i^{end}, \{w_j\}\}$ consists of start and end timestamps for each sentence described with set of words from a vocabulary set $w_j \in V$.

Most of the architectures for dense video captioning comprises two components (1) Proposal generation module and (2) Captioning module. The modules can be trained in different ways like separate training, alternative training or in a end-to-end manner. The task of the proposal generation module is to take as input video frames v and output event proposals $P = \{t_i^{start}, t_i^{end}, confidence_i\}$. Depending on the architecture, the proposal generation module can also output additional parameters like in [2, 4, 5] which can be utilized by captioning module. The proposals can also be in the form of offsets (center and length) as in [4, 6, 7, 8, 9]. The captioning module outputs descriptions for each of the proposed event in usually a single sentence.

1.3 Objectives

-
-

1.4 Datasets

1.4.1 Textually Annotated Cooking Scenes (TACoS)

It is a subset of MP-II Composites. TACoS was further processed to provide coherent textual descriptions for high quality videos. The TACoS dataset was constructed by filtering through MP-II Composites, while restricting to only those activities that involve manipulation of cooking ingredients, and has at least 4 videos for the same activity. As a result, TACoS contains 26 fine grained cooking activities in 127 videos. For each video, 20 different textual descriptions were collected. The dataset comprises 11,796 sentences containing 17,334 actions descriptions. The dataset also provides the alignment of sentences describing activities by obtaining approximate time stamps where each activity starts and ends.

1.4.2 TACoS-MultiLevel

It was collected on the TACoS corpus. For each video in the TACoS corpus, three levels of descriptions were collected that include: (1) detailed description of video with no more than 15 sentences per video; (2) a short description that comprises 3-5 sentences per video; and finally (3) a single sentence description of the video. Annotation of the data is provided in the form of tuples such as object, activity, tool, source and target with a person always being the subject.

1.4.3 Microsoft Video Description (MSVD)

It comprises 1,970 YouTube clips with human annotated sentences. The audio is muted in all clips to avoid bias from lexical choices in the descriptions. Furthermore, videos containing subtitles or overlaid text were removed during the quality control process of the dataset formulation. The duration of each video in this dataset is typically between 10 to 25 seconds mainly showing one activity. The dataset comprises multilingual (such as Chinese, English, German etc) human generated descriptions.

1.4.4 Montreal Video Annotation Dataset (M-VAD)

It is based on the Descriptive Video Service (DVS) and contains 48,986 video clips from 92 different movies. Each clip is spanned over 6.2 seconds on average and the entire time for the complete dataset is 84.6 hours. The total number of sentences is 55,904, with few clips associated with more than one sentence. The vocabulary of the dataset spans about 17,609 words (Nouns-9,512: Verbs-2,571: Adjectives-3,560: Adverbs-857). The dataset split consists of 38,949, 4,888 and 5,149 video clips for training, validation and testing respectively.

1.4.5 MPII Movie Description Corpus (MPII-MD)

It contains transcribed audio descriptions extracted from 94 Hollywood movies. These movies are subdivided into 68,337 clips with an average length of 3.9 seconds paired with 68,375 sentences amounting to almost one sentence per clip. Every clip is paired with one sentence that is extracted from the script of the movie and the audio description data. The Audio Descriptions (ADs) were collected first by retrieving the audio streams from the movie using online services MakeMkV 1 and Subtitle Edit 2. Then the transcribed texts were aligned with associated spoken sentences using their time stamps. The total time span of the dataset videos is almost 73.6 hours and the vocabulary size is 653,467.

1.4.6 MSR Video-to-Text (MSR-VTT)

It contains a wide variety of open domain videos for video captioning tasks. It comprises 7180 videos subdivided into 10,000 clips. The clips are grouped into 20 different categories.. The dataset is divided into 6513 training, 497 validation and 2990 test videos. Each video comprises 20 reference captions annotated by AMT workers. In terms of the number of clips with multiple associated sentences, this is one of the largest video captioning datasets. In addition to video content, this dataset also contains audio information that can potentially be used for multimodal research.

1.4.7 ActivityNet Captions

It contains 100k dense natural language descriptions of about 20k videos from ActivityNet that correspond to approximately 849 hours. On average, each description is composed of 13.48 words and covers about 36 seconds of video. There are multiple descriptions for every video and when combined together, these descriptions cover 94.6% content present in the entire video. In addition, 10% temporal overlap makes the dataset especially interesting and challenging for studying multiple events occurring at the same time.

1.4.8 ActivityNet Entities

It is the first video dataset with entities grounding and annotations. This dataset is built on the training and validation splits of the ActivityNet Captions dataset, but with different captions. In this dataset, noun phrases (NPs) of video descriptions have been grounded to bounding boxes in the video frames. The dataset comprises 14281 annotated videos, 52k video segments with at least one noun phrase annotated per segment and 158k bounding boxes with annotations. The dataset employs a training set (10k) similar to ActivityNet Captions. However, the validation set of ActivityNet Captions is randomly and evenly split into ANet-Entities validation (2.5k) and testing (2.5k) sets.

1.4.9 YouCook

It consists of 88 YouTube cooking videos of different people cooking various recipes. The background (kitchen / scene) is different in most of the videos. The dataset is divided into six different cooking styles, for example grilling, baking etc. For machine learning, the training set contains 49 videos and the test set contains 39 videos. The object categories for the dataset include “utensils”, “bowls” and “food” etc.

1.4.10 YouCook2

YouCook-II Dataset consists of 2000 videos uniformly distributed over 89 recipes. The cooking videos are sourced from YouTube and offer all challenges of open domain videos such as variations in camera position, camera motion and changing backgrounds. The complete dataset spans a total play time of 175.6 hrs and has a vocabulary of 2600 words. The videos are further divided into 3-16 segments per video with an average of 7.7 segments per video elaborating procedural steps. Individual segment length varies from 1 to 264 seconds. The average length of each video is 316 seconds reaching up to a maximum of 600 seconds. The dataset is randomly split into train, validation and test sets with the ratio of 66

1.4.11 MP-II Cooking

Max Planck Institute for Informatics (MP-II) Cooking dataset comprises 65 fine grained cooking activities, performed by 12 participants preparing 14 dishes such as fruit salad and cake etc. The 65 cooking activities include “wash hands”, “put in bowl”, “cut apart”, “take out from drawer” etc. When the person is not in the scene for 30 frames (one second) or is performing an activity that is not annotated, a “background activity” is generated. In total, the dataset comprises 44 videos (888,775 frames), with an average length per clip of approximately 600 seconds. The dataset spans a total of 8 hours play length for all videos, and 5,609 annotations.

1.4.12 VideoStory

VideoStory is a multi sentence description dataset comprising 20k social media videos. This dataset is aimed to address the story narration or description generation of long videos that may not sufficiently be illustrated with a single sentence. Each video is paired with at least one paragraph. The average number of temporally localized sentences per paragraph are 4.67. There are a total of 26245 paragraphs in the dataset comprising 123k sentences with an average of 13.32 words per sentence. On average, each paragraph covers 96.7% of video content. The dataset contains about 22% temporal overlap between co-occurring events. The dataset has training, validation and test split of 17908, 999, and 1011 videos respectively and also proposes a blind test set comprising 1039 videos.

1.4.13 Charades

It contains 9848 videos of daily indoor household activities. Videos are recorded in 15 different indoor scenes and restricted to use 46 objects and 157 action classes only. The dataset comprises 66500 annotations describing 157 actions. It also provides 41104 labels to its 46 object classes. Moreover, it contains 27847 descriptions covering all the videos. The videos in the dataset depict daily life activities with an average duration of 30 seconds. The dataset is split into 7985 and 1863 videos for training and test purposes respectively.

1.4.14 Video Titles in the Wild (VTW)

It contains 18100 video clips with an average of 1.5 minutes duration per clip. Each clip is described with one sentence only. However, it incorporates a diverse vocabulary, where on average one word appears in not more than two sentences across the whole dataset. Besides the single sentence per video, the dataset also provides accompanying descriptions (known as augmented sentences) that describe information not present in the visual content of the clip. The dataset is proposed for video title generation as opposed to video content description but can also be used for language-level understanding tasks including video question answering.

1.4.15 Kinetics

A collection of large-scale, high-quality datasets of URL links of up to 650,000 video clips that cover 400/600/700 human action classes, depending on the dataset version. The videos include human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. Each action class has at least 400/600/700 video clips. Each clip is human annotated with a single action class and lasts around 10 seconds.

Dataset	**Domain**	**#classes**	**#videos**	**Avg len**	**#clips**	**#sent**	**#words**	**vocab**	**len (hrs**
TACoS	cooking	26	127	360 sec	7,206	18,227	146,771	28,292	15.9
TACos Multilevel	cooking	1	185	360 sec	14,105	52,593	2,000	-	27.1
MSVD	open	218	1970	10 sec	1,970	70,028	607,339	13,010	5.3
M-VAD	movie	-	92	6.2 sec	48,986	55,904	519,933	17,609	84.6
MPII-MD	movie	-	94	3.9 sec	68,337	68,375	653,467	24,549	73.6
MSR-VTT	open	20	7,180	20 sec	10,000	200,000	1,856,523	29,316	41.2
ActivityNet Captions	open	-	20,000	180 sec	-	100,000	1,348,000	-	849.0
ActivityNet Entities	social media	-	14,281	180 sec	52k	-	-	-	-
YouCook	cooking	6	88	-	Nil	2,688	42,457	2,711	2.3
YouCook II	cooking	89	2,000	316 sec	15.4k	15.4k	-	2,600	176.0
MP-II Cooking	cooking	65	44	600 sec	-	5,609	-	-	8.0
VideoStory	social media	-	20k	-	123k	123k	-	-	396.0
Charades	human	157	9,848	30 sec	-	27,847	-	-	82.01
VTW	open	-	18,100	90 sec	-	44,613	-	-	213.2
Kinetics 700	human	700	6,50,000	10 sec	700	-	-	-	1806

Table 1.1: Datasets and their characteristics

1.5 Evaluation Metrics

Text Generation is a tricky domain. Academics as well as the industry still struggle for relevant metrics for evaluation of the generative models' qualities. Every generative task is different, having its own subtleties and peculiarities.These metrics can be applied to the numerous tasks such as:

- Short or long-form text generation
- Machine Translation
- Summarisation
- Chatbots and dialogue systems
- Multimedia systems like speech2text, image/video captioning

Following are certain metrics that are used to evaluate the above natural langauge generation tasks:

1.5.1 BLEU: Bilingual Evaluation Understudy)

BLEU [10] is a precision focused metric that calculates n-gram overlap of the target and generated texts. This n-gram overlap means that this evaluation scheme is word-position independent apart from n-grams' term associations. BLEU also consists of a brevity penalty i.e. a penalty applied when the generated text is too small compared to the target text.

1.5.2 ROUGE: Recall Oriented Understudy for Gisting Evaluation

ROUGE [11] is a set of metrics for evaluating automatic summarization of texts as well as machine translations. There are 3 types of ROUGE: ROUGE-N, the most common ROUGE type which means n-gram overlap. Second is ROUGE-L which checks for Longest Common Subsequence instead of n-gram overlap. The third is ROUGE-S which focuses on skip grams.

1.5.3 METEOR: Metric for Evaluation of Translation with Explicit Ordering

METEOR [12] is an metric that works on word alignments. It computes one to one mapping of words in generated and reference texts. Traditionally, it uses WordNet or porter stemmer. Finally, it computes an F-score based on these mappings. The metric was designed to fix some of the problems found in the more popular BLEU metric, and also produce good correlation with human judgement at the sentence or segment level. This differs from the BLEU metric in that BLEU seeks correlation at the corpus level.

1.5.4 CIDEr: Consensus based Image Description Evaluation

CIDEr [13] is a metric used to evaluate image descriptions that uses human consensus. It uses a triplet-based method of collecting human annotations to measure consensus followed by stemming and grouping the words into n-grams. This metric is especially useful in image and video annoated tasks.

1.5.5 WMD: Word Mover's Distance

WMD [14] is a fundamental technique for measuring the similarity of two documents. As the crux of WMD, it can take advantage of the underlying geometry of the word space by employing an optimal transport formulation. WMD leverages the results of advanced embedding techniques like word2vec and Glove. It suggests that distances between embedded word vectors are to some degree, semantically meaningful. It utilizes this property of word vector embeddings and treats text documents as a weighted point cloud of embedded words.

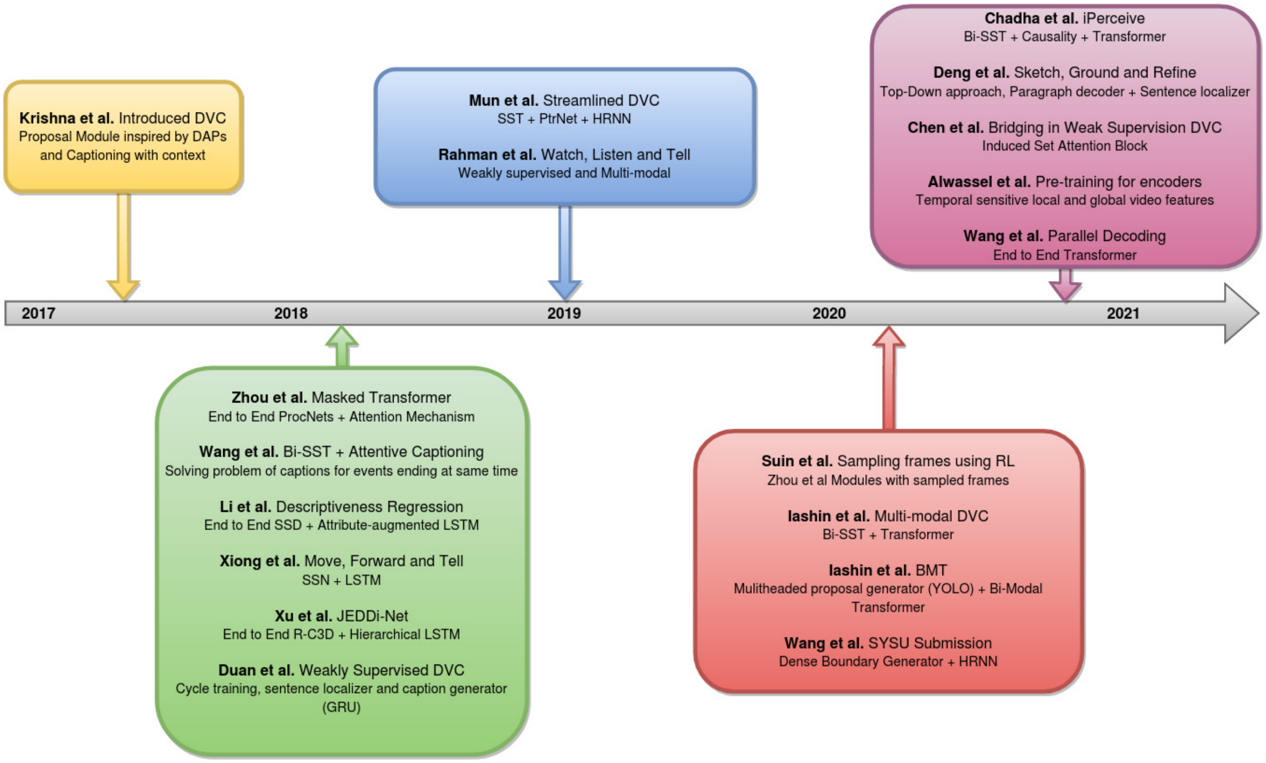
1.5.6 SPICE: Semantic Propositional Image Captioning Evaluation

SPICE [15] is an automated caption evaluation metric defined over scene graphs. It first transforms both generated and target captions into an intermediate representation that encodes semantic propositional content. It then creates a scene graph based on certain object classes, attribute types and relations which in turn is used to calculate the F-score b/w the generated and target captions.

Chapter 2

Literature Review

The problem of dense video captioning was introduced by Krishna *et al* in [2] by proposing ActivityNet Captions dataset. Their architecture involved a proposal module and captioning module. The proposal module was inspired from DAPs [16], while the captioning module incorporated LSTM with contextual features along with event feature as inputs. The architecture was able to detect events and generate captions of the video in a single pass without the need of a time consuming sliding window approach. However, since the features were dependent on the end location of the event, the model generated same captions for events ending at same timestamp.



Krishna et al. Introduced DVC
Proposal Module inspired by DAPs and Captioning with context
Mun et al. Streamlined DVC
SST + PtrNet + HRNN
Rahman et al. Watch, Listen and Tell
Weakly supervised and Multi-modal
Chadha et al. iPerceive
Bi-SST + Causality + Transformer
Deng et al. Sketch, Ground and Refine
Top-Down approach, Paragraph decoder + Sentence localizer
Chen et al. Bridging in Weak Supervision DVC
Induced Set Attention Block
Alwassel et al. Pre-training for encoders
Temporal sensitive local and global video features
Wang et al. Parallel Decoding
End to End Transformer
2017
2018
2019
2020
2021
Zhou et al. Masked Transformer
End to End ProcNets + Attention Mechanism
Wang et al. Bi-SST + Attentive Captioning
Solving problem of captions for events ending at same time
Li et al. Descriptiveness Regression
End to End SSD + Attribute-augmented LSTM
Xiong et al. Move, Forward and Tell
SSN + LSTM
Xu et al. JEDDi-Net
End to End R-C3D + Hierarchical LSTM
Duan et al. Weakly Supervised DVC
Cycle training, sentence localizer and caption generator (GRU)
Suin et al. Sampling frames using RL
Zhou et al Modules with sampled frames
Iashin et al. Multi-modal DVC
Bi-SST + Transformer
Iashin et al. BMT
Mulitheaded proposal generator (YOLO) + Bi-Modal Transformer
Wang et al. SYSU Submission
Dense Boundary Generator + HRNN

Figure 2.1: Evolution of dense video captioning methods over time.

Following the release of ActivityNet Captions dataset in 2017, many researchers were able to surpass the results of baseline model and achieve state-of-the-art. Zhou *et al* [9] addressed the problem of little influence of language desciptions on event proposals if the two modules are trained separately. They introduced an end-to-end masked transformer for propogating captioning error to proposal module for better performance. Furthermore, they proposed a self-attention mechanism for learning long-range dependencies in video. The proposal module was based on ProcNets [17]. The caption decoder employed a differentiable proposal mask to account for features in the respective event. Wang *et al* [5] employed Bi-SST as their proposal generator to account for past and future context information. The captioning module consisted of attentive fusion of context features, the weights of which were decided by a context gating mechanism. The architecture selected final captions based on joint ranking method which accounted for both proposal and caption confidence. Li *et al* [4], inspired by object localization networks like [18, 19], presented an end-to-end model with descriptiveness regression. An attribute-augmented LSTM network optimized using reinforcement learning was used for captioning module. Xiong *et al* [20] strived to generate relevant, coherent and concise descriptions using SSN [21] for event localization and LSTM for event selection and caption generation. Reinforcement learning with sentence-level reward is used to train the captioning network. Xu *et al* [7] proposed JEDDi-Net, an end-to-end architecture incorporating visual and language contexts. Segment Proposal Network inspired from R-C3D [22] is used for proposal generation. Hierarchical LSTM with caption-level controller network and word-level sentence decoder is used for caption generation. The proposal features are represented using 3D Segment-of-Interest Pooling. Duan *et al* [23] introduced weak supervision training with no need of temporal annotations of events. They trained sentence localizer and caption generator in cyclic manner minimizing the reconstruction loss. However, the method struggles to detect beginning of events properly.

Mun *et al* [24] tackles the challenge of coherent captioning by considering temporal dependency between events. They used SST for event proposals, PtrNet for event sequence generation and HRNN for captioning. Rahman *et al* [25] utilized weak supervision method from [23] and were the first to try multi-modal approach for dense video captioning. They show how audio alone can be competitive to the previous visual based results. The paper also discusses various methods to encode audio (MFCC, CQT, SoundNet) and context fusion techniques for multiple modalities (Multiplicative Mixture, Multi-modal context fusion, MUTAN). The architecture suffered from proposal localization accuracy due to weak supervision. Furthermore, the results are also affected due to unavailability of part of the dataset as some videos are not available on their respective urls.

Suin *et al* [26] aimed to reduce computational cost by processing fewer frames. They used deep reinforcement-based approach to describe multiple events in a video by watching a portion of the frames. The event proposal and captioning modules were inspired from [9]. Iashin *et al* [27] shows the importance of audio and speech modalities alongside visual features for dense video captioning task. They employ a Bi-SST for proposal and the captioning module consists of a Transformer architecture with three blocks: an encoder, decoder and generator. The model was able to achieve better performance than the then existing methods despite of unavailability of full dataset for multiple modalities. Iashin *et al* [6] utilized audio and video with Bi-modal Transformer for captioning. The proposal generator consisted of multiheaded method, inspired from YOLO object detector [28]. Their ablation analysis depicted stronger contribution of visual cues alone than audio cues alone. However, both modalities combined gave better results. Wang *et al* [29] adapt DBG [30] for temporal event proposals alongwith ESGN [24] for candidate subset selection. The captioning module consists of an encoder-decoder architecture with CMG (cross-modal gating) block to adaptively balance the visual and linguistic information.

Chadha *et al* [31] proposed to handle cognitive error (causality between events) and incorrect attention (attending to objects in the frame). Their end-to-end model consisted of Bi-SST for proposals, Common-Sense Reasoning for causality learning and Transformer based architecture [27] for captioning. The common-sense reasonining module employed a borrow-put experiment for determining dependency between events and generate context-aware features. Deng *et al* [8] introduced a top-down approach, reversing the usual detect-then-describe method. The architecture first generates a multi-sentence paragraph to describe the whole video and then localize each sentence for events. The captions are then refined using dual-path cross attention module. The top-down approach increased coherency in captions. Chen *et al* [32] worked on closely bridging event localization and captioning modules for weakly supervised learning. The Induced Set Attention Block helps the captioner to learn highly abstracted global structure of the video. The method suffers from detecting visually small concepts/objects. Alwassel *et al* [33] introduce a supervised pre-training paradigm for temporal action localization. The paradigm also considers background clips and global video information, to improve temporal sensitivity. Wang *et al* [34] formulated dense video captioning as a set prediction task and introduced an end-to-end dense video captioning framework with parallel decoding (PDVC). PDVC adopts the vision transformer to learn attentive interaction of different frames. Two prediction heads run in parallel over query features, leveraging the mutual benefits between two tasks of localization and captioning.

Paper	Set	Ground Truth Proposals				Learnt Proposals			
		M	**C**	**B@3**	**B@4**	**M**	**C**	**B@3**	**B@4**
Krishna *et al* [2] DCE	Validation	8.88	25.12	4.09	1.60	5.69	12.43	1.90	0.71
	Test	9.46	24.56	7.12	3.98	4.82	17.29	3.86	2.20
Zhou *et al* [9] Masked Transformer	Validation	11.16	47.71	5.76	2.71	4.98	9.25	2.42	1.15
	Test	10.12				10.12			
Wang *et al* [5] Bi-SST	Validation	10.89				5.86	7.99	2.55	1.31
	Test					9.65			
Li *et al* [4] Descriptivness Regr.	Validation	10.33	26.26	4.55	1.71	6.93	13.21	2.27	0.74
	Test					12.96			
Xu *et al* [7] JEDDi-Net	Test			4.06	1.63	8.58	19.88	4.06	1.63
Mun *et al* [24] Streamlined DVC	Validation	13.07	43.48	4.41	1.28	8.82	30.68	2.94	0.93
	Test					8.19			
Duan *et al* [23] Weakly Supervised DCE	Test			2.62	1.27	6.3	18.77	2.62	1.27
Rahman *et al* [25] Watch, Listen, Tell	Validation *	7.23	25.36	3.04	1.46	4.93	13.79	1.85	0.9
Suin *et al* [26] Efficient framework for DVC	Validation			2.87	1.35	6.21	13.82	2.87	1.35
Iashin *et al* [27] Multi-modal DVC	Validation *	11.72		5.83	2.86	7.31		2.6	1.07
Iashin *et al* [6] BMT	Validation *	10.90		4.63	1.99	8.44		3.84	1.88
Xiong *et al* [20] Move Forward and Tell	Validation			2.84	1.24	7.08		2.84	1.24
Wang *et al* [29] SYSU	Validation	14.85				10.31			
	Test					9.28			
Chadha *et al* [31] iPerceive	Validation *	12.27		6.13	2.98	7.87		2.93	1.29
Deng *et al* [8] Sketch, Ground and Refine	Validation				1.67	9.37	22.12		1.67
Chen *et al* [32] Towards Bridging EC-SL	Validation			2.78	1.33	7.49	21.21	2.78	1.33
Alwassel *et al* [33] TSP with BMT	Validation			4.16	2.02	8.75		4.16	2.02
Wang *et al* [34] Parallel decoding	Validation *	11.26	53.65		3.12	8.08	28.59		1.96

Table 2.1: Performance comparison of previous methods on the ActivityNet Captions dataset (* some videos unavailable)

2.1 Identified Themes

Dense video captioning can be decomposed into two parts: event localization and event description. Existing research methodologies can be grouped into different categories based on training methods, modalities used and ordering of tasks.

2.1.1 Based on Training schemes

:

1. **Independent training**
2. **Alternate training**: alternate between i) training the proposal module only and ii) training the captioning module on the positive event proposals while fine-tuning the proposal module.
3. **End-to-End**
4. **Weakly Supervised**

2.1.2 Based on Modalities

:

1. **Uni-modal**: Only visual features
2. **Multi-modal**: Combinations of visual, audio and speech features.

2.1.3 Based on Task ordering

:

1. **Top-Down**: localize-then-describe
2. **Bottom-Up**: describe-localize-refine

2.2 Research Literature Gaps

Chapter 3

Methodology

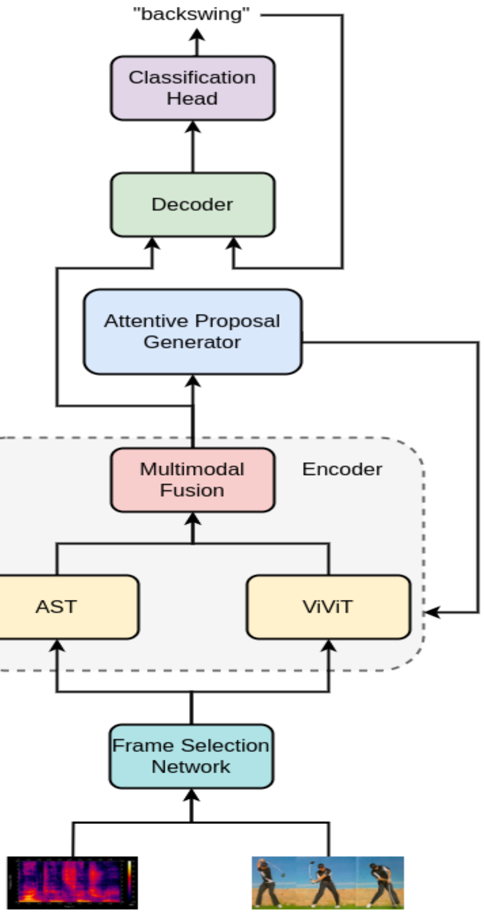

"backswing"
Classification Head
Decoder
Attentive Proposal Generator
Multimodal Fusion
Encoder
AST
ViViT
Frame Selection Network

Figure 3.1: Proposed model for Dense Video Captioning

3.0.1 Multimodal Feature Extraction

Feature extraction is the backbone of our solution to tackle the task of Dense Video Captioning [2]. A rich feature space would enhance the representational power of the model, thereby leading to more meaningful and accurate event proposals. Most previous works have utilized one modality only (i.e. video) to generate feature vectors as input to the proposal generator. However, audio cues, in conjuction with video is a strong event indicator. These events are accompanied with a sharp change in their corresponding audio spectogram which can be learnt by the model to better determine the precise boundary of events. Thus, we aim to combine features generated using both video and audio. Currently, we are exploring two methods of fusion which include the cross-attention mechanism [6] and common space proejction using contrastive learning [35].

Video Features

Video features are the most important part of the encoder. Without a robust and rich feature space for videos, the proposal generator would never be able to learn accurate event boundaries. Previous state-of-the-art video encoders [36], [37], [38] use CNN-based architectures. Although they have strong inductive bias and translational invariance, they fail to model long-range temporal dependencies which are of paramount importance when encoding videos. Moreover, CNNs require different architectures to model different modalities which can become complex when combining multiple modalities such as video and audio. Transformers [1] can overcome these barriers using its attention mechanism without comprimising on its statistical and computational efficiency. Moreover, transformers can use the same building blocks across different modalities without many modifications. We aim to use the recently proposed ViViT model [39], a purely attention-based video encoder which has outperformed previous approaches across several datasets such as Kinetics 400 and 600, Epic Kitchens 100, Something-Something v2 and Moments in Time. Even though ViViT requires several orders of magnitude more training data as compared to its CNN counterparts, the authors of ViViT propose several methods to limit its training by using pretrained weights in conjunction with strong regularisation and specific fine-tuning.

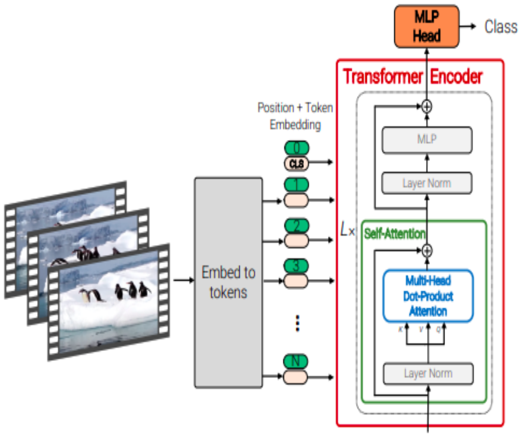


Figure 3.2: ViViT encoder by Dosovitskiy *et al*. (Image courtesy [39])

Audio Features

Audio is an important aspect of any video. Not only does audio suggest the duration of an event, it can also signify the magnitude or significance of that event. Thus, audio becomes a powerful accessory to images when representing a video. We aim to use the recently proposed Audio Spectogram Transformer (AST) [40] which has outperformed current state-of-the-art models in audio classification. Moreover, AST uses pretrained weights from ViT [41], just like ViViT. We believe that this would thus, work cohesively with ViViT and lead to richer multimodal features.

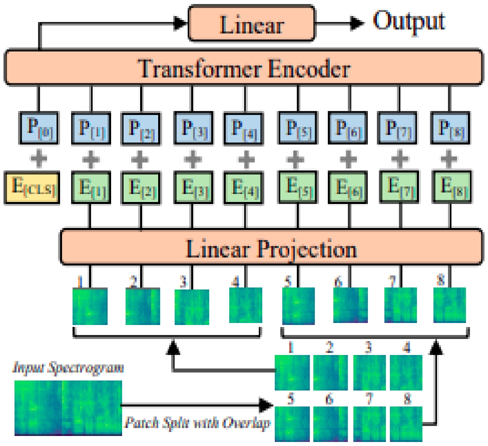

Linear
Output
Transformer Encoder
P[0]
P[1]
P[2]
P[3]
P[4]
P[5]
P[6]
P[7]
P[8]
E[CLS]
E[1]
E[2]
E[3]
E[4]
E[5]
E[6]
E[7]
E[8]
Linear Projection
1
2
3
4
5
6
7
8
Input Spectrogram
Patch Split with Overlap

Figure 3.3: AST encoder by Gong *et al*. (Image courtesy [40])

3.0.2 Training

Knowledge Distillation

We aim to use the Knowledge Distillation framework (student-teacher paradigm) to train the model, either in individual modules or as a whole. For the video encoder (ViViT), we would use a strong CNN-based video classifier as the teacher model to introduce inductive bias within the transformer and reduce training.Touvron *et al* introduced a method for knowledge distillation [42] using a distillation token to compute the loss based on the softmax generated by the teacher model.

We also aim to train the proposal generator using a student-teacher paradigm with the current SOTA DVC models.

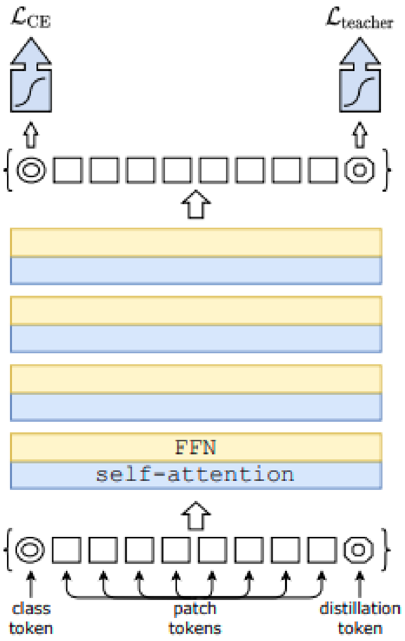

$\mathcal{L}_{\text{CE}}$
$\mathcal{L}_{\text{teacher}}$
FFN
self-attention
class token
patch tokens
distillation token

Figure 3.4: Distillation through attention by Touvran *et al*. (Image courtesy [42])

Chapter 4

Feature Extraction

For the task of Dense Video Captioning, a rich feature space would enhance the representational power of the model, thereby leading to more meaningful and accurate event proposals and captions. In most previous works, convolutional neural networks are commonly used to extract features from videos for any downstream tasks. Moreover, most localization methods use video features extracted by models that are trained for the task of Trimmed Action Classification (TAC), on datasets such as Kinetics[43] and Sports-1M, such as R(2+1)D[38], I3D[36], C3D and others. Additionally, most previous works focus on using a single modality (i.e. video) or multiple modalities based only on video, like RGB and optical flow features, for example from I3D[36], to generate feature vectors as input to the downstream model. Some works have used multiple modalities including video (RGB, optical flow) and audio, like [6]. The motivation for using multiple modalities is so that the feature space can grow richer, and that multiple modalities can complement each other to give strong event indicators.

In this section, we explain how features are extracted from videos in the form of two modalities: video and audio. We use the *Video Vision Transformer (ViViT)* [39] for video feature extraction and the *Audio Spectrogram Transformer (AST)* [40] for audio feature extraction. We also consider the temporal sensitivity of features as a characteristic that can benefit DVC.

4.1 Feature Extraction Methodology

4.1.1 Video Features Extraction Using ViViT

Video features are the most important part of the encoder. Without a robust and rich feature space for videos, the downstream model would never be able to learn accurate event boundaries or captions. Previous state-of-the-art video encoders [36], [37], [38] use CNN-based architectures. Although they have strong inductive bias and translation invariance, they fail to model long-range temporal dependencies which are of importance when encoding videos. Moreover, CNNs require different architectures to model different modalities which can become complex when combining multiple modalities such as video and audio. Transformers [1] can overcome these barriers using their attention mechanism without compromising on statistical and computational efficiency. Moreover, transformers can use the same building blocks across different modalities without many modifications. We aim to use the recently proposed ViViT model [39], a purely attention-based video encoder which has outperformed previous approaches for action recognition such as Kinetics 400 and 600, Epic Kitchens 100, Something-Something v2 and Moments in Time. Even though ViViT requires several orders of magnitude more training data as compared to its CNN counterparts, the authors of ViViT propose several methods to limit its training requirements by using pretrained weights in conjunction with strong regularization and specific fine-tuning.

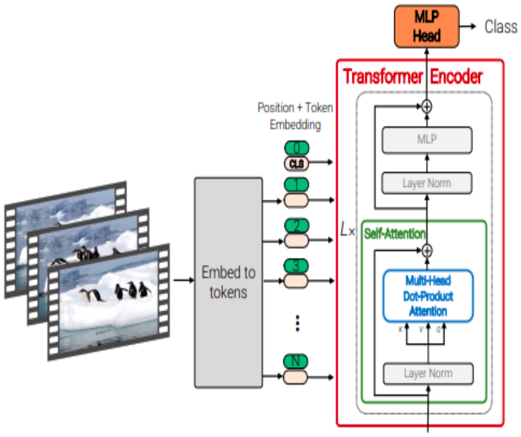


Figure 4.1: ViViT encoder by Dosovitskiy *et al*. (Image courtesy [39])

Arnab *et al* propose two methods for mapping a video to its corresponding embeddings for input to ViViT, *Uniform frame sampling* and *Tubelet embedding* in [39]. We use the uniform frame sampling method in our feature extraction framework. Arnab *et al* also propose different attention mechanisms for the ViViT model; we use the *spatio-temporal attention* mechanism. The flow of data is as follows:

1. Consider the original video clip X as consisting of T frames, each of width W and height H. The video has C channels. Hence the shape of video frames of X is (T, H, W, C).
2. We perform certain preprocessing steps on the video frames:
   (a) We resize the spatial dimensions of the video frames. The shorter edge is resized to 256, and the longer edge is resized maintaining the aspect ratio.
   (b) We normalize all channels of the video frames.
   (c) Generalizability and robustness of the features motivate data augmentation of the video frames. To this end, we randomly flip the video frames horizontally, and take random crops of spatial dimensions 224×224 during training. Note that for validation and inference, we skip the random horizontal flip step and take a center crop of the same dimensions.
3. Now, the uniform frame sampling method proposed by Arnab *et al* in [39] is used to extract tokens mapped to a 768-dimensional space. This involves using a three-dimensional convolutional layer to project the frames to patch embeddings, so that the shape of the embeddings are $(T, N, 768)$, where N is the number of patches.
4. The patch embeddings are then flattened to $(T \times N, 768)$, and concatenated with a *class token* for input to the encoder layers of ViViT.
5. For each encoder layer, the input is subject to (1) positional encoding, (2) layer normalization, (3) multi-head self-attention across all tokens, (4) layer normalization and (5) two fully connected layers with GeLU activation and dropout.
6. Finally, the class token from the output of the last encoder layer is used as the feature representation of the clip X.

More details about the different frame sampling methods, attention mechanisms and the ViViT model are given in B.5

4.1.2 Audio Features Extraction using AST

Audio is an important aspect of any video. Not only does audio suggest the duration of an event, it can also signify the magnitude or significance of that event. Thus, audio becomes a powerful accessory to images when representing a video. Convolutional neural networks (CNNs) have been widely used to learn representations from raw spectrograms for end-to-end modelling, as the inductive biases inherent to CNNs such as spatial locality and translation equivariance are believed to be helpful. In order to better capture long-range global context, a trend is to add a self-attention mechanism on top of the CNN. Such CNN-attention hybrid models have achieved state-of-the-art (SOTA) results. We aim to use the recently proposed Audio Spectrogram Transformer (AST) [40] which has outperformed current state-of-the-art models in audio classification. Audio Spectrogram Transformer is a convolution-free, purely attention based model that is directly applied to an audio spectrogram and can capture long-range global context even in the lowest layers.

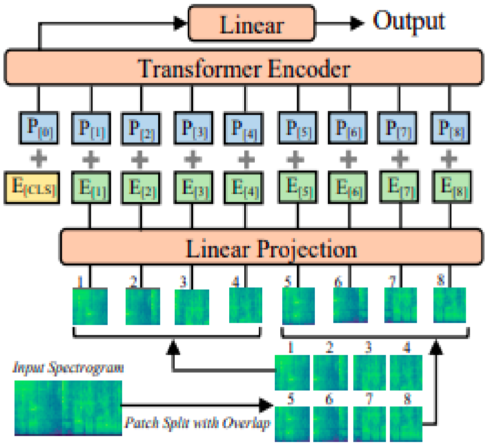


Figure 4.2: AST encoder by Gong *et al*. (Image courtesy [40])

The flow of data is as follows:

1. The audio signal is first pre-emphasised, then sliced into (overlapping) frames and a window function is applied to each frame. Following that, we perform a Short-Time Fourier Transform on each frame, calculate the power spectrum, and then compute the filter banks.
2. Thus a sequence of 128-dimensional log Mel filterbank (fbank) features is computed with a 25ms Hamming window every 10ms from the input audio waveform of t seconds.
3. The spectrogram is then split into a sequence of N 16x16 patches with a time and frequency overlap of 6, where N is the number of patches and the Transformer's effective input sequence length.
4. Each 16×16 patch is then flattened into a 1D patch embedding of size 768 using a linear projection layer.
5. Trainable positional embedding is added to each patch embedding which allows the model to capture the spatial structure of the 2D audio spectrogram.
6. The CLS token is then appended to the beginning of the sequence, which is subsequently fed into the encoder layer of AST.
7. Finally, the class token from the output of the last encoder layer is used as the feature representation of the clip.

4.2 Temporal Sensitivity of Features

As mentioned previously, most localization methods use video features extracted by models that are trained for Trimmed Action Classification (TAC) tasks. These features are not necessarily optimal for Temporal Action Localization (TAL) and other dependent tasks such as Action Proposal Generation and Dense Video Captioning. Alwassel *et al*, in their paper titled *TSP: Temporally Sensitive Pretraining of Video Encoders for Localization Tasks*, claim that this is due to the inherent lack of temporal sensitivity in features from TAC trained video encoders [33]. Yet, these features are prominently used for TAL and DVC tasks. This discrepancy is explained by the following reasons:

- Due to a lot of existing research for TAC tasks, a lot of established models exist that can be used off the shelf for video feature extraction
- The video encoders usually cannot be directly trained along with downstream tasks of TAL or DVC due to resource constraints; it is impractical to fit long untrimmed videos in commodity GPUs without drastically downsampling them in terms of space or time.

To solve this problem, Alwassel *et al* introduced a supervised pre-training paradigm for TAL that aims to instill the ability to produce temporally sensitive features; video encoders are trained to explicitly discriminate between foreground and background clips in untrimmed videos. They show that TSP improves performance for TAL, action proposal and DVC tasks consistently on different datasets as well as with different downstream predictive models and architectures, clearly establishing that temporally sensitive features contribute positively to the task at hand. TSP involves training encoders with a downstream task involving two classification heads: (1) *action classification* head and (2) *temporal region classification* head. This architecture is applied to clips from untrimmed videos. Optionally, along with local features of every clip, a *global video feature (GVF)* is also used for the temporal region classification to further improve temporal sensitivity [33]. A summary of their work is given in B.6.

4.2.1 Temporally Sensitive Pretraining of Proposed Feature Extractors

We extend the TSP framework proposed by Alwassel *et al* [33] to use temporally sensitive features in our work for DVC.

Architecture

The model used for TSP consists of one or more *backbone* architectures for particular modalities, which in our case are ViViT - for video modality, and AST - for audio modality. The constructed model is such that different backbones and their modalities can be plugged in and out as required for experimentation. The input modality data of a clip is fed in to each backbone separately, and the output feature representations from each backbone are combined and passed through a fully connected layer to give a final feature representation. This feature representation is fed into the action classification head and the temporal region classification head to give the respective outputs. Note that we do not use a GVF in the current implementation of our pretraining framework, and plan to include this in later work.

Data Flow

In our case, the following data flow is followed (considering two backbones: ViViT and AST):

1. Video frames are extracted from the clips, and are preprocessed and augmented as mentioned in 4.1.1 making the features at this level of the shape (B, C, T, H, W), where B is the batch size, C is the number of channels (3 for RGB), T is the temporal length of the clip in terms of number of frames and H and W are the height and width (spacial dimensions) of each frame in the clip respectively.
2. Audio frames are extracted from the clips as explained in 4.1.2. At this stage, they are of the shape (B, N, F), where N is the number of time frames and F is the frequency bins.
3. Video features of the shape (B, C, T, H, W) are fed into ViViT, and the class token of shape $(d_v,)$ of ViViT is used as the feature representation f_v of the video modality of the clip.
4. Audio features of the shape (B, N, F) are fed into AST, and the class token of the shape $(d_a,)$ of AST is used as the feature representation f_a of the audio modality of the clip.
5. These two features, f_a and f_v, are then combined using some function, for example addition or concatenation. In general, if there are n backbones and we have features $f_1, f_2, f_3, ..., f_n$, this combiner function must combine all these and give a single-dimensional feature f' of dimensions $(d',)$.
6. The feature f' is fed into a fully connected layer that converts it into a feature f of shape $(d,)$. This feature f is the final feature representation that is used for downstream tasks.

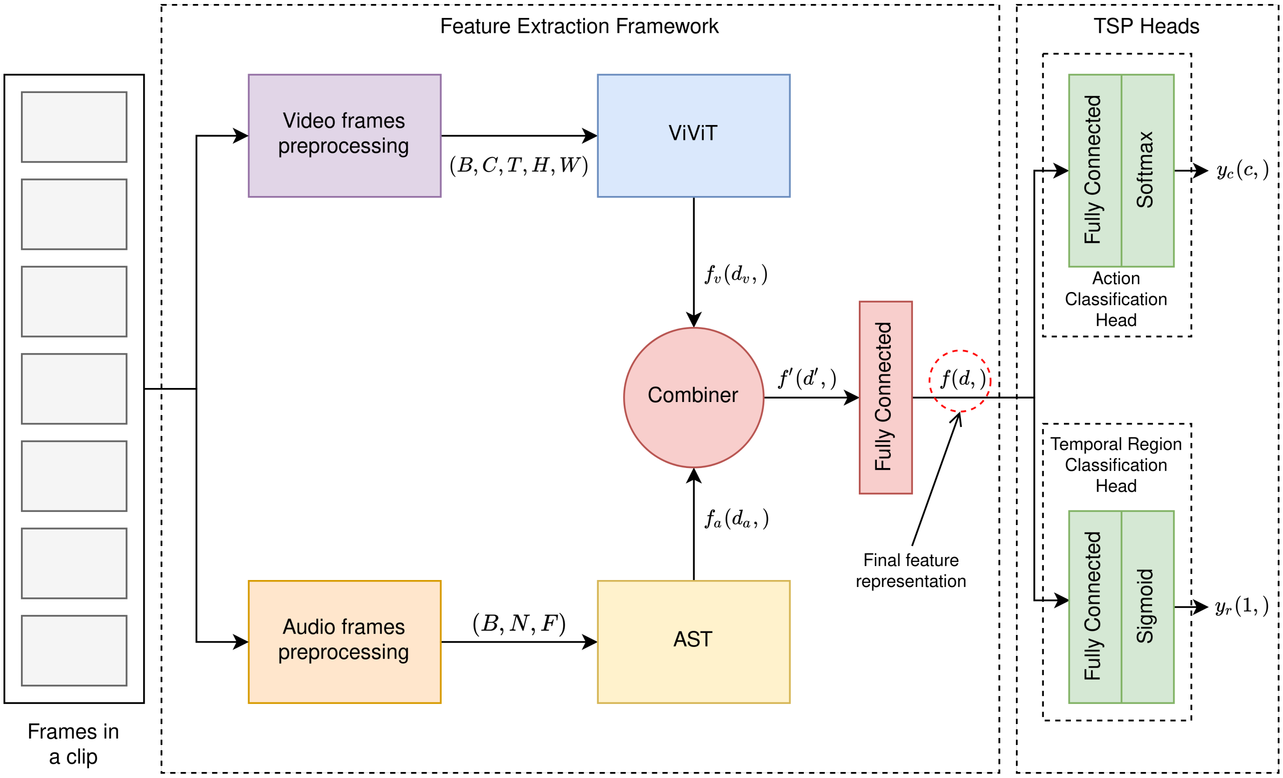

Feature Extraction Framework
TSP Heads
Video frames preprocessing
ViViT
(B, C, T, H, W)
$f_v(d_v,)$
Combiner
$f'(d',)$
Fully Connected
$f(d,)$
Final feature representation
$f_a(d_a,)$
Audio frames preprocessing
(B, N, F)
AST
Frames in a clip
Fully Connected
Softmax
$y_c(c,)$
Action Classification Head
Temporal Region Classification Head
Fully Connected
Sigmoid
$y_r(1,)$

Figure 4.3: The proposed feature extraction framework with ViViT and AST backbones. TSP heads for action classification and temporal region classification are shown on the right.

7. f is fed to the action classification head, which consists of a simple fully connected layer followed by softmax activation layer, to give the action classification logits y_c as output.
8. f is also fed (optionally combined with the GVF, f_g, to improve the temporal sensitivity of the feature representation) to the temporal region classification head, which consists of a fully connected layer followed by the sigmoid activation layer, to give the binary temporal region classification logit y_r as output.

The global video feature is extracted in the same manner as local features, with the additional step of *pooling* across all local features $f_1, f_2, ..., f_n$ for a video.

Loss functions

As mentioned, there are two classification heads on the top of the TSP framework:

- **Action Classification**: This head predicts action classes for a given input clip. The head can be represented as $f \rightarrow y_c$. The cross-entropy loss function is used for this head, with mean reduction (L_c):

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y} log(\hat{\mathbf{y}})$$

- **Temporal Region Classification**: This head predicts temporal region classes for a given input clip, i.e. whether the particular clip lies in an action (foreground) temporal region or non-action (background) temporal region. The head can be represented as $f \rightarrow y_r$. For binary classification, the cross entropy function reduces to binary cross entropy (L_r).

The two losses, L_c and L_r are given loss coefficients α_c and α_r for relative importance for contributing towards the total loss:

$$L = \alpha_c L_c + \alpha_r L_r$$

We evaluate the effectiveness of this temporally sensitive pretraining in the Results section.

Appendix A

Supplementary Details

Appendix B

Paper Summaries

B.1 Dense-Captioning Events in Videos

B.1.1 Overview

Ranjay Krishna *et al*, in their 2017 paper titled *Dense-Captioning Events in Videos* [2], proposed the problem of *dense video captioning*. The network was able to output overlapping events and events with varying time scales. The captioning module output also considered past and future context of the representative event. They also introduced new datset with over 20K annotated videos for this problem - ActivityNet Captions.

B.1.2 Datasets

- ActivityNet Captions

B.1.3 Performance

Krishna *et al* compared captioning results with LSTM-YT[44], S2VT[45] and H-RNN[46] using BLEU@1-4, METEOR and CIDEr. Since these models were only for captioning, the results were compared by providing ground-truth proposal boundaries. The metrics showed better results even when their pipeline generated captions on learnt proposals. The model was tested with different type of contexts input in captioning model, and full-context (*both past and future*) proved to better for longer videos. The results of event localization depicted how multi-stride network performed better for videos with more number of events. The model also reported state-of-the-art results for the task of vdeo and paragraph retrieval

B.1.4 Methodology

Krishna *et al* architecture consists of two main components:

1. Event proposal module
2. Captioning module with context

Flow of pipeline

1. C3D features of sampled video frames.
2. Sampling done at different strides to capture all kind of events.
3. Features fed into LSTM type of network inspired by DAPs[16] for generating proposals.
4. Captioning module used a LSTM network with following inputs:
   (a) hidden representation of past events h_i^{past}
   (b) hidden representation of current event h_i
   (c) hidden representation of future events h_i^{future}

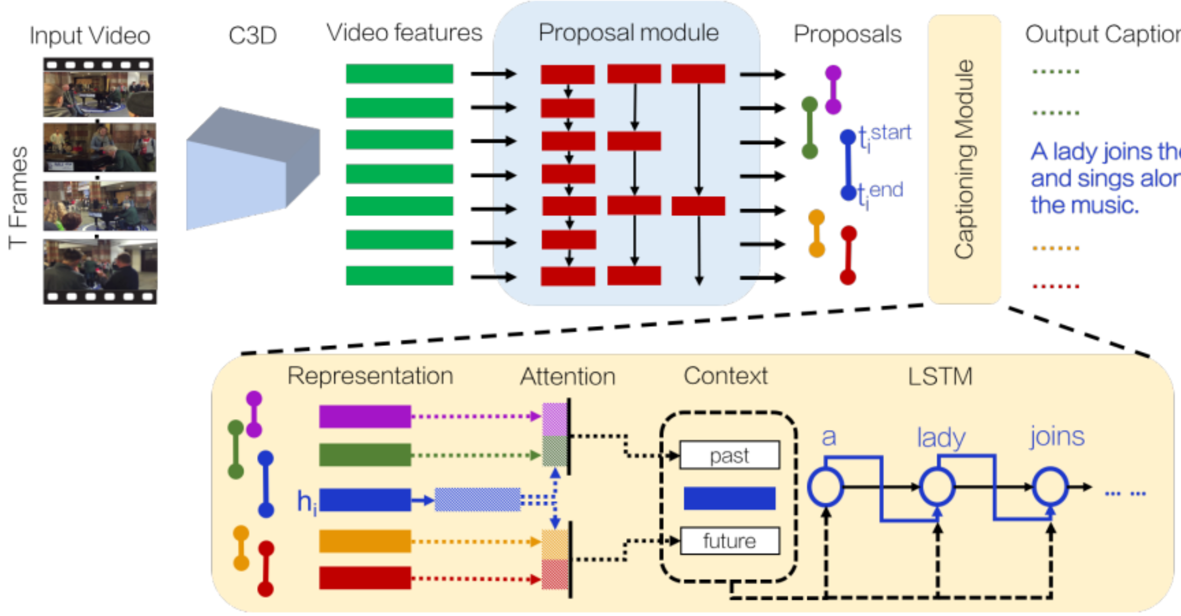

Input Video
T Frames
C3D
Video features
Proposal module
Proposals
t_i^{start}
t_i^{end}
Captioning Module
Output Captions
A lady joins the man and sings along to the music.
Representation
Attention
Context
LSTM
h_i
past
future
a
lady
joins

Figure B.1: Pipeline introduced by Krishna *et al* (Image courtesy [2])

B.1.5 Conclusion

Krishna *et al* tackled the task of captioning videos with correlation between events. They also showed the results on streaming videos with current event relating to only past events.

B.2 A Better Use of Audio-Visual Cues: Dense Video Captioning with Bi-modal Transformer

B.2.1 Overview

Iashin and Rahtu, in their 2020 paper titled *A Better Use of Audio-Visual Cues* [6], approached the task of Dense Video captioning by using two types of input features: video as well as audio. They identified the lack of research into utilizing the audio track of videos to generate dense captions, despite the natural co-occurence of the two. They introduce a nove *Bimodal Transformer* which has the similar encoder-decoder architecture as the traditional Transformer model, but generalizes it to two input modalities: audio features (using VGGish [47]) and visual features (using I3D [36]). Their framework also consists of a *multiheaded bimodal proposal generator*, inspired by the *YOLO* object detector [28] which generates event proposals for caption generation.

B.2.2 Datasets

- ActivityNet Captions [2]

B.2.3 Performance

Iashin *et al* report state-of-the-art-performance for the dense video captioning task on the published validation subset of the ActivityNet Captions dataset in terms of BLEU@3-4 metrics and near state-of-the-art performance in terms of the METEOR metric. Considering the task of proposal generation in isolation, they report state-of-the-art performance with the F1 score of 60.27%, on the same dataset.

B.2.4 Methodology

Iashin *et al* use a Inflated 3D Network pre-trained on the *Kinetics* dataset [43] for extracting visual features from video frames. For audio features, the VGGish network [47] is used. These features are stacked in sequence for each video. The caption tokens are embedded with pre-trained *GloVe* [48].

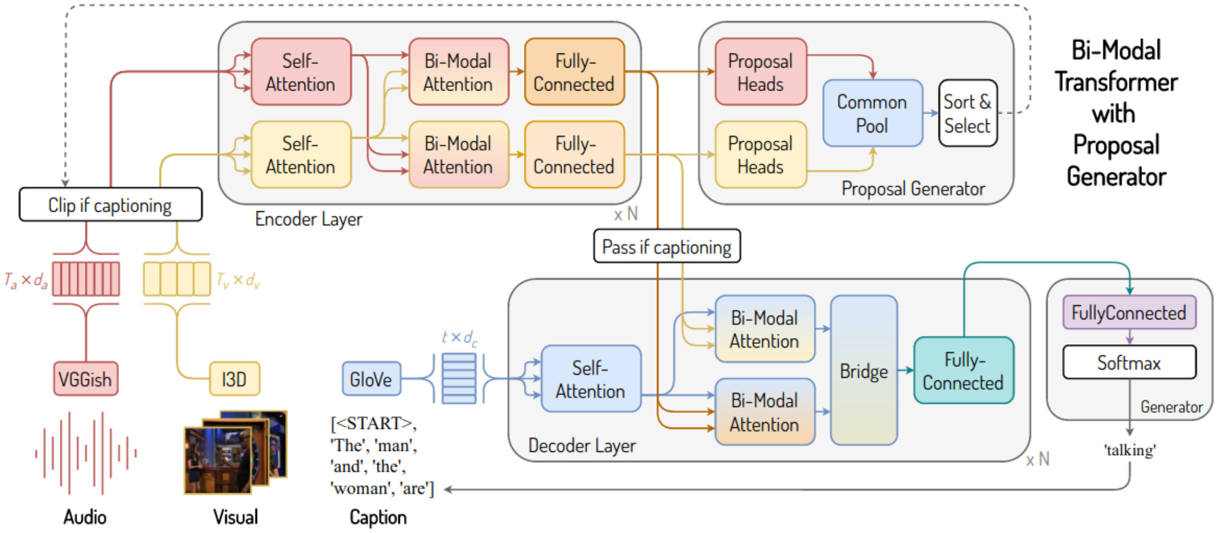

Self-Attention
Bi-Modal Attention
Fully-Connected
Self-Attention
Bi-Modal Attention
Fully-Connected
Proposal Heads
Common Pool
Sort & Select
Proposal Heads
Proposal Generator
Bi-Modal Transformer with Proposal Generator
Clip if captioning
Encoder Layer
x N
Pass if captioning
$T_a \times d_a$
$T_v \times d_v$
VGGish
I3D
$t \times d_c$
GloVe
Self-Attention
Bi-Modal Attention
Bi-Modal Attention
Bridge
Fully-Connected
Decoder Layer
x N
FullyConnected
Softmax
Generator
'talking'
[<START>, 'The', 'man', 'and', 'the', 'woman', 'are']
Audio
Visual
Caption

Figure B.2: Dense Video Captioning Framework introduced by Iashin *et al* (Image courtesy [6])

The design proposed by Iashin *et al* consists of two major functional components:

- Bi-modal Transformer
- Multiheaded Proposal Generator

For inference, the features flow through the components as follows:

1. Both feature sequences, audio A and visual V are self-attended and then passed through an N-layered encoder to produce bi-modal sequence representations using novel bi-modal multi-headed attention blocks to fuse features from both sequences. These bimodal attention blocks output two sequences of features, A^v (visual attended audio features) and V^a (audio attended visual features).

2. A^v and V^a are then given to the Proposal Generator, which consists of two distinct sets of *proposal generator heads*, one set for each input modality. Each proposal generator head makes proposal predictions at each time step independently, forming a common pool of cross-modal predictions. Top k predictions are selected (using a confidence score) from this pool to be the output of the proposal generator: the proposals.

3. The generated proposals are used to clip the input feature sequences which are then given to the Bimodal Encoder. The same self-attention followed by bimodal attention is carried out on these clipped features.

4. The encoder's outputs are passed to the bi-modal attention blocks in every decoder layer of the Bimodal Decoder, along with the representation of previously generated caption words. The caption embeddings are self-attended, and then carries out bi-modal encoder-decoder attention, producing $C_n^{A^v}$ (audio-visual attended previous captions) and $C_n^{V^a}$ (visual-audio attended previous captions). After a bridge connection and positional encoding (to add order information to the permutation invariant transformer architecture), two fully connected layers with ReLU activation between them and a softmax activation in the final layer are used to model the distribution of the next caption token over the vocabulary.

Proposal Generation Module

The design of individual proposal generator heads in the Proposal Generator is inspired by the YOLO object detector [28]. A set of anchors is determined apriori by running K-Means Clustering on the ground truth event lengths. Each centroid of the cluster is taken as an anchor in the anchor set ψ. The distance metric used here is the Euclidean distance, in contrast to IoU (Intersection over Union) used in YOLO. Each proposal generator head predicts three values: (1) center of the event c, (2) log coefficient l and (3) objectness score o. Using these values, the temporal boundaries and confidence of the event predicted are calculated:

$$center = p + \sigma(c)$$

$$length = anchor\_length \cdot e^l$$

$$confidence = \sigma(o)$$

Each proposal generator head consists of three 1D convolutional layers, through which the sequence length is preserved using padding and unit stride. The first convolutional layer has kernel size from a predetermined set of kernel sizes (calculated for each modality using K-Means clustering on event ground truth event lengths; the idea is to match receptive field sizes with event sizes). The other two layers have unit kernel size.

The loss for center c and log coefficient l, collectively called *Localization error* is the Mean Squared Error (MSE), while the confidence loss is a weighted sum of Binary Cross Entropy (BCE) of objectness and no-objectness loss.

Captioning Module

The captioning module is essentially the Bimodal Transformer, consisting of a bimodal encoder and a bimodal decoder. The input to the bimodal encoder during captioning is feature sequences A and V, which temporally correspond to a proposal. These two sequences go through N encoder layers, each of which consist of (1) multiheaded self-attention of the two input sequences, (2) multiheaded bi-modal attention of the two sequences, giving A^v and V^a and then (3) two position-wise fully-connected layers with residual connections. These sublayers have separate trainable weights for both modalities.

The bimodal decoder is given the previous sequence of caption tokens as input, along with the output of the encoder A_N^v and V_N^a. Each decoder layer consists of the following sublayers: (1) self-attention of caption tokens, (2) bimodal attention of captions with A^v and V^a, (3) a bridge connection to concatenate and combine $C_n^{A^v}$ (audio-visual attended previous captions) and $C_n^{V^a}$ (visual-audio attended previous captions) and (4) two position-wise fully connected layers which act as the classification layers for the next caption token.

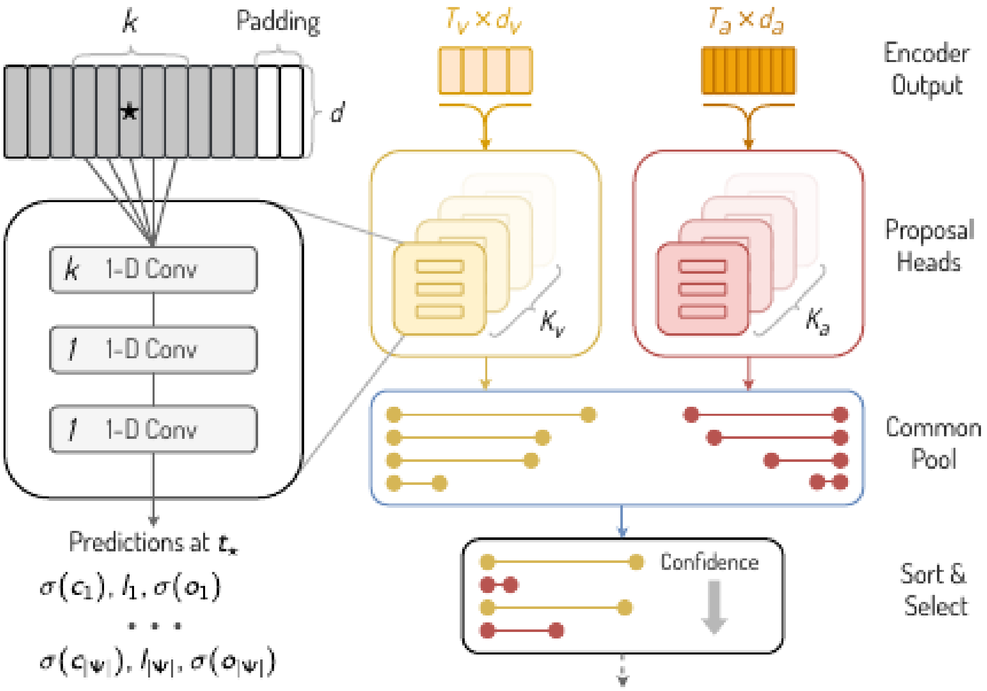

k
Padding
d
★
$T_v \times d_v$
$T_a \times d_a$
Encoder Output
Proposal Heads
K_v
K_a
k 1-D Conv
1 1-D Conv
1 1-D Conv
Common Pool
Confidence
Sort & Select
Predictions at t_*
$\sigma(c_1), l_1, \sigma(o_1)$
. . .
$\sigma(c_{|\Psi|}), l_{|\Psi|}, \sigma(o_{|\Psi|})$

Figure B.3: Bi-modal Multiheaded Proposal Generator: Architecture (Image courtesy [6])

Training Procedure

First, the captioning module is trained using ground truth proposals and their given captions using KL-divergence loss and label smoothing. The bimodal encoder weights are then frozen and then the proposal generator is trained using the trained bimodal encoder. Each proposal head uses Mean Squared Error for localization and binary cross-entropy for proposal loss.

B.2.5 Conclusion

The work by Iashin *et al* provides a compelling case for exploring the use of multiple modalities for the task of Dense Video Captioning. Their ablation study shows that performance with using only visual features is more than using only audio features, indicating visual modality has a stronger signal for video understanding than audio modality; however, performance using both modalities gives consistently better results than single modalities in all settings.

B.3 Vision Transformer (ViT) - An image is worth 16x16 words: Transformers for image recognition at scale

B.3.1 Overview

Dosovitskiy *et al*, in their 2020 paper titled *An image is worth 16x16 words: Transformers for image recognition at scale* [41], aim to replace CNN based architectures with transformers for a multitude of computer vision tasks. They propose a transformer-based architecture inspired by Vaswani *et al* [1], replacing the work embeddings with non-overlapping 'patch embeddings', extracted from images. The authors show that if ViT is pre-trained on large amounts of data, it can produce better results as compared to its CNN counterparts in various image recognition benchmarks.

B.3.2 Datasets

- Pre-trained on:
  - ILSVRC-2012 ImageNet dataset with 1k classes and 1.3M images
  - ImageNet 21k dataset with 21k classes and 14M images
  - JFT with 18k classes and 303M high-resolution images
- Benchmark tasks
  - ImageNet (validation labels)
  - CIFAR 10/100
  - Oxford-IIIT Pets
  - Oxford Flowers-102
  - 19-task Visual Task Adaptation Benchmark (VTAB) classification suite.

B.3.3 Performance

Dosovitskiy *et al* report state-of-the-art-performance for various image recognition tasks (datasets mentioned above), when compared with current state-of-the-art models such as Big Transfer (BiT) [49] and Noisy Student [50].

B.3.4 Methodology

Dosovitskiy *et al* use the original transformer model [1], but with different input embeddings that are extracted from images. The transformer model then outputs a classification ([class]) token which represents the entire image. This [class] token is then used for downstream tasks (mentioned above).

Input: Patch embeddings

The input consists of flattened nxn patches of an image. These patches are non-overlapping and span the entire image.

- The original image $x = R^{H \times W \times C}$.
- This is converted to a flattened 2D patch $x_p = R^{N \times (P^2 \cdot C)}$ where $H \times W$ is the resolution of the image, C is the number of channels, $P \times P$ is the resolution of each patch.
- $N = HW/P^2$ represents the number of patches extracted from the image and serve as the sequence length for the transformer. These are the patch embeddings.
- Position embeddings are added to the above patch embeddings. These can be:
  - 1D, like in transformers.
  - 2D, based on row and column indices.

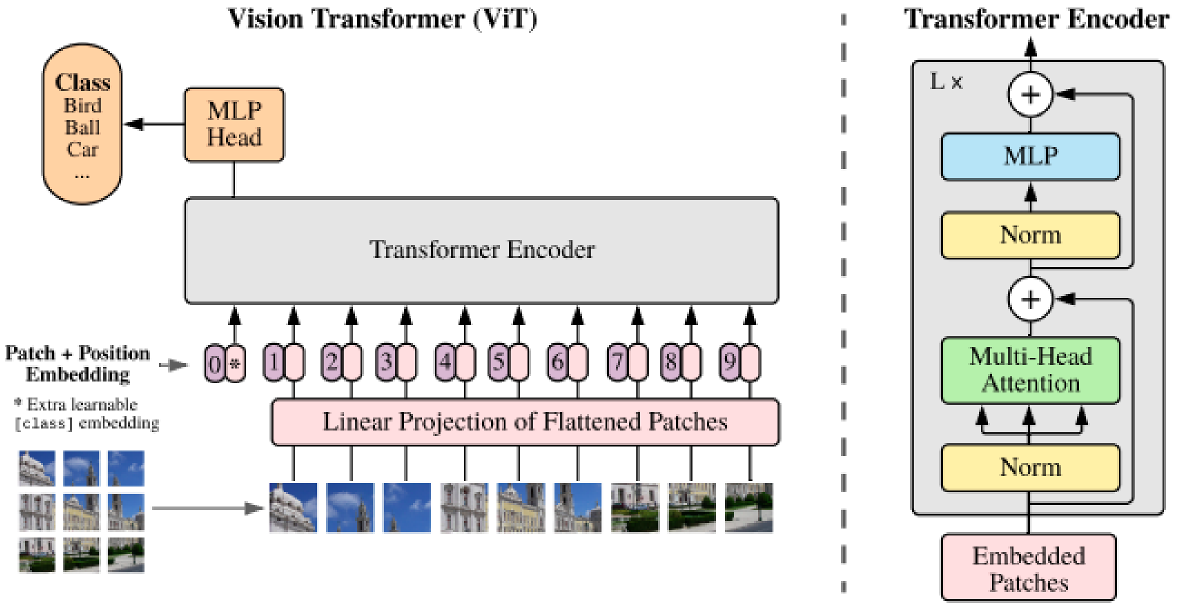

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L ×
+
MLP
Norm
+
Multi-Head Attention
Norm
Embedded Patches

Figure B.4: Video Transformer introduced by Dosovitskiy *et al* (Image courtesy [41])

$$
\begin{aligned}
\mathbf{z}_0 &= [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E}; \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, && \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \\
\mathbf{z'}_\ell &= \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, && \ell = 1 \ldots L \\
\mathbf{z}_\ell &= \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, && \ell = 1 \ldots L \\
\mathbf{y} &= \mathrm{LN}(\mathbf{z}_L^0)
\end{aligned}
$$

Figure B.5: Video Transformer introduced by Dosovitskiy *et al* (Courtesy [41])

  - Relative distances between patches to capture the true spatial positioning of each patch within the image.

- A learnable [class] token is prepended to the image sequence like the BERT model. z_0 is the input [class] token and z_l is the output token. z_l is a vector that represents the entire image and can be used for downstream tasks.
- These embeddings are mapped onto a D-dimensional space which is used by the encoder (ViT).

Transformer Encoder

The transformer model proposed in [1] is inherently the same one used by ViT. The only difference is that a normalisation layer is added before the multi-headed self attention block. Additionally, ViT used the GELU non-linearity as part of the multi-layer perceptron block.

Training Procedure

Pretrained on datasets such as ImageNet, ImageNet-21k, JFT-300M for better performance.

- Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size = 4096, weight decay = 0.1
- Strong regularization is used throughout the encoder. Dropout, when used, is applied after every dense layer except for the the query-key-value projection layers and directly after the addition of positional embeddings

- Training is done on a resolution of $224(14 \times 14)$

B.3.5 Conclusion

The work by Dosovitskiy *et al* provides a compelling case for using transformers instead of CNN-architectures in image recognition tasks. Even though it requires large amount of pre-training, it can be scaled much better than its CNN counterparts and can meet the performance (and even exceed it in certain cases) of current state-of-the-art CNN-based models.

B.4 AST: Audio Spectrogram Transformer

B.4.1 Overview

Gong *et al* in their 2021 paper titled *AST: Audio Spectrogram Transformer* [40] proposed a convolution-free, solely attention-based model that can capture long-range global context even in the lowest layers and is directly applied to an audio spectrogram. They also suggest a method for transferring knowledge from the ImageNet-pretrained Vision Transformer (ViT) to AST, which can greatly enhance performance.

B.4.2 Datasets

- AudioSet
- ESC-50
- Speech Commands

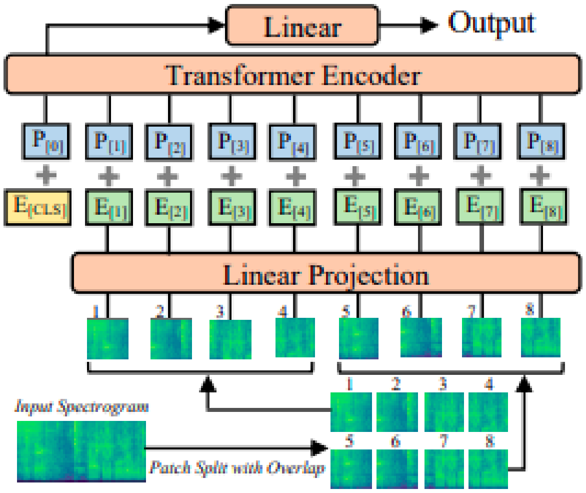


Figure B.6: Audio Spectrogram Transformer by Gong *et al* (Courtesy [40])

B.4.3 Methodology

- A sequence of 128-dimensional log Mel filterbank (fbank) features is computed with a 25ms Hamming window every 10ms from the input audio waveform of t seconds. As a result, the AST receives a 128x100t spectrogram as input.
- The spectrogram is then split into a sequence of N 16x16 patches with a time and frequency overlap of 6, where N is the number of patches and the Transformer's effective input sequence length.
- Each 16×16 patch is then flattened into a 1D patch embedding of size 768 using a linear projection layer. This linear projection layer is called patch embedding layer.
- Since the Transformer architecture does not capture the input order information and the patch sequence is also not in temporal order, trainable positional embedding (also of size 768) is added to each patch embedding which allows the model to capture the spatial structure of the 2D audio spectrogram.
- The CLS token is then appended to the beginning of the sequence, which is subsequently fed into the Transformer.
- A Transformer consists of several encoder and decoder layers. For classification tasks, AST only uses the encoder of the Transformer.

- The [CLS] token output from the Transformer encoder serves as the audio spectrogram representation. The audio spectrogram representation is mapped to labels for classification using a linear layer with sigmoid activation.
- AST model also allows cross-modality transfer learning since images and audio spectrograms have similar formats. So ImageNet-pretrained CNN weights are used as initial CNN weights for audio classification training.

B.4.4 Training procedure

- Model is pre-trained with the ImageNet dataset.
- Model is trained with a batch size of 12, the Adam optimizer, and uses binary cross-entropy loss.
- For balanced set experiments, the initial learning rate of 5e-5 is used and the model is trained for 25 epochs, the learning rate is cut into half every 5 epochs after the 10th epoch.
- For full set experiments, the initial learning rate of 1e-5 is used and the model is trained for 5 epochs, the learning rate is cut into half every epoch after the 2nd epoch.
- Mean Average Precision (mAP) is used as the main evaluation metric.

B.4.5 Conclusion

Gong et al present Audio Spectrogram Transformer which is a convolution-free, purely attention-based model for audio classification which features a simple architecture and superior performance.

B.5 ViViT: A Video Vision Transformer

B.5.1 Overview

Arnab *et al*, in their 2021 paper titled *ViViT: A Video Vision Transformer* [39], extend the Vision Transformer (ViT) [41] to tackle video recognition tasks. They propose various attention architectures to model the spatial and temporal interaction within video frames. The also propose embedding extraction methods for videos which are on similar lines to those done for images.

B.5.2 Datasets

- Pre-trained on:
  - ILSVRC-2012 ImageNet dataset with 1k classes and 1.3M images
  - ImageNet 21k dataset with 21k classes and 14M images
  - JFT with 18k classes and 303M high-resolution images
- Benchmark tasks
  - Kinetics 400/600
  - Epic Kitchens
  - Something-Something v2
  - Moments in Time

B.5.3 Performance

Arnab *et al* report state-of-the-art performance for various video recognition tasks (datasets mentioned above), when compared with current state-of-the-art models.

B.5.4 Methodology

Arnab *et al* use the ViT model [41] as its foundation, but with different input embeddings that are extracted from video and different attention blocks. The transformer model then outputs a classification ([class]) token which represents the entire video. This [class] token is then used for downstream tasks (mentioned above).

Input: Token embeddings

Two methods for mapping a video to its corresponding embeddings are proposed, namely:

- *Uniform frame sampling*
  - The original video $x = R^{T \times H \times W \times C}$.
  - From this, we extract $N = n_t \cdot n_h \cdot n_w$ tokens which are subsequently mapped to d-dimensional space. Here, n_t corresponds to the number of frames in the video.
- *Tubelet embedding*
  - This method models the 3D convolutions or "tubes".
  - $n_t = T/t$ frames are sampled, each with $n_h \cdot n_w$ non-overlapping patches.
  - From this, we extract $N = n_t \cdot n_h \cdot n_w$ tokens which are subsequently mapped to d-dimensional space.
- The final output consists of $x = R^{N \times d}$
- A learnable [class] token may be prepended to the token sequence based on the attention architecture being used.

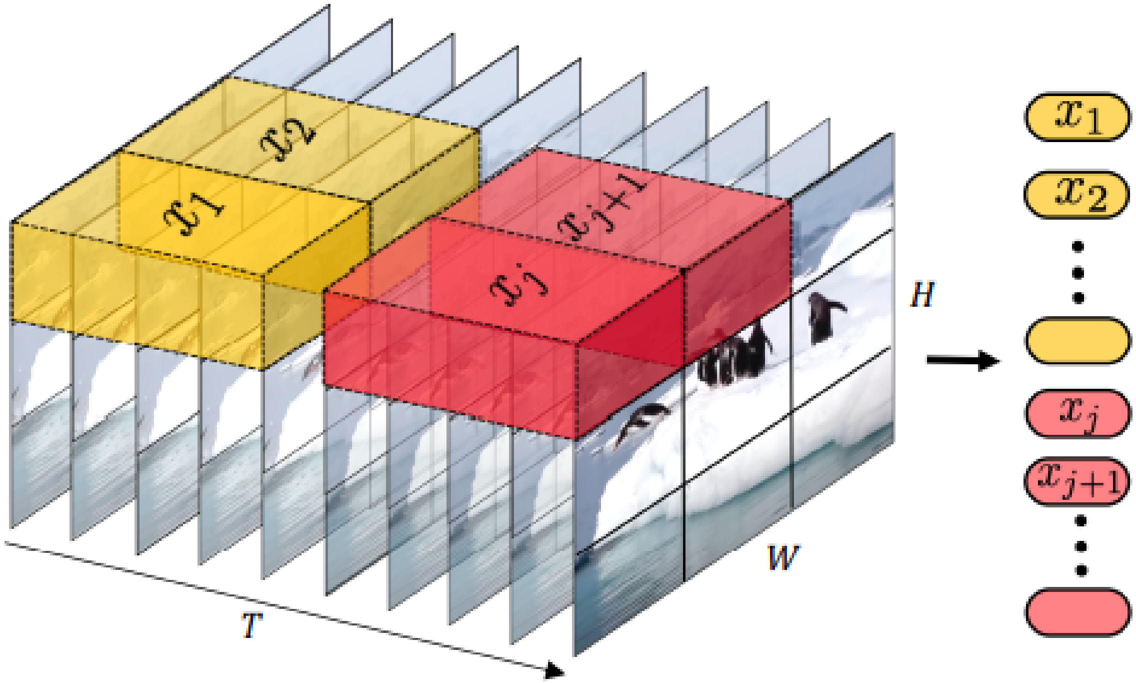


Figure B.7: Video Transformer introduced by Arnab *et al* (Courtesy [39])

Transformer Encoder

ViViT extends the transformer model proposed in [41] by making use of different attention architectures to model the spatio-temporal interaction between the token embeddings.

Four attention architectures are proposed, namely:

- *Spatio-temporal attention*
  - This architecture mimics the multi-headed self attention block [1] across all $N \times d$ tokens.
  - However, it has quadratic time complexity as it models all pairwise interactions between all spatio-temporal tokens.
- *Factorised encoder*
  - This architecture consists of two transformer encoders to model spatial and temporal interactions between the tokens separately.
  - First, the spatial encoder gets $n_h \cdot n_w$ tokens in order to attend to tokens within the same temporal index (same frame/tube).
  - Next, n_t [class] tokens representing $n_h \cdot n_w$ tokens of each frame are given as input the the temporal encoder.
  - The temporal encoder outputs another [class] token which is the collective representation of all the frames in the video.
  - Model 2 does have more parameters that model 1, but consists of fewer floating point operations due to a decrease in the number of tokens per encoder.
- *Factorised Self-Attention*
  - This architecture consist of two multi-headed self attention blocks within a single encoder.
  - The first one models spatial interactions between tokens and the second one, temporal.
  - The first attention block gets tokens $x = R^{n_t \times n_h \cdot n_w \times d}$ which include tokens within the same temporal index.

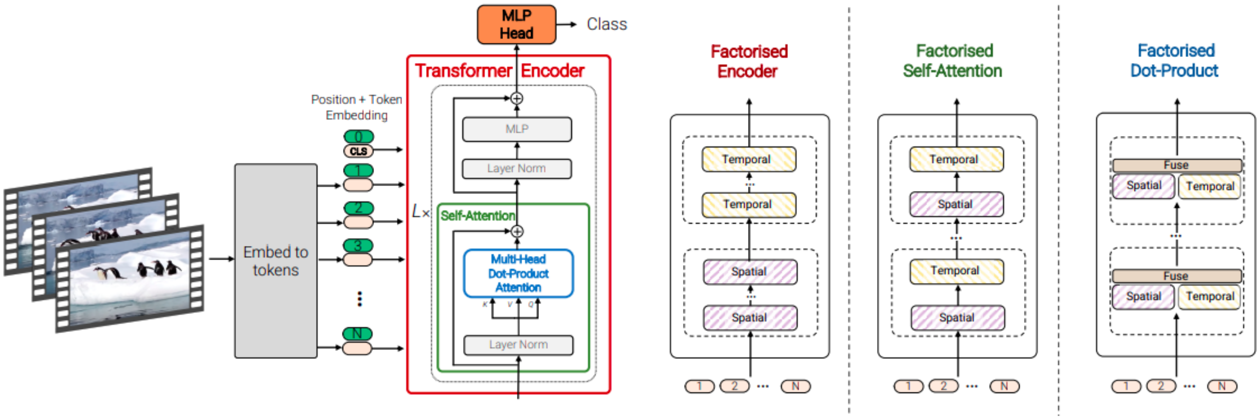

MLP Head
Class
Transformer Encoder
Position + Token Embedding
0
CLS
1
2
3
N
Embed to tokens
L×
MLP
Layer Norm
Self-Attention
Multi-Head Dot-Product Attention
K
V
Q
Layer Norm
Factorised Encoder
Temporal
Temporal
Spatial
Spatial
1
2
N
Factorised Self-Attention
Temporal
Spatial
Temporal
Spatial
1
2
N
Factorised Dot-Product
Fuse
Spatial
Temporal
Fuse
Spatial
Temporal
1
2
N

Figure B.8: Video Transformer introduced by Arnab *et al* (Image courtesy [39])

  - The second attention block gets tokens $y = R^{n_h \cdot n_w \times n_t \times d}$ which include tokens within the same spatial index.
  - Model 3 has fewer parameters but the same complexity as model 2.

- *Factorised Dot-Product Attention*
  - This architecture aims to fuse the spatial and temporal attention within the same multi-headed self attention block, but with different heads.
  - The spatial attention heads get $x = R^{n_t \times n_h \cdot n_w \times d}$ tokens as input whereas the temporal attention heads get $y = R^{n_h \cdot n_w \times n_t \times d}$ as input.
  - Model 4 has the same number of parameters as model 1 and the same complexity as models 2 and 3.

- The final output can either be a [class] token or a 1D vector obtained by average pooling all output vectors.

Training Procedure

Pre-trained weights of the ViT model are leveraged due to the size of various video recognition tasks (many orders of magnitude smaller than image recognition datasets). The model can then be trained on various video recognition datasets such as Kinetics and Epic Kitchens.

- Kinetics
- Strong regularization is used throughout the encoder. Dropout, when used, is applied after every dense layer except for the the query-key-value projection layers and directly after the addition of positional embeddings
- Training is done on a resolution of $224(14 \times 14)$

B.5.5 Conclusion

The work by Arnab *et al* provides a compelling case for using transformers instead of CNN-architectures in video recognition tasks. Even though it requires large amount of pre-training, it can be scaled much better than its CNN counterparts and can meet the performance (and even exceed it in certain cases) of current state-of-the-art CNN-based models.

B.6 Temporally Sensitive Pretraining of Video Encoders for Localization Tasks

B.6.1 Overview

Most localization methods use video features extracted by models that are trained for Trimmed Action Classification (TAC). They're not necessarily suited for Temporal Action Localization (TAL), since TAC-pretrained features tend to be temporally insensitive. For example, R(2+1)D, I3D and C3D have become the de facto video feature extractors for TAL, Action Proposal and DVC tasks; these are trained on TAC. The reasons for most work using TAC trained video encoders are (1) many established models exist for TAC and (2) it is impractical to fit untrimmed videos in commodity GPUs without drastically downsampling space or time. [33].

Alwassel *et al*, in their 2021 paper titled *TSP: Temporally Sensitive Pretraining of Video Encoders for Localization Tasks* introduce a supervised pre-training paradigm for TAL. This paradigm also considers background clips (which are not as important as for TAC) and global video information, to improve temporal sensitivity.

B.6.2 Contributions and Findings

- TSP trains an encoder to explicitly discriminate between foreground and background clips in untrimmed videos
- Temporally-sensitive features from TSP improves performance for TAL, action proposal and DVC Tasks
- Consistent performance gains for multiple algorithms, architectures and datasets
- TAL performance is boosted on short action instances

B.6.3 Pre-training Methodology

The major goal of TSP is to incorporate temporal sensitivity in video encoders. Hence, this pre-training involves training encoders on the task of classifying foreground clips, and classifying whether a clip is inside or outside the action. The data used for TSP consists of untrimmed videos with temporal annotations. The video encoder is trained end-to-end, from raw video input. From an untrimmed video, X is sampled, which has dimensions $3 \times L \times W \times H$, where L is the number of frames and W and H are the frame dimensions. 3 is the number of channels, i.e., RGB. Since there is a natural imbalance in the annotations of foreground and background clips, X is sampled in a way that an equal number of clips from each class is chosen. Each X is annotated with two labels:

- y^c: action class label, *if it is from a foreground clip*
- y^r: binary temporal region label, i.e. whether whether the clip is from a foreground/action region ($y^r = 1$) or background/no-action region ($y^r = 0$) of the video

B.6.4 Local and Global Feature Encoding

The encoder E transforms a clip X to a local F-dimensional feature f. Global Video Feature (GVF) is the max-pooled feature from all local features. For classifying the temporal region (foreground or background), combine each local feature f is combined with GVF; hence GVF acts as a conditional vector.

The video encoders used are ResNet3D and R(2+1)D, while the datasets used are ActivityNet v1.3 and THUMOS-14 [51]. The classification component involves two heads:

- An $F \times C$ fully connected layer for action classification: $f \to y^c$ (logits vector)
- A $2F \times 2$ fully connected layer for temporal region label (background foreground): $[f, GVF] \to y^r$

Cross-entropy loss is used for both classification heads. If the clip is a foreground clip, then loss from both heads are taken into account, relatively weighted by α^r and α^c. If the clip is a background clip, only the temporal region classification loss is taken, weighted by α^r.

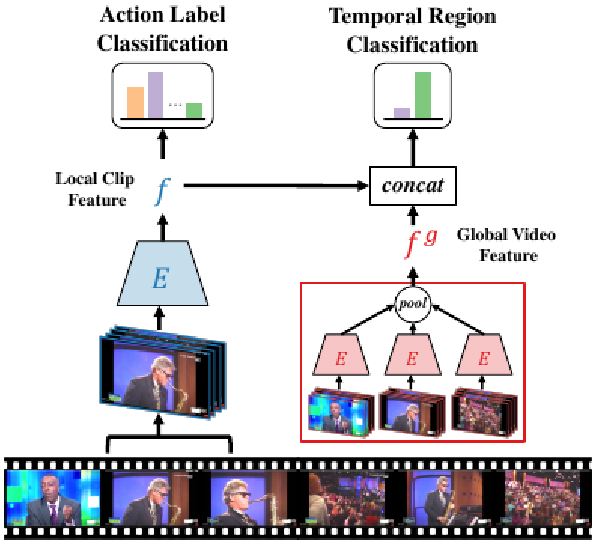


Figure B.9: The TSP framework. Image courtesy [33]

B.6.5 Performance

Using the pre-trained video encoder, Alwassel *et al* take some implementations of event localisation methods and feed temporally sensitive features to them. The algorithms used are:

- GTAD: sub-graph localization for Temporal Action Detection
- BMN: Boundary-matching network for Temporal Action Proposal Generation
- BMT: Bimodal Transformer for DVC [6] (audio features kept same)
- P-GCN: Graph CNN for Temporal Action Localization

Their observations are as follows [33]:

- Improved performance on multiple target tasks, such as TAC, TAL and DVC
- Consistency of performance regardless of type of video encoder used
- Consistent improvement in performance for all localization algorithms
- Indications of applicability of TSP on other datasets as well as its transferability

B.6.6 Conclusion

Alwassel *et al* present Temporally Sensitive Pretraining, a novel supervised pretraining approach for video encoders. This approach considers background clips along with foreground clips, and uses global information to gain temporal sensitivity. Their results indicate it is advantageous to use TSP features over other popular features to build accurate models [33], especially for DVC.

B.7 End-to-end Dense Video Captioning with Parallel Decoding

B.7.1 Overview

Wang *et al* in their paper, titled *End-to-end Dense Video Captioning with Parallel Decoding* [34] aim to address certain issues that they identify in existing DVC work, namely, the two-stage design and handcrafted and heuristic components for event proposals generation. They model DVC as a set prediction task, and propose a model which, using a transformer architecture based on the deformable DeTR [], performs event localization and event captioning in parallel. They utilize an event counter head to predict number of events based on video understanding.

B.7.2 Limitations Identified

Wang *et al* identify the following limitations of previous DVC work:

Two-stage design

Most models use a localize-then-describe strategy for DVC tasks, i.e. they first localize the events, and then feed these events to a captioning module for generating captions. These methods usually are not trained end-to-end; rather the two stages are trained separately. Consequently, the mutual promotion of the two subtasks is limited; captioning is considered a downstream task and hence inter-task associations are not learned.

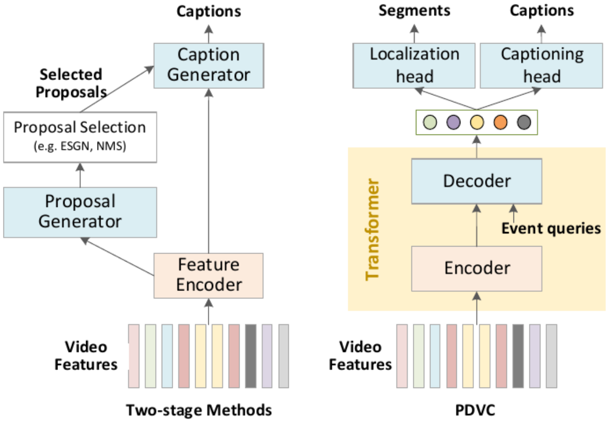

Captions
Caption Generator
Selected Proposals
Proposal Selection (e.g. ESGN, NMS)
Proposal Generator
Feature Encoder
Video Features
Two-stage Methods
Segments
Captions
Localization head
Captioning head
Transformer
Decoder
Event queries
Encoder
Video Features
PDVC

Figure B.10: Two-stage architecture versus proposed single-stage PDVC architecture. Image courtesy [34]

Unreliable Event Count Estimation

The number of event proposals is often heuristically determined, for example by applying manual thresholds on confidence scores, choosing the top k events, non-maximum suppression (NMS), et cetera. These methods introduce a lot of design issues, assumptions and hyperparameters, and the authors label these as *hand-crafted components* [34]. An unreliable event count estimation can cause either missing information in captions due to under-estimation, or redundancy in captions due to over-estimation.

Design of Proposal Generators

Most proposal generators use anchors and/or post-processing of events. Again, this brings in more hyperparameters and assumptions.

B.7.3 Proposed Solution

Wang *et al* model the task of dense video captioning as a *set prediction task* [34], i.e. the problem is to predict a set of $\{t_j^s, t_j^e, S_j\}$, where t_j^s and t_j^e are the start and end timestamps of the event j, and S_j is its caption.

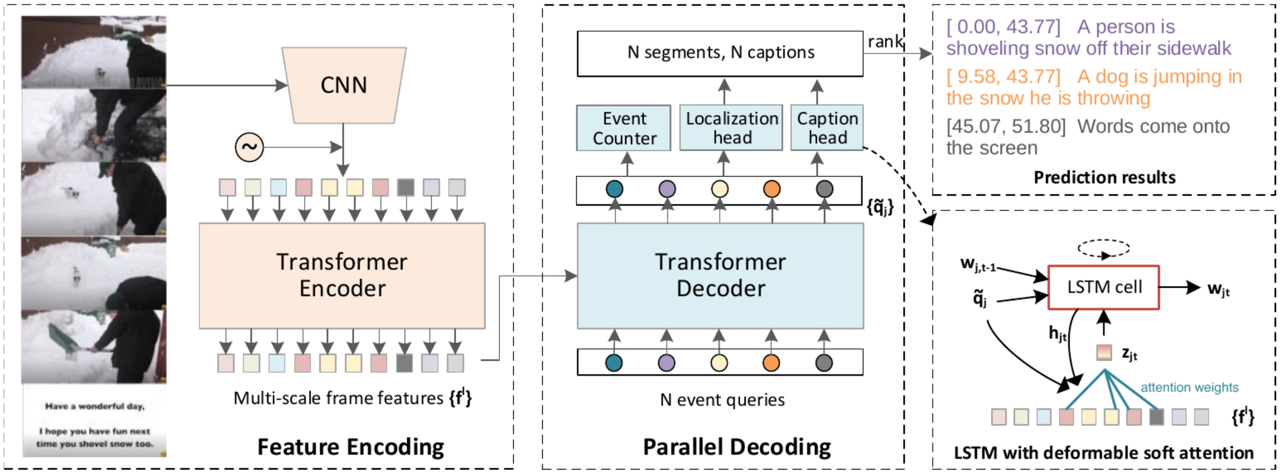

Have a wonderful day,
I hope you have fun next time you shovel snow too.
CNN
Transformer Encoder
Multi-scale frame features {f^l}
Feature Encoding
N segments, N captions
rank
Event Counter
Localization head
Caption head
{q̃_j}
Transformer Decoder
N event queries
Parallel Decoding
[0.00, 43.77] A person is shoveling snow off their sidewalk
[9.58, 43.77] A dog is jumping in the snow he is throwing
[45.07, 51.80] Words come onto the screen
Prediction results
w_j,t-1
q̃_j
LSTM cell
w_jt
h_jt
z_jt
attention weights
{f^l}
LSTM with deformable soft attention

Figure B.11: PDVC Architecture. Image courtesy [34]

This set is of size N_{set}, which is predicted by the *event counter*. The event counter learns to predict the number of events in a video based on video understanding instead of heuristics.

The model performs the subtasks of event localization and caption generation in parallel, making the model a *single-stage pipeline* as opposed to the majority of previous work. This enables the end-to-end training of the entire model, and performance is boosted by mutual promotion of the two tasks.

The authors report localization performance being at par with state-of-the-art, while captioning quality is reported to be better than state-of-the-art at the time.

Feature Encoding

The features are extracted from videos using pretrained video encoders C3D and TSN. Temporal convolutional layers are applied to get *multi-scale features*, to get feature sequences across multiple resolutions [34]. These multi-scale features with positional encoding are given to the encoder of the deformable transformer, which uses *multi-scale deformable attention* to give a context vector as output.

Parallel Decoding

The decoder layers of the deformable transformer, which use multi-scale deformable attention in place of cross-attention (self-attention remaining unchanged as in [1]). The decoder layers query event-level features from the context vector conditioned on N learnable embeddings (termed *event queries*, q_j) and corresponding scalar reference point p_j. An event query q_j serves as an initial guess of event features and p_j serves as that of the center point of the event, and these are refined at each decoder layer. The output representation from decoder layers is given to the following three heads directly and in parallel.

Localization Head

This head performs box prediction with a center and length offset with respect to the reference point, and performs binary classification to output the foreground confidence for each event query. Both these predictions are performed using fully connected layers. The output of this head is $\{t_j^s, t_j^e, c_j^{loc}\}$, where c_j^{loc} is the localization (foreground) confidence.

Captioning Head

The authors propose two variants for the captioning head: (1) a vanilla LSTM and (2) an LSTM using deformable soft attention. The latter uses cues from the combination of caption words and the event features, while the former uses only event features.

Event Counter

This head consists of a max-pooling layer to compress event queries into a global feature vector, which is fed to a fully connected layer to predict vector r_len, from which N_{set} is found by the $argmax$ function.

Finally, the top N_{set} events from N event queries in terms of their overall confidence scores c_j are selected.

$$c_j = c_j^{loc} + \mu \frac{1}{M_j^{\gamma}} \sum_{t=1}^{M_j} log(c_{jt}^{cap})$$

where μ is a balancing factor between localization and captioning confidence, γ is a modulation factor to rectify the influence of caption length M_j.

Set Prediction Loss

The Hungarian algorithm is used to match predicted events with ground truths to find best bipartite matching results. The two sets are the predicted events and the ground truth events. Matching cost is defined as:

$$C = \alpha_{giou} L_{giou} + \alpha_{cls} L_{cls}$$

and the set prediction loss is defined as:

$$L = \beta_{giou} L_{giou} + \beta_{cls} L_{cls} + \beta_{ec} L_{ec} + \beta_{cap} L_{cap}$$

Here L_{giou} is the generalized IoU between predicted temporal segments and ground truth segments, L_{cls} is the focal loss between predicted classifications score and ground truth labels, L_{ce} is the cross-entropy between predicted event count distribution and ground truth and L_{cap} is the cross-entropy loss between predicted word distribution and ground truth (normalized by caption length).

B.8 End-to-End Object Detection with Transformers (DETR)

B.8.1 Overview

Carion *et al* in their 2020 paper titled *End-to-End Object Detection with Transformers* aims to streamline the detection pipeline by effectively removing the need for many hand-designed components by introducing a DEtection TRansformer or DETR which includes set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture.

B.8.2 DETR Model

Paper proposed two essential components for direct set predictions in detection:

- Set prediction loss forces unique matching between ground truth boxes and predicted boxes.
- An architecture that predicts a set of objects and models their relation in a single pass.

Object detection set prediction loss

- loss finds the optimal bipartite match between predicted and ground truth objects and then optimises object-specific (bounding box) losses.
- We pad the ground truth object with no objects if there are fewer objects in it to achieve one-to-one matching for direct set prediction without duplicates.

$$\hat{\sigma} = \arg\max_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} L_{match}(yi, \hat{y}_{\sigma(i)})$$

- The next step is to calculate the Hungarian loss function for all pairings that were matched in the previous stage.

$$L_{Hungarian}(y, \hat{y}) = \sum_{i=0}^{N} [-log\hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{c_i \neq \phi} L_{box}(b_i, \hat{b}_{\hat{\sigma}}(i))]$$

DETR architecture

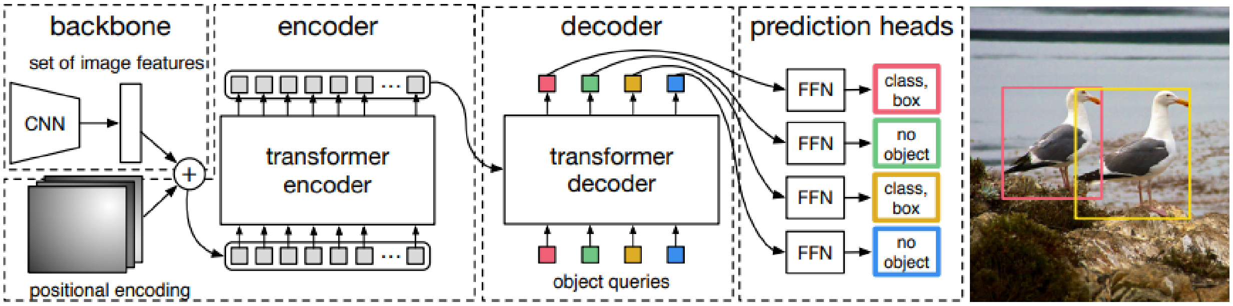

backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
class, box
FFN
no object
FFN
class, box
FFN
no object

Figure B.12: DETR architecture by Carion *et al* (Courtesy [52])

Backbone

- It is a conventional CNN backbone which takes images as input and generates a lower-resolution activation map.

Transformer encoder

- As the encoder expects a sequence as input, spatial dimensions are collapsed into one dimension, resulting in a d×HW feature map.
- Each encoder layer has a standard architecture that includes a multi-head self-attention module and a feed forward network (FFN).
- Since the transformer architecture is permutation-invariant, we supplement it with fixed positional encodings that are added to the input of each attention layer.

Transformer decoder

- The decoder follows the transformer's standard architecture, transforming N embeddings of size d with multi-headed self and encoder-decoder attention mechanisms.
- At each decoder layer, it decodes the N items in parallel. The N input embeddings must be distinct to produce different results because the decoder is permutation-invariant. These input embeddings are called object queries since they are learned positional encodings.
- The decoder converts the N object queries into an output embedding. A feed forward network decodes them independently into box coordinates and class labels, yielding N final predictions.

Prediction feed-forward networks

- The final prediction is computed by a 3-layer perceptron with ReLU activation function and hidden dimension d, and a linear projection layer.
- It predicts the box's normalised centre coordinates, height, and width in relation to the input image, whereas the linear layer uses a softmax function to predict the class label.

B.8.3 Conclusion

Carion *et al* present DETR, a new design for object detection systems based on transformers and bipartite matching loss for direct set prediction. On the challenging COCO dataset, the technique produces results that are comparable to an optimised Faster R-CNN baseline. DETR is easy to set up and maintain, with a modular design that allows for easy expansion and competitive outcomes. Furthermore, it outperforms Faster R-CNN on huge objects.

B.9 Deformable DETR: Deformable Transformers for End-to-End Object Detection

B.9.1 Overview

Zhu *et al* in their 2021 paper titled *Deformable DETR: Deformable Transformers* for End-to-End Object Detection aims to replace the vanilla dense attention, which is the main computational bottleneck in DETR, with a deformable attention module. This can significantly reduce computational costs while also improving convergence.

B.9.2 Architecture

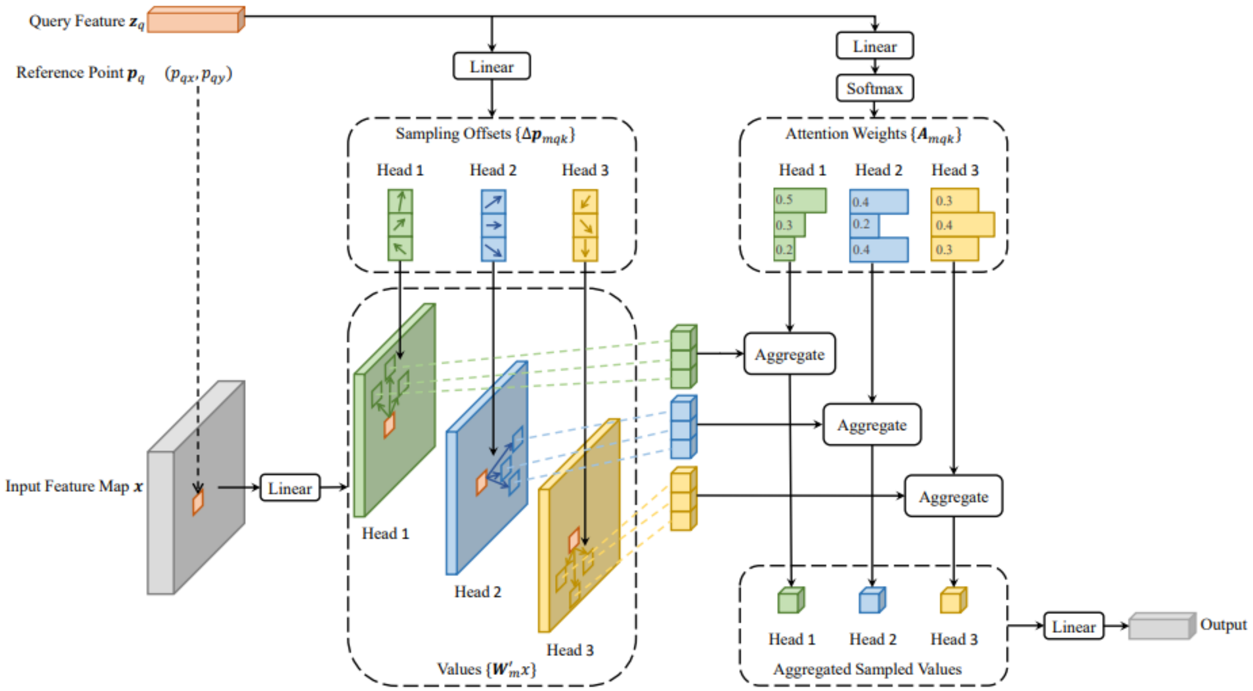

Query Feature z_q
Reference Point p_q (p_{qx}, p_{qy})
Linear
Linear
Softmax
Sampling Offsets $\{\Delta p_{mqk}\}$
Head 1
Head 2
Head 3
Attention Weights $\{A_{mqk}\}$
Head 1
Head 2
Head 3
0.5
0.3
0.2
0.4
0.2
0.4
0.3
0.4
0.3
Aggregate
Aggregate
Aggregate
Input Feature Map x
Linear
Head 1
Head 2
Head 3
Values $\{W'_m x\}$
Head 1
Head 2
Head 3
Aggregated Sampled Values
Linear
Output

Figure B.13: Deformable Attention Module by Zhu *et al* (Courtesy [53])

Deformable Attention Module

- It attends to only a small set of key sampling points around a reference point, regardless of the spatial size of the feature maps. By assigning only a small fixed number of keys for each query, the issues of convergence and feature spatial resolution can be mitigated.
- Given an input feature map $x \in R^{C \times H \times W}$, let q index a query element with content feature zq and a 2-d reference point pq, the deformable attention feature is calculated by

$$DeformAttn(z_q, p_q, x) = \sum_{m=1}^{M} W_m [\sum_{k=1}^{K} A_{mqk} \cdot W'_m x(p_q + \Delta p_{mqk})]$$

  where k indexes the sampled keys, m indexes the attention head, and K is the total sampled key number. Δp_{mqk} and A_{mqk} denote the sampling offset and attention weight of the k^{th} sampling point in the m^{th} attention head, respectively.

Multi-scale Deformable Attention Module

- Deformable attention module can be easily extended for multi-scale feature maps.
- Let $\{x^l\}_{l=1}^{L}$ be the input multi-scale feature maps, where $x^l \in R^{C\times H^l \times W^l}$. Let $\hat{p}_q \in [0,1]^2$ be the normalised coordinates of the reference point for each query element q, then the multi-scale deformable attention module is applied as

  $$DeformAttn(z_q, \hat{p}_q, \{x^l\}_{l=1}^{L}) = \sum_{m=1}^{M} W_m[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlqk} \cdot W'_m x^l(\phi_l(\hat{p}_q) + \Delta p_{mlqk})]$$

  where l indexes the input feature level, m indexes the attention head, and k indexes the sampling point. Δp_{mlqk} and A_{mlqk} denote the sampling offset and attention weight of the kth sampling point in the lth feature level and the mth attention head, respectively

Deformable Transformer Encoder

- The proposed multiscale deformable attention module replaces the transformer attention modules that process feature maps in DETR.
- The encoder's input and output are both multi-scale feature maps with the same resolutions.

Deformable Transformer Decoder

- The decoder includes cross-attention and self-attention modules.
- Object queries in the cross attention modules extract features from feature maps, where the key elements are from the encoder's output feature maps. Object queries interact with each other in the self-attention modules, where the key elements are of the object queries.
- Because the proposed deformable attention module is intended to process convolutional feature maps as key elements, the cross-attention module is replaced with the multi-scale deformable attention module, while the self-attention modules remain unaltered.

B.9.3 Conclusion

Zhu *et al* present Deformable DETR, an efficient and fast-converging end-to-end object detector. The multi-scale deformable attention modules, which are an efficient attention mechanism in processing image feature maps, are at the core of Deformable DETR. .

B.10 Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity

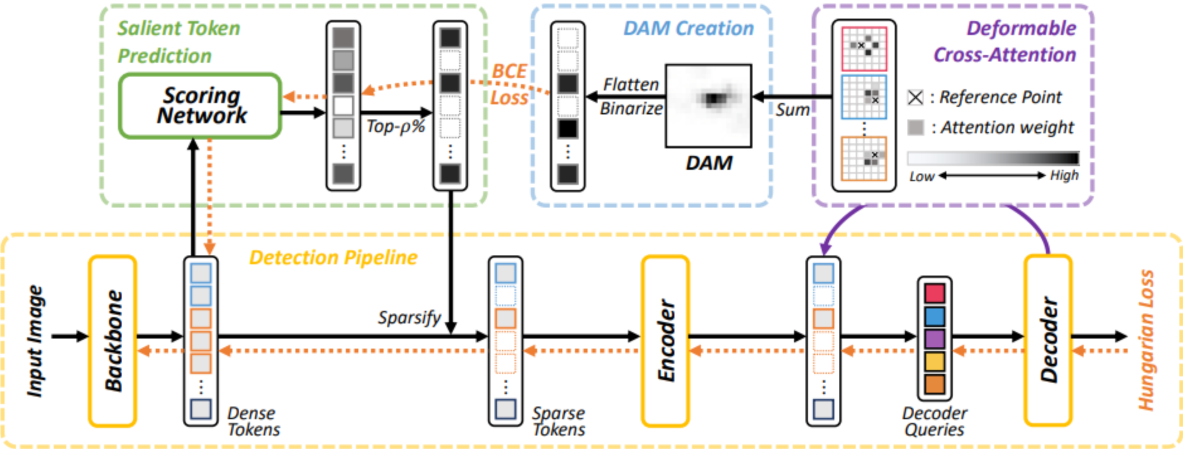
B.10.1 Overview

Roh *et al* in their 2021 paper *Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity* aims to solve the bottleneck of Deformable DETR by sparsifying the encoder token by using DAM (Decoder Attention Map) predictor.

B.10.2 Methodology

Encoder Token Sparsification

In token sparsification scheme, the encoder module selectively refines a small number of encoder tokens. This encoder token subset is obtained from the backbone feature map xfeat with a certain criterion like Decoder Cross-Attention Map (DAM). For features that are not updated in this process, the values of xfeat are passed through the encoder layers without being changed



Salient Token Prediction
Scoring Network
Top-ρ%
BCE Loss
DAM Creation
Flatten
Binarize
DAM
Sum
Deformable Cross-Attention
: Reference Point
: Attention weight
Low
High
Detection Pipeline
Input Image
Backbone
Dense Tokens
Sparsify
Sparse Tokens
Encoder
Decoder Queries
Decoder
Hungarian Loss

Figure B.14: Decoder cross Attention Map (DAM) by Rho *et al* (Courtesy [54])

Decoder Cross-Attention Map

- A cross attention map from the transformer decoder is used to find the salient tokens.
- A scoring network is used to determine which encoder tokens should be further modified on the fly by predicting a pseudo groundtruth of the saliency defined by decoder cross-attention maps.
- The saliency of each input token is determined by aggregating the decoder cross-attentions between all object queries and the encoder output. It generates a single map of the same size as the backbone's feature map, which is known as the Decoder cross Attention Map (DAM).
- To train the scoring network, we binarize DAM so that only the top-k percentage of encoder tokens are kept. This is because the goal is to find a small subset of encoder tokens that the decoder references the most, rather than predicting how many times each encoder token will be referenced by the decoder.
- To predict how likely a given encoder token is to be included in the top-p % most referenced tokens, a 4-layer scoring network g is used, and the network is trained by minimising the binary cross entropy (BCE) loss between the binarized DAM and prediction.

$$L_{dam} = -\frac{1}{N}\sum_{i=1}^{N} BCE(g(X_{feat})_i, DAM_i^{bin})$$

Encoder Auxiliary Loss

Auxiliary loss can be applied to the encoder layers without sacrificing too much computational effort because the sparse detr model has a sparsified token set. Apart from better efficiency and performance, the encoder auxiliary loss has another advantage: it allows to stack more encoder layers safely without failing to converge.

B.10.3 Conclusion

Roh *et al* propose the encoder token sparsification method, which reduces the encoder's attention complexity. They also propose a new sparsification criterion for selecting the informative subset from the complete token set which is Decoder Cross-Attention Map (DAM). Even when just 10% of the encoder token is used, Sparse DETR outperforms Deformable DETR and reduces overall computation by 38.

List of Figures

1.1 A Convolutional Network . . . 10
1.2 The Transformer architecture. Image courtesy [1] . . . 11

2.1 Evolution of dense video captioning methods over time. . . . 17

3.1 Proposed model for Dense Video Captioning . . . 21
3.2 ViViT encoder by Dosovitskiy *et al*. (Image courtesy [39]) . . . 22
3.3 AST encoder by Gong *et al*. (Image courtesy [40]) . . . 23
3.4 Distillation through attention by Touvran *et al*. (Image courtesy [42]) . . . 23

4.1 ViViT encoder by Dosovitskiy *et al*. (Image courtesy [39]) . . . 26
4.2 AST encoder by Gong *et al*. (Image courtesy [40]) . . . 27
4.3 The proposed feature extraction framework with ViViT and AST backbones. TSP heads for action classification and temporal region classification are shown on the right. . . . 29

B.1 Pipeline introduced by Krishna *et al* (Image courtesy [2]) . . . 34
B.2 Dense Video Captioning Framework introduced by Iashin *et al* (Image courtesy [6]) . . . 35
B.3 Bi-modal Multiheaded Proposal Generator: Architecture (Image courtesy [6] . . . 37
B.4 Video Transformer introduced by Dosovitskiy *et al* (Image courtesy [41]) . . . 39
B.5 Video Transformer introduced by Dosovitskiy *et al* (Courtesy [41]) . . . 39
B.6 Audio Spectrogram Transformer by Gong *et al* (Courtesy [40]) . . . 41
B.7 Video Transformer introduced by Arnab *et al* (Courtesy [39]) . . . 44
B.8 Video Transformer introduced by Arnab *et al* (Image courtesy [39]) . . . 45
B.9 The TSP framework. Image courtesy [33] . . . 47
B.10 Two-stage architecture versus proposed single-stage PDVC architecture. Image courtesy [34] . 48
B.11 PDVC Architecture. Image courtesy [34] . . . 49
B.12 DETR architecture by Carion *et al* (Courtesy [52]) . . . 51
B.13 Deformable Attention Module by Zhu *et al* (Courtesy [53]) . . . 53
B.14 Decoder cross Attention Map (DAM) by Rho *et al* (Courtesy [54]) . . . 55

List of Tables

1.1 Datasets and their characteristics 14

2.1 Performance comparison of previous methods on the ActivityNet Captions dataset (* some videos unavailable) 19

Bibliography

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

[2] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*, 2017.

[3] Nayyer Aafaq, Ajmal Mian, Wei Liu, Syed Zulqarnain Gilani, and Mubarak Shah. Video description. *ACM Computing Surveys*, 52(6):1–37, Jan 2020.

[4] Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei. Jointly localizing and describing events for dense video captioning, 2018.

[5] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. Bidirectional attentive fusion with context gating for dense video captioning, 2018.

[6] Vladimir Iashin and Esa Rahtu. A better use of audio-visual cues: Dense video captioning with bi-modal transformer, 2020.

[7] Huijuan Xu, Boyang Li, Vasili Ramanishka, Leonid Sigal, and Kate Saenko. Joint event detection and description in continuous video streams, 2018.

[8] Chaorui Deng, Shizhe Chen, Da Chen, Yuan He, and Qi Wu. Sketch, ground, and refine: Top-down dense video captioning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 234–243, 2021.

[9] Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. *arXiv preprint arXiv:1804.00819*, 2018.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation, 2002.

[11] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries, 2004.

[12] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments, 2005.

[13] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation, 2015.

[14] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances, 2015.

[15] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation, 2016.

[16] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016.

[17] Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards automatic learning of procedures from web instructional videos, 2017.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[20] Yilei Xiong, Bo Dai, and Dahua Lin. Move forward and tell: A progressive generator of video descriptions, 2018.

[21] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks, 2017.

[22] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection, 2017.

[23] Xuguang Duan, Wenbing Huang, Chuang Gan, Jingdong Wang, Wenwu Zhu, and Junzhou Huang. Weakly supervised dense event captioning in videos, 2018.

[24] Jonghwan Mun, Linjie Yang, Zhou Ren, Ning Xu, and Bohyung Han. Streamlined dense video captioning, 2019.

[25] Tanzila Rahman, Bicheng Xu, and Leonid Sigal. Watch, listen and tell: Multi-modal weakly supervised dense event captioning, 2019.

[26] Maitreya Suin and A. N. Rajagopalan. An efficient framework for dense video captioning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12039–12046, Apr. 2020.

[27] Vladimir Iashin and Esa Rahtu. Multi-modal dense video captioning, 2020.

[28] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[29] Teng Wang, Huicheng Zheng, and Mingjing Yu. Dense-captioning events in videos: Sysu submission to activitynet challenge 2020, 2020.

[30] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator, 2019.

[31] Aman Chadha, Gurneet Arora, and Navpreet Kaloty. iperceive: Applying common-sense reasoning to multi-modal dense video captioning and video question answering, 2020.

[32] Shaoxiang Chen and Yu-Gang Jiang. Towards bridging event captioner and sentence localizer for weakly supervised dense event captioning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8421–8431, 2021.

[33] Humam Alwassel, Silvio Giancola, and Bernard Ghanem. Tsp: Temporally-sensitive pretraining of video encoders for localization tasks, 2021.

[34] Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. End-to-end dense video captioning with parallel decoding, 2021.

[35] Hassan Akbari, Linagzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text, 2021.

[36] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset, 2018.

[37] Lorenzo Torresani Du Tran, Heng Wang and Matt Feiszli. Video classification with channel-separated convolutional networks, 2019.

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition, 2017.

[39] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer, 2021.

[40] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[41] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[42] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[43] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017.

[44] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text, 2015.

[45] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks, 2015.

[46] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks, 2016.

[47] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[48] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[49] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Large scale learning of general visual representations for transfer. *CoRR*, abs/1912.11370, 2019.

[50] Qizhe Xie, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves imagenet classification. *CoRR*, abs/1911.04252, 2019.

[51] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

[52] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[53] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2020.

[54] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity, 2021.