National Technical University of Athens
School of Electrical and Computer Engineering
Speech and Language Processing Group

Self-Attention Based Generative Adversarial Networks for Unsupervised Video Summarization

Interdepartmental Graduate Studies Program
Data Science and Machine Learning

MSc Diploma Thesis

of

MARIA NEKTARIA MINAIDI

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Athens, November 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Speech and Language Processing Group

Self-Attention Based Generative Adversarial Networks for Unsupervised Video Summarization

Interdepartmental Graduate Studies Program
Data Science and Machine Learning

MSc Diploma Thesis

of

MARIA NEKTARIA MINAIDI

Supervisor: Alexandros Potamianos
Associate Professor, NTUA

Approved by the examination committee on 8th November 2022.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Alexandros Potamianos	Constantinos Tzafestas	Georgios Siolas
Associate Professor, NTUA	Associate Professor, NTUA	Laboratory Teaching Staff, NTUA

Athens, November 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Speech and Language Processing Group

(Signature)

.....
Maria Nektaria Minaidi
8th November 2022

Περίληψη

Στην παρούσα διπλωματική εργασία καταπιανόμαστε με το θέμα της αυτόματης παραγωγής περίληψης βίντεο, στηριζόμενοι στην μη-επιβλεπόμενη μάθηση και στα δίκτυα προσοχής. Στην σημερινή εποχή ο όγκος των δεδομένων που παράγονται καθημερινά αυξάνεται με εκθετικό ρυθμό. Δεδομένης αυτής της αύξησης, η ανάγκη που υπάρχει προκειμένου οι χρήστες να επιλέγουν, να περιηγηθούν και να καταναλώνουν εκτεταμένες συλλογές βίντεο, αλλά και η αποτελεσματική αποθήκευση του μεγάλου όγκου δεδομένων, ολοένα και αξάνεται. Για την κάλυψη των συγκεκριμένων αναγκών κρίνεται αναγκαία και διερευνάται η αυτόματη παραγωγή περιλήψης βίντεο, στόχος της οποίας αποτελεί η δημιουργία ενός χρονικά συντομότερου βίντεο με είσοδο το αρχικό.

Δεδομένης της πρόσφατης ανάπτυξης που έχουν γνωρίσει τα νευρωνικά δίκτυα, έχουν προταθεί τα τελευταία χρόνια πολλές αρχιτεκτονικές περίληψης βίντεο, στηριζόμενες σε βαθιά νευρωνικά δίκτυα. Στην συγκεκριμένη εργασία, αντιμετωπίζουμε την περίληψη βίντεο ως πρόβλημα επιλογής των βασικότερων και πιο χαρατηριστικών πλάνων (ακολουθία διαδοχικών καρέ) και χρησιμοποιούμε τεχνικές βαθιάς μάθησης και παραγωγικά ανταγωνιστικά δίκτυα προκειμένου να δημιουργήσουμε ένα μοντέλο που συνοψίζει αποτελεσματικά τα εισερχόμενα βίντεο. Το οπτικό περιεχόμενο του εκάστοτε βίντεο, μοντελοποιείται ως ένα διάνυσμα χαρηκτηριστικών της οπτικής πληροφορίας του κάθε καρέ.

Αρχικά, με κίνητρο να ξεπεράσουμε τα μειονεκτήματα των Νευρωνικών Δικτύων Μακράς και Βραχείας Μνήμης, καθώς και να αξιοποιήσουμε τα προτερήματα των μηχανισμών προσοχής, χτίζουμε το μοντέλο μας επεκτείνοντας ένα απλό παραγωγικό ανταγωνιστικό δίκτυο, ενσωματώνοντας σε αυτό μηχανισμούς προσοχής σε διαφετερικά μέρη της αρχιτεκτονικής. Έπειτα, εκτελώντας ένα σύνολο πειραμάτων στα μοντέλα που προκύπτουν, προσδιορίζουμε την σημασία που έχει η προσθήκη της προσοχής και η βελτίωση της χρονικής μοντελοποίησης των καρέ, στην τελική επιλογή των χαρακτηριστικών πλάνων και στην βελτίωση της αποτελεσματικότητας του συστήματος.

Τέλος, αξιολογούμε τα παραπάνω μοντέλα σε δύο ευρέως διαδεδομένα σύνολα δεδομένων, τα οποία αποτελούνται από μικρού μήκους βίντεο και έχουν χρησιμοποιηθεί εκτενώς για την εκπαίδευση και αξιολόγηση μοντέλων περίληψης βίντεο. Επιπροσθέτως, στηριζόμενοι σε μία ακόμη βάση δεδομένων, δημιουργούμε ένα επιπλέον σύνολο δεδομένων, το οποίο αποτελείται από βίντεο μεγαλύτερης διάρκειας και στο οποίο αξιολογούμε τα μοντέλα μας. Η ικανότητα γενίκευσης του μοντέλου μας, καθώς και η χρήση μηχανισμών προσοχής, κρίνονται αποτελεσματικά σε κάθε περίπτωση, καθώς τα αποτελέσματα δείχνουν ότι οι μηχανισμοί προσοχής ως μηχανισμοί επιλογής καρέ, υπερέχουν των σύγχρονων μεθόδων περίληψης βίντεο στα σύνολα δεδομένων SumMe και TVSum.

Keywords

Βαθιά Μάθηση, Μη-Επιβλεπόμενη Περίληψη Βίντεο, Παραγωγικά Ανταγωνιστικά Δίκτυα, Μηχανισμός Προσοχής, Νευρωνικά Δίκτυα Μακράς και Βραχείας Μνήμης, Βαθιά Νευρωνικά Δίκτυα.

Abstract

In this diploma thesis we tackle the topic of video summarization based on unsupervised learning and attention networks. In today's era, the amount of data that is generated on a daily basis is increasing at an exponential rate. Given this growth, the need for users to select, browse, and consume such extensive collections of videos, as well as efficiently store the large amounts of data, is increasing. In order to meet these needs, automatic video summarization, which aims to provide a short visual summary of an original, full-length video, is considered necessary and is being researched.

Given the recent development of neural networks, many video summarization architectures based on deep neural networks have been proposed in the recent years. In this work, we tackle video summarization as a problem of selecting the most characteristic key-shots (sequence of consecutive frames) and use deep learning techniques and generative adversarial networks to build a model that efficiently summarizes the input videos. The visual content of each video is modeled as a feature vector of the visual information of each frame.

Firstly, motivated by the desire to overcome the disadvantages of Long Short-Term Memory Networks, as well as to exploit the advantages of attention mechanisms, we build our model by extending a simple generative adversarial network, incorporating into it attention mechanisms in different parts of the architecture. Then, by running a set of experiments on the resulting models that act as an ablation study, we determine the importance of incorporating attention and improving the temporal modeling of the frames, for the selection of key-shots and improving the efficiency of our architecture.

Finally, we evaluate the above models on two popular datasets, which consist of short videos and have been extensively used to train and evaluate video summarization models. Additionally, relying on one more database, we create an additional dataset, consisting of longer videos, on which we evaluate our models. The generalizability of our model, as well as the use of attention mechanisms, are judged effective in each case, as the results showcase that using self-attention mechanisms as the frame selection mechanism outperforms the state-of-the-art approaches on SumMe and TVSum.

Keywords

Deep Learning, Unsupervised Video Summarization, Generative Adversarial Networks, Long Short-Term Memory Networks, Deep Neural Networks.

to my parents

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον καθηγητή κ. Αλέξανδρο Ποταμιάνο για την επίβλεψη της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να συυνεργαστώ μέσω αυτής με το εργαστήριο Επεξεργασίας Φωνής και Φυσικής Γλώσσας. Οι γνώσεις, οι συμβουλές και τα ερεθίσματα που μου έδωσε η ομάδα, αποτέλεσαν καθοριστικά στην εκπόνηση της εργασίας μου.

Στη συνέχεια, θέλω να ευχαριστήσω ιδιαίτερα και τους διδακτορικούς ερευνητές της ομάδας του κ. Ποταμιάνου και ειδικότερα τον Χάρη Παπαϊωάνου, ο οποίος είχε πάντα τη διάθεση να με κατευθύνει και του οποίου η συμβολή ήταν καθοριστική στην παρούσα εργασία.

Ένα μεγάλο ευχαριστώ οφείλω, επίσης, στην οικογένεια μου, τους γονείς μου και τον αδερφό μου, για την συνεχή στήριξη, εμπιστοσύνη και αγάπη που μου δείχνουν. Τους είμαι ευγνώμων και νιώθω πραγματικά τυχερή. Τέλος, θα ήθελα να πω ένα ακόμη μεγάλο ευχαριστώ στους φίλους μου, τους ανθρώπους που είναι πάντα δίπλα μου.

Αθήνα, Νοέμβρης 2022

Μαρία Νεκταρία Μηναΐδη

Table of Contents

List of Figures

List of Tables

Εκτεταμένη Ελληνική Περίληψη

Εισαγωγή

Στην σημερινή εποχή ο όγκος και το είδος των πληροφοριών που καταναλώνουν οι χρήστες καθημερινά, αυξάνεται με μεγάλη ταχύτητα. Ως εκ τούτου, κρίνεται ζωτικής σημασίας να αναπτυχθούν συστήματα τα οποία έχουν την δυνατότητα να βοηθήσουν τους χρήστες στο να αλληλεπιδρούν με μεγάλες ποσότητες πληροφορίας, όσο το δυνατόν πιο αποτελεσματικά. Το βίντεο αποτελεί την πιο δημοφιλή πηγή πληροφορίας με την οποία αλληλεπιδρά η πλειοψηφία των χρηστών του διαδικτύου και η ζήτηση που έχει είναι μεγάλη. Ως άνθρωποι, άλλωστε, απολαμβάνουμε την αφήγηση μίας ιστορίας και την ευακρία της μεγαλύτερης συναισθηματικής σύνδεσης που αυτή προσφέρει.

Η περίληψη βίντεο ορίζεται ως η διαδικασία παραγωγής μιας σύνοψης ενός αρχικού βίντεο, η οποία διατηρεί τα πιο σημαντικά νοηματικά σημεία του αρχικού βίντεο και την ομαλή ροή της ιστορίας, ενώ ταυτόχρονα δίνει τη δυνατότητα στον χρήστη να έχει άμεση πρόσβαση στα καίρια αυτά σημεία. Κρίνεται σημαντικό η περίληψη που παραγεται να περιέχει όσο το δυνατόν λιγότερη περιττή πληροφορία [14]. Η περίληψη βίντεο στοχεύει στην κάλυψη των ολοένα αυξανόμενων αναγκών των χρηστών του διαδικτύου και οι εφαρμογές της, εκτός από την αποτελεσματική περιήγηση και ανάκτηση πληροφοριών, περιλαμβάνουν τη σύνοψη βίντεο από κάμερες παρακολούθησης, ιατρικών βίντεο, βίντεο που τραβήχτηκαν από μη επανδρωμένα εναέρια οχήματα [15], την δημιουργία αποτελεσματικών ευρετηρίων κ.α.

Η συγκεκριμένη ερευνητική περιοχή παραμένει αρκετά ενεργή και τα τελευταία χρόνια έχουν αναπτυχθεί πολλά μοντέλα βαθιάς μηχανικής μάθησης τα οποία αφορούν την αυτόματη περίληψη βίντεο. Οι μέθοδοι αυτοί βαθιάς μηχανικής μάθησης, ξεπερνούν, όσον αφορά την επίδοση, πολλές κλασικές μεθόδους, όπως είναι τεχνικές που στηρίζονται στις αραιές αναπαραστάσεις ή σε αλγόριθμους ομαδοποίησης. Αυτόματη περίληψη ενός βίντεο αποτελεί το προκύπτον σύνολο είτε από αντιπροσωπευτικά καρέ, στην περίπτωση του οποίου έχουμε την στατική περίληψη, είτε από αντιπροσωπευτικές σκηνές, όπου και έχουμε την δυναμική περίληψη βίντεο. Και στις δύο περιπτώσεις, τα αντιπροσωπευτικά καρέ διατάσσονται στην περίληψη με χρονολογική σειρά. Επιπλέον, ανάλογα με τον τύπο των δεδομένων που χρησιμοποιεί μια μέθοδος για την εξαγωγή της περίληψης, υπάρχουν μονοτροπικές και πολυτροπικές προσεγγίσεις. Οι μονοτροπικές προσεγγίσεις χρησιμοποιούν μόνο την οπτική πληροφορία του βίντεο για την εξαγωγή των χαρακτηριστικών του καρέ, ενώ οι πολυτροπικές προσεγγίσεις εκμεταλλεύονται επίσης και τα διαθέσιμα μεταδεδομένα ήχου ή/και κειμένου.

Στην παρούσα διπλωματική εργασία, με κίνητρο τα παραπάνω, διερευνούμε τις προκλήσεις και τις ευκαιρίες που παρουσιάζονται κατά τη δημιουργία μοντέλων τεχνητής νοημοσύνης που στοχεύουν στην αυτόματη περίληψη βίντεο. Για το σκοπό αυτό, μελετάμε την περίληψη βίντεο

ως ένα πρόβλημα βαθιάς μηχανικής μάθησης και προετείνουμε ένα μονοτροπικό σύστημα, το οποίο στηρίζεται στην μη-επιβλεπόμενη μάθηση και βασίζεται σε ένα παραγωγικό ανταγωνιστικό δίκτυο, καθώς επίσης και σε μηχανισμούς προσοχής, προκειμένου να πετύχει το σκοπό της εξαγωγής της περίληψης από τα βίντεο, ως ένα σύνολο αντιπροσωπευτικών καρέ. Πιο συγκεκριμένα οι κύριες συνεισφορές μας έχουν ως εξής:

- Διερευνούμε την αποτελεσματικότητα της ενσωμάτωσης μηχανισμών προσοχής και transformers σε διάφορα μέρη της αρχιτεκτονικής SUM-GAN [2], η οποία βασίζεται σε ένα παραγωγικό ανταγωνιστικό δίκτυο (GAN). Ειδικότερα, εκπαιδεύουμε έναν αριθμό διαφορετικών μοντέλων, χρησιμοποιώντας έναν μετασχηματιστή ως επιλογέα καρέ (SUM-GAN-SAT), κωδικοποιητή (SUM-GAN-STD, SUM-GAN-STSED) και κωδικοποιητή-αποκωδικοποιητή (SUM-GAN-SAT, SUM-GAN-ST).

- Με βάση τα παραπάνω αποτελέσματα, προτείνουμε τη χρήση ενός μηχανισμού αυτοπροσοχής ως επιλογέα καρέ, διατηρώντας παράλληλα την αρχιτεκτονική βασισμένη σε Δίκτυα Μακράς Βραχύχρονης Μνήμης (LSTM) για τον κωδικοποιητή και τον αποκωδικοποιητή (SUM-GAN-AED).

- Αξιολογούμε τα μοντέλα μας σε δύο δημοφιλή σύνολα δεδομένων, SumMe [16] και TVSum [17]. Επιτυγχάνουμε κορυφαίες επιδόσεις στο SumMe και ανταγωνιστικά αποτελέσματα, σε σχέση με τα κορυφαία της βιβλιογραφίας στο TVSum.

- Επεκτείνουμε την αξιολόγησή των μοντέλων μας σε ένα ακόμη σύνολο δεδομένων, το οποίο δημιουργήσαμε στηριζόμενοι στην βάση δεδομένων COGNIMUSE [18] και στο οποίο το μοντέλο μας πετυχαίνει επίσης ανταγωνιστική απόδοση.

Θεωρητικό Υπόβαθρο

Μηχανισμός Προσοχής

Η διάδοση της πληροφορίας μέσα από μία σειρά αναδρομικών συνδέσεων έχει ως επίπτωση την απώλεια μέρους αυτής και η ακολουθιακή φύση τους δυσχαιρένει τον παράλληλο υπολογισμό. Μία σύγχρονη λύση σε αυτό το πρόβλημα εμφανίστηκε μέσω του Μηχανισμού της Προσοχής, ο οποίος προτάθηκε από τους Bahdanau et al. [7]. Η σημαντικότερη ιδιότητα του μηχανισμού είναι ότι βοηθάει τα μοντέλα να δώσουν περισσότερη προσοχή σε συγκεκριμένα σημεία κατά την επεξεργασία των δεδομένων.

Αυτό-Προσοχή Η Αυτο-Προσοχή είναι ένας μηχανισμός προσοχής που μπορεί να εφαρμοστεί σε μία πρόταση, για την οποία δεν υπάρχει κάποια επιπλέον πληροφορία και απεικονίζεται στο Σχήμα 1. Σε αντίθεση με τα ΑΝΔ, οι υπολογισμοί σε κάθε χρονικό βήμα είναι ανεξάρτητοι όλων των υπολοίπων βημάτων και επομένως μπορούν να γίνουν παράλληλα.

Κάθε είσοδος x_i μπορεί να παίξει τρεις διαφορετικούς ρόλους στην διαδικασία. Μπορούμε με βάση αυτό, να δώσουμε τους ορισμούς των "query", "key" και "value" που αποτελούν και τις πιο σημαντικές έννοιες του μηχανισμού της προσοχής:

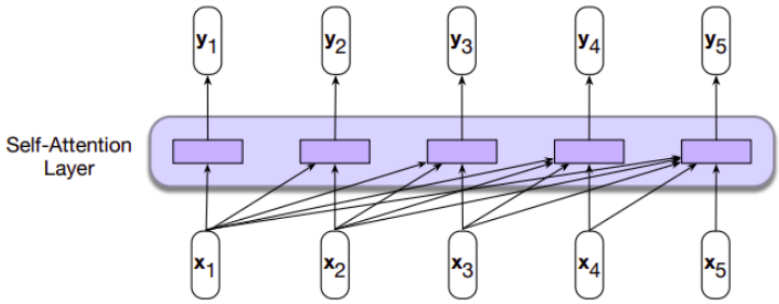

Figure 1. *Η ροή της πληροφορίας σε έναν μηχανισμό Αυτο-Προσοχής. Πηγή: [1]*

1. **query** είναι το στοιχείο-στόχος που συγκρίνεται με κάθε στοιχείο εισόδου που προηγείται αυτού.

2. **key** είναι κάθε στοιχείο εισόδου που προηγείται του στοιχείου-στόχου και συγκρίνεται με αυτό.

3. **value** είναι το στοιχείο που χρησιμοποιείται για τον υπολογισμό της τελικής εξόδου.

Ο μηχανισμός αυτό προσοχής δίνεται από τον εξής τύπο:

$$SelfAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{1}$$

όπου οι πίνακες Q, K και V αναφέρονται στο query, key και value, αντίστοιχα.

Μετασχηματιστής Ο Μετασχηματιστής [8] είναι μια αρχιτεκτονική βαθιάς μηχανικής μάθησης που εισήχθη για να λύσει το πρόβλημα της αυτόματης μετάφρασης. Είναι εμπνευσμένος από την επιτυχία του μηχανισμού της προσοχής και είναι σε έση να επεξεργάζεται δεδομένα παράλληλα. Το γεγονός αυτό τον καθιστά γρήγορο και αποδίδει καλύτερα από τις ανατροφοδοτούμενες αρχιτεκτονικές. Πρόκειται για μια αρχιτεκτονική κωδικοποιητή και αποκωδικοποιητή, οι οποίοι αποτελούνται από πολλά στοιβαγμένα ίδια επίπεδα. Κάθε επίπεδο του κωδικοποιητή αποτελείται από υπο-επίπεδα αυτό-προσοχής και δίκτυα πρόσθιας τροφοδότησης δικτύων. Ο αποκωδικοποιητής έχει την ίδια δομή, με την προσθήκη ενός στρώματος διασταυρούμενης προσοχής. Το στρώμα αυτό-προσοχής δημιουργεί μια αναπαράσταση της ακολουθίας με άση τα συμφραζόμενα, αναλύοντας την εξάρτηση μεταξύ των μερών της. Το υπο-επίπεδο διασταυρούμενης προσοχής είναι υπεύθυνο για την ανάλυση της εξάρτησης μεταξύ των ακολουθιών εισόδου και εξόδου. Το αποτέλεσμα του τελευταίου επιπέδου του αποκωδικοποιητή μετατρέπεται τελικά σε πιθανότητες του κάθε στοιχείου της εισόδου, χρησιμοποιώντας έναν γραμμικό μετασχηματισμό και μια συνάρτηση softmax. Ο μετασχηματιστής διατηρεί την πληροφορία για την σχετική θέση των στοιχείων της εισόδου του. Οι πληροφορίες των απόλυτων και σχετικών θέσεων των στοιχείων της ακολουθίας κωδικοποιούνται στις διανυσματικές τους αναπαραστάσεις. Αυτό γίνεται μέσω άθροισης ενός διανύσματος που αντιπροσωπεύει τις θέσεις κάθε εισόδου, που ονομάζεται κωδικοποίηση θέσης και στηρίζεται σε ημιτονοειδείς συναρτήσεις.

Δεδομένα

Τα δεδομένα στα οποία εκπαιδεύονται και ελέγχονται τα μοντέλα μας είναι τα σύνολα δεδομένων SumMe [16], TVSum[17] και COGNIMUSE[18], τα οποία θα παρουσιάσουμε ακολούθως.

Το σύνολο δεδομένων SumMe, δημιουργήθηκε από τους Gygli et al. [16], αποτελείται από 25 βίντεο από διακοπές, εκδηλώσεις και αθλήματα, τραβηγμένα τόσο σε πρώτο πρόσωπο όσο και σε τρίτο. Είναι ακατέργαστα ή ελάχιστα επεξεργασμένα βίντεο. Η διάρκεια των βίντεο κυμαίνεται από 1 έως 6 λεπτά περίπου. Κάθε βίντεο έχει σχολιαστεί από 15 έως 18 χρήστες με τη μορφή περιλήψεων σκηνών από τους χρήστες και έχουν μήκος μεταξύ 5% και 15% της αρχικής διάρκειας βίντεο. Επιπλέον, εκτός από τις προαναφερθείσες περιλήψεις χρηστών, παρέχεται επίσης μια περίληψη που θεωρείται βασική με τη μορφή βαθμολογιών σπουδαιότητας σε επίπεδο σκηνών, που υπολογίζονται ως μέσος όρος των περιλήψεων καρέ.

Το σύνολο δεδομένων TVSum, δημιουργήθηκε από τον Song et al. [17], αποτελείται από 50 βίντεο, διάρκειας από 1 έως 11 λεπτά, και την βαθμολογία σημαντικότητας των σκηνών, που λαμβάνεται μέσω crowdsourcing. Περιέχει βίντεο από 10 κατηγορίες TRECVid Multimedia Event Detection (MED) [19], 5 ανά κατηγορία, που ελήφθησαν από το YouTube, χρησιμοποιώντας την κατηγορία ως όρο της αναζήτησης. Τα βίντεο που συλλέγονται αντιπροσωπεύουν διάφορα είδη, όπως ειδήσεις, οδηγίες χρήσης, ντοκιμαντέρ και περιεχόμενο που δημιουργείται από χρήστες (vlog, εγωκεντρικό). Τα βίντεο TVSum έχουν σχολιαστεί από 20 χρήστες με τη μορφή βαθμολογιών σπουδαιότητας σε επίπεδο σκηνών και καρέ (από 1 έως 5). Παρόμοια με το SumMe, παρέχεται μια ενιαία βασική περίληψη με τη μορφή βαθμολογιών σπουδαιότητας σε επίπεδο καρέ, που υπολογίζονται ως μέσος όρος των βαθμολογιών όλων των χρηστών, για κάθε βίντεο του συνόλου δεδομένων.

Η βάση δεδομένων COGNIMUSE, που δημιουργήθηκε από τους Zlaintsi et al. [18], είναι μια βάση δεδομένων βίντεο, πολυτροπικά σχολιασμένη με αισθητηριακή και σημασιολογική βαρύτητα, ακουστικά και οπτικά συμβάντα, σχέσεις μεταξύ των διάφορων μέσων, καθώς και συναισθήματα. Δημιουργήθηκε ως ένα πλαίσιο που θα βοηθούσε την εκπαίδευση και την αξιολόγηση αλγορίθμων ανίχνευσης σημαντικών γεγονότων και περίληψης βίντεο, καθώς και στην ανάλυση περιεχομένου. Το σύνολο δεδομένων αποτελείται από συνεχόμενα τμήματα 30 λεπτών (με το τελικό πλάνο/σκηνή να περιλαμβάνεται) από επτά ταινίες του Χόλιγουντ (τρεισήμισι ώρες συνολικά), οι οποίες είναι: «A Beautiful Mind» (BMI), «Chicago» (CHI), «Crash» (CRA), «The Departed» (DEP), «Gladiator» (GLA), «Lord of the Rings-the Return of the King» (LOR) και η ταινία κινουμένων σχεδίων «Finding Nemo» (FNE). Περιλαμβάνουν επίσης και βασικές έννοιες, όπως είναι ο κύριος χαρακτήρας/οι, η επιθυμία/στόχος που υπάρχει και η σύγκρουση, καθώς και χαρακτηριστικά όπως η μουσική, οι έντονες χρωματικές παραλλαγές, ηχητικά και οπτικά εφέ, ταχύτητα της δράσης κ.λπ., τα οποία είναι σημαντικά για την ανάπτυξη της πλοκής, και συνεπώς η μοντελοποίησή τους οδηγεί σε αποτελεσματικές περιλήψεις. Αυτά τα επτά τμήματα ταινιών αποτελούν τη βάση για το σύνολο δεδομένων COGNIMUSE που χρησιμοποιείται στη μελέτη μας.

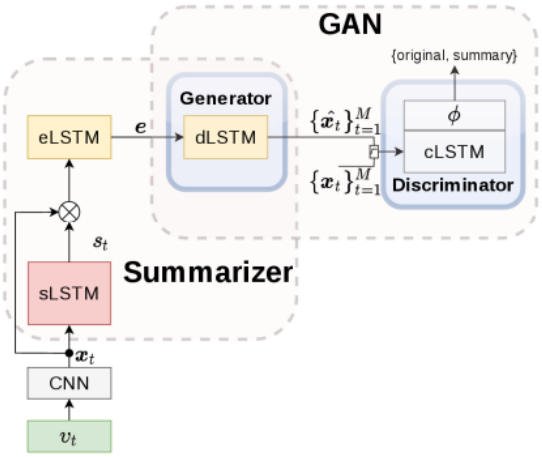

Figure 2. *Η βασική SUM-GAN αρχιτεκτονική. Source: [2]*

Προσεγγίσεις

Βασικό Μοντέλο

Το βασικό πλαίσιο παραγωγικού αντιπαραθετικού δικτύου [2], απεικονίζεται στο σχήμα 2. Ο επιλογέας καρέ, ο κωδικοποιητής και ο αποκωδικοποιητής αποτελούν από κοινού τον Συνοψιστή, ενώ ο αποκωδικοποιητής (γεννήτρια) μαζί με τον διαχωριστή αποτελούν το Παραγωγικό Ανταγωνιστικό Δίκτυο. Ο κωδικοποιητής και ο αποκωδικοποιητής σχηματίζουν επίσης έναν Αυτόματο Κωδικοποιητή Μεταβλητών (VAE), ο οποίος βοηθά στην εκπαίδευση δημιουργώντας μια υποκείμενη αναπαράσταση του βίντεο και εισάγοντας ένα πρόσθετο διάνυσμα βαθμολογιών καρέ [20]. Η προσέγγιση προτείνει έναν μηχανισμό επιλογής βασικών καρέ, που ελαχιστοποιεί την απόσταση μεταξύ των χαρακτηριστικών των αρχικών βίντεο και των βίντεο που προκύπτουν ως ανακατασκευή από τις προβλεπόμενες περιλήψεις. Ο συνοψιστής και ο διαχωριστής εκπαιδεύονται αντιπαραθετικά, χωρίς επίβλεψη, έως ότου ο διαχωριστής δεν μπορεί να διακρίνει τα ανακατασκευασμένα βίντεο από τα πρωτότυπα βίντεο.

Πιο συγκεκριμένα, ο επιλογέας καρέ είναι ένα αμφίδρομο LSTM και ο κωδικοποιητής και ο αποκωδικοποιητής είναι επίσης LSTM. Έστω $\mathbf{X} \in \mathbb{R}^{M \times N}$ τα χαρακτηριστικά καρέ του βίντεο εισόδου, που προέρχονται από το συνελικτικό νευρωνικό δίκτυο (CNN), όπου M είναι ο αριθμός των χαρακτηριστικών των καρέ και N ο αριθμός των καρέ. $\mathbf{x}_i \in \mathbb{R}^M$, $i \in [1, N]$, είναι το διάνυσμα χαρακτηριστικών που περιγράφει το καρέ i^{th}. Σε κάθε βήμα i, ο επιλογέας τροφοδοτείται με το $\mathbf{x}_i$ και μας δίνει ένα κανονικοποιημένο διάνυσμα βαρών $\mathbf{s} \in \mathbb{R}^N$, όπου $\mathbf{s}_i \in [0, 1]$. Τα σταθμισμένα χαρακτηριστικά των καρέ, $\mathbf{x}_i\mathbf{s}_i$, αντιστοιχούν στην περίληψη και τροφοδοτούνται στον κωδικοποιητή, ο οποίος δίνει ένα διάνυσμα κρυφής κατάστασης, $\mathbf{e} \in \mathbb{R}^H$. Ο αποκωδικοποιητής λαμβάνει ως είσοδο το $\mathbf{e}$ και κατασκευάζει μια ακολουθία χαρακτηριστικών που αντιπροσωπεύουν το βίντεο εισόδου $\hat{\mathbf{x}} \in \mathbb{R}^N$. Τέλος, το $\hat{\mathbf{x}}$ προωθείται στον διαχωριστή που βασίζεται στο LSTM, ο οποίος στοχεύει να το ταξινομήσει ως «πρωτότυπο» ή «περίληψη».

Ο μεταβλητός αυτό-κωδικοποιητής (VAE ή Variational Auto Encoder) προτάθηκε από τους Kingma et al. [20] και ορίζει μια μεταγενέστερη κατανομή στα παρατηρούμενα δεδομένα,

δεδομένης μιας λανθάνουσας μεταβλητής. Έστω $\mathbf{e} \sim p_e(\mathbf{e})$ μία προγενέστερη κατανομή της κρυφής μεταβλητής και x τα δεδομένα που παρατηρούνται. Μπορούμε να ερμηνεύσουμε το $\mathbf{e}$ ως την κωδικοποίηση του $\mathbf{x}$ και να ορίσουμε το $q(\mathbf{e}|x)$ ως την πιθανότητα να παρατηρήσουμε ένα στοιχείο $\mathbf{x}$. Είναι τυπικό να ορίζεται το $e \sim p_e(e)$ ως η τυπική κανονική κατανομή. Ομοίως, το $p(\mathbf{x}|\mathbf{e})$ προσδιορίζει την υπό συνθήκη κατανομή για το $\mathbf{x}$, δεδομένου του $\mathbf{e}$. Το VAE εκπαιδεύεται ελαχιστοποιώντας τον αρνητικό λογάριθμο:

$$-log\frac{p(x|e)p(e)}{q(e|x)} = \underbrace{-log(p(x|e))}_{\mathcal{L}_{reconst}} + \underbrace{\mathcal{D}_{KL}(q(e|x)||p(e))}_{\mathcal{L}_{prior}} \tag{2}$$

Οι Mahasseni et al. [2] είναι οι πρώτοι που συνδυάζουν έναν επιλογέα καρέ που βασίζεται σε LSTM με έναν Μεταβλητό Αυτό-Κωδικοποιητή και έναν εκπαιδεύσιμο διαχωριστή.

Η εκπαίδευση του μοντέλου καθορίζει τις παραμέτρους του συνοψιστή, $\{\theta_s, \theta_\epsilon, \theta_d\}$, που αντιστοιχούν στον επιλογέα καρέ, τον κωδικοποιητή και τον αποκωδικοποιητή, και τις παραμέτρους του GAN, $\{\theta_d, \theta_c\}$, που ορίζουν τον αποκωδικοποιητή και τον ταξινομητή. Ορίζονται τέσσερις συναρτήσεις κόσοτυς: του GAN, $\mathcal{L}_{GAN}$, το κόστος ανακατασκευής (Reconstruction Loss), $\mathcal{L}_{reconst}$, η απώλεια αραιότητας (Sparsity Loss) $\mathcal{L}_{sparsity}$ και η προγενέστερη απώλεια (Prior Loss), $\mathcal{L}_{prior}$.

Ας ορίσουμε τώρα καθεμία από τις παραπάνω συναρτήσεις κόσοτυς. Η απώλεια αραιότητας (Sparsity Loss) χρησιμοποιείται ως κανονικοποίηση και τιμωρεί την επιλογή μεγάλου αριθμού βασικών καρέ στη σύνοψη. Συμβολίζεται ως:

$$\mathcal{L}_{sparsity} = \| \frac{1}{N}\sum_{t=1}^{N} s_t - \sigma \|_2 \tag{3}$$

όπου M είναι ο συνολικός αριθμός καρέ και σ είναι ο ρυθμός σύνοψης, μια υπερπαράμετρος που αντιπροσωπεύει το ποσοστό των καρέ που πρόκειται να πειληφθουν στην περίληψη.

Το Προγενέστερο Κόστος (Prior Loss) ορίζεται ως:

$$\mathcal{L}_{prior} = \mathcal{D}_{KL}(q(\mathbf{e}|\mathbf{x})||\mathcal{N}(0,1))) \tag{4}$$

όπου $\mathcal{D}_{KL}$ είναι η απόκλιση Kullback–Leibler και υποδηλώνει ένα μέτρο για το πόσο μια κατανομή είναι διαφορετική από μια δεύτερη.

Το Κόστος Ανακατασκευής (Reconstruction Loss) ορίζεται ως:

$$\mathcal{L}_{reconst} = \mathbb{E}[-\log p(\phi(\mathbf{x})|\mathbf{e})] \tag{5}$$

όπου η μέση τιμή $\mathbb{E}$ προσεγγίζεται ως ο εμπειρικός μέσος όρος των παραδειγμάτων εκπαίδευσης, το $\phi(\mathbf{x})$ είναι η έξοδος του τελευταίου κρυφού στρώματος του διαχωριστή και $p(\phi(\mathbf{x})|\mathbf{e}) \propto exp(- \| \phi(\mathbf{x}) - \phi(\hat{\mathbf{x}}) \|^2)$. Η απώλεια ανακατασκευής διασφαλίζει ότι η σύνοψη διατηρεί τις κύριες πληροφορίες του αρχικού βίντεο και ελαχιστοποιεί την απώλεια πληροφοριών του ανακατασκευασμένου βίντεο.

Το Αντιπαραθετικό Κόστος Loss ορίζεται ως:

$$\mathcal{L}_{GAN} = log(LSTM(\mathbf{x})) + log(1 - LSTM(\hat{\mathbf{x}})) + log(1 - LSTM(\hat{\mathbf{x}}_p)) \quad (6)$$

όπου LSTM(·) είναι η δυαδική έξοδος του ταξινομητή και αντιπροσωπεύει την εμπιστοσύνη του κατά την ταξινόμηση του αρχικού βίντεο, της σύνοψης που δημιουργήθηκε και της βασικής περίληψης.

SUM-GAN

Η ανακατασκευή του βασικού μοντέλο, ονομάζεται SUM-GAN και έχει δημιουργηθεί χρησιμοποιώντας τον κώδικα[1] του [2]. Η αρχιτεκτονική του μοντέλου απεικονίζεται στο σχήμα 2 και υλοποιείται χρησιμοποιώντας Python 3.7 [21] και PyTorch 1.0.1 [22]. Βασιζόμενοι σε αυτόν τον κώδικα, αφαιρέσαμε το στρώμα γραμμικής συμπίεσης που τοποθετείται πριν από τον επιλογέα καρέ και τη σταθερή εκπαίδευση GAN που χρησιμοποιείται. Προσπαθώντας να αναπαραγάγουμε την ακριβή αρχιτεκτονική που περιγράφεται από τους Mahasseni et al. [2], συναντήσαμε την ακόλουθη ασυνέπεια: το αρχικό έγγραφο αναφέρει ότι το κρυφό μέγεθος του κωδικοποιητή και του αποκωδικοποιητή, είναι 2048, ωστόσο το αρχικό μέγεθος εισόδου $\mathbf{x}_t$, πριν από οποιαδήποτε επεξεργασία, είναι 1024, σύμφωνα με την έξοδο του επιπέδου pool5 του δικτύου GoogLeNet[23]. Ο ταξινομητής LSTM λαμβάνει ως είσοδο το $\mathbf{x}_t$ και το $\hat{\mathbf{x}}_t$, κάτι που μας οδηγεί σε ασυνεπές μέγεθος εισόδου. Για να αποφύγουμε αυτή τη σύγκρουση, αποφασίσαμε να μειώσουμε το κρυφό μέγεθος τόσο του κωδικοποιητή όσο και του αποκωδικοποιητή LSTM σε 1024. Αυτή είναι η μόνη αλλαγή που κάναμε, σε σύγκριση με την εκτέλεση του [2] όπως περιγράφεται στην δημοσίευσή του. Εν τέλει, κάθε στοιχείο του δικτύου αποτελείται από ένα LSTM 2 επιπέδων, με 1024 κρυφές μονάδες σε κάθε επίπεδο.

SUM-GAN-LC

Στο SUM-GAN-LC, προσθέτουμε ένα γραμμικό στρώμα συμπίεσης στην αρχιτεκτονική που απεικονίζεται στο σχήμα 2, το οποίο μειώνει τον αριθμό των εκπαιδεύσιμων παραμέτρων. Η μείωση του μεγέθους των διανυσμάτων εισόδου, οδηγεί σε βελτίωση της απόδοσης, όπως σημειώνουν οι Apostolidis et al. [13]. Το γραμμικό στρώμα συμπίεσης μειώνει σημαντικά τον αριθμό των παραμέτρων και βοηθάει την ικανότητα εκπαίδευσης του μοντέλου στην περίπτωση μικρών συνόλων δεδομένων (όπως για το SumMe), ενώ μικρότερη επίδραση παρατηρείται στην περίπτωση μεγαλύτερων συνόλων δεδομένων (όπως για το TVSum). Κάθε στοιχείο της αρχιτεκτονικής αποτελείται από ένα LSTM 2 επιπέδων, με 500 κρυφά στρώματα σε κάθε επίπεδο.

Προτεινόμενα Μοντέλα

Σε αυτή την ενότητα περιγράφουμε λεπτομερώς το προτεινόμενο μοντέλο μας, SUM-GAN-AED, καθώς και τις παραλλαγές του. Παρουσιάζουμε ένα μοντέλο περίληψης βίντεο που βασίζεται στην αρχιτεκτονική SUM-GAN [2]. Βασιζόμαστε σε μια διαθέσιμη υλοποίηση μιας παραλλαγής του συγκεκριμένου μοντέλου και διεξάγουμε τα πειράματα που περιγράφονται στις

[1] https://github.com/j-min/ Adversarial$_V ideo_S ummary$

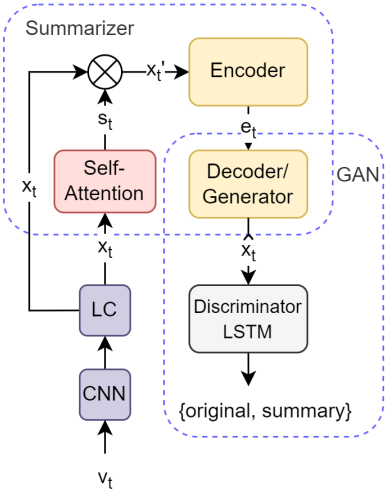

Figure 3. *Η αρχιτεκτονική του μοντέλου SUM-GAN-AED.*

ακόλουθες υποενότητες. Η εκπαίδευση και ο έλεγχος του μοντέλου μας και των παραλλαγών του ακολουθούν αυτή του βασικού SUM-GAN. Επιπλέον, εκπαιδεύουμε τα μοντέλα μας σε τρία σύνολα δεδομένων, SumMe [16], TVSum [17] και COGNIMUSE [18], για κάθε ένα από τα πειράματα.

SUM-GAN-AED

Η προσέγγισή μας εισάγει έναν επιλογέα καρέ στην αρχιτεκτονική GAN, που βασίζεται στον μηχανισμό της προσοχής. Το μοντέλο είναι εμπνευσμένο από τα [24] και [25]. Ο μηχανισμός αυτο-προσοχής καταγράφει αποτελεσματικά τις μακροπρόθεσμες χρονικές εξαρτήσεις, σε αντίθεση με τη χρήση ενός LSTM που δεν τις καταγράφει επαρκώς [26]. Έτσι, οδηγούμαστε σε ταχύτερους υπολογισμούς και σε ένα πιο αντιπροσωπευτικό διάνυσμα βαθμολογιών καρέ [26]. Η αρχιτεκτονική του προτεινόμενου μοντέλου SUM-GAN-AED απεικονίζεται στο σχήμα 3.

Model Variants

Συνεχίζοντας, διερευνούμε περαιτέρω την αποτελεσματικότητα της προσθήκης μηχανισμών αυτοπροσοχής στο μοντέλο SUM-GAN, προκειμένου να βελτιώσουμε την μοντελοποίηση των μακροπρόθεσμων χρονικών εξαρτήσεων στο βίντεο εισόδου. Συγκεκριμένα, πειραματιστήκαμε με την αντικατάσταση των LSTM στα διάφορα σημεία της αρχιτεκτονικής SUM-GAN-AED με μετασχηματιστές. Προτείνονται οι ακόλουθες αλλαγές: αντικατάσταση του LSTM στον κωδικοποιητή με μετασχηματιστή (SUM-GAN-STD, SUM-GAN-STSED) και αντικατάσταση των LSTM στον κωδικοποιητή και του αποκωδικοποιητή επίσης με έναν μετασχηματιστή (SUM-GAN-SAT, SUM-GAN-ST). Για τα μοντέλα SUM-GAN-STD, SUM-GAN-STSED και SUM-GAN-ST χρησιμοποιούμε ένα αμφίδρομο LSTM αντί για μια μονάδα αυτοπροσοχής ως επιλογέα καρέ. Η αξιολόγηση των διαφόρων παραλλαγών LSTM, αυτοπροσοχής και μετασχηματιστή υπογραμμίζει ποια τμήματα στην αρχιτεκτονική ωφελούνται περισσότερο από τη βελτίωση της μοντελοποίησης των χρονικών εξαρτήσεων. Τα προτεινόμενα μοντέλα παρουσιάζονται αναλυτικά στη συνέχεια.

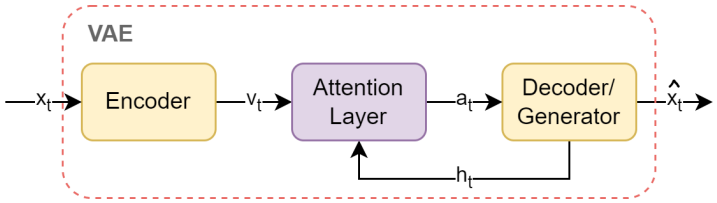

Figure 4. *Η αρχιτεκτονική VAE του μοντέλου SUM-GAN-SEAD.*

SUM-GAN-SEAD

Στο SUM-GAN-SEAD εισάγουμε έναν μηχανισμό προσοχής στη μονάδα VAE της αρχιτεκτονικής, όπως απεικονίζεται στο σχήμα 4. Έτσι, προσεγγίζεται σταδιακά ή λήψη των αποφάσεων. Το μοντέλο αυτό εμπνεύστηκε από τους Apostolidis et al. [24], οι οποίοι επεκτείνουν τον VAE στο αρχικό μοντέλο SUM-GAN-sl, προσθέτοντας προσοχή. Συγκεκριμένα, τα βάρη της προσοχής κάθε καρέ θεωρήθηκαν ως τυχαίες μεταβλητές και ένας λανθάνοντας χώρος υπολογίστηκε και για αυτά, έτσι ο αποκωδικοποιητής ενημερώνει τις κρυφές του καταστάσεις με βάση και τους δύο λανθάνοντες χώρους. Καταλήγουν συνεπώς σε ένα δίκτυο κωδικοποιητή-προσοχής-αποκωδικοποιητή για την παραγωγή της περίληψης. Ο μηχανισμός προσοχής επιτρέπει στον αποκωδικοποιητή να εστιάζει επιλεκτικά μόνο σε ένα υποσύνολο εισόδων, των οποίων αυξάνει τα βάρη προσοχής.

SUM-GAN-STD

Στο SUM-GAN-STD, ανταλλάσσουμε τον κωδικοποιητή LSTM του [2] με έναν μετασχηματιστή [8]. Ο επιλογέας καρέ είναι ένα αμφίδρομο LSTM, ενώ ο αποκωδικοποιητής και ο διαχωριστής είναι LSTM όπως στο [2]. Η χρήση μετασχηματιστή εμπνεύστηκε από μια σειρά μελετών που χρησιμοποιούν την αρχιτεκτονική του για την παραγωγή πείληψης βίντεο [27, 28] και από την ικανότητα του να αντιμετωπίζει τις αδυναμίες των ανατροφοδοτούμενων συστημάτων, καθώς βασίζεται στην αυτοπροσοχή [8]. Όπως εξηγείται από τους Narasimhan et al. [29] ο μετασχηματιστής μπορεί επίσης να συλλάβει τις μακροχρόνιες εξαρτήσεις μιας ακολουθίας, γεγονός που οδηγεί σε καλύτερα μοντελοποιημένες σχέσεις των καρέ.

Στο μοντέλο μας χρησιμοποιήσαμε την υλοποίηση PyTorch του Μετασχηματιστή [22]. Η αρχιτεκτονική SUM-GAN-STD φαίνεται στο σχήμα 5. Μετά το στρώμα γραμμικής συμπίεσης και τον επιλογέα καρέ, το σταθμισμένο διάνυσμα χαρακτηριστικών προωθείται στον μετασχηματιστή, η έξοδος του οποίου τροφοδοτείται στον αποκωδικοποιητή και στη συνέχεια στον διαχωριστή LSTM. Η έξοδος του μετασχηματιστή είναι η συνενωμένη έξοδος καθεμιάς από τις κεφαλές προσοχής πολλαπλών κεφαλών και διαμορφώνεται ως εξής:

$$\mathbf{y}_t = \text{ concat(head1, head2,..., headN) } \mathbf{W}_o \tag{7}$$

όπου:

$$head_i = Attention(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{K}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}}) \tag{8}$$

για κάθε κεφαλή προσοχής του μετασχηματιστή, όπου $\mathbf{K}, \mathbf{V}, \mathbf{Q}$ είναι οι πίνακες κλειδιού, τιμής και ερωτήματος.

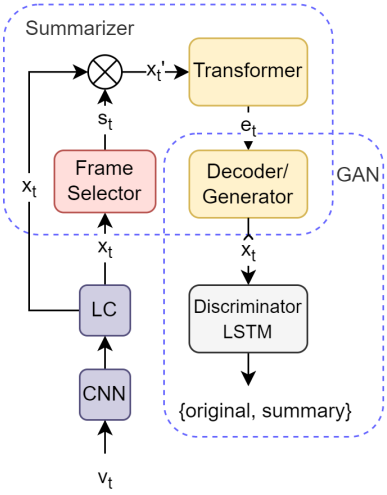

Figure 5. *Η αρχιτεκτονική SUM-GAN-STD.*

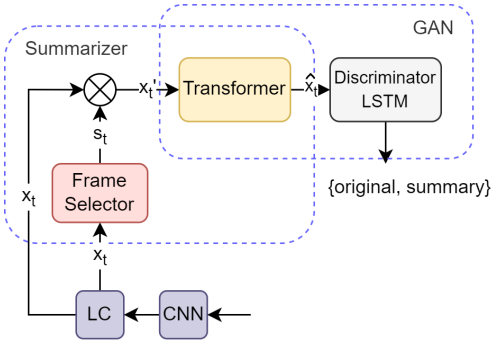

Figure 6. *Η αρχιτεκτονική SUM-GAN-ST.*

SUM-GAN-ST

Στο SUM-GAN-ST αντικαθιστούμε τον VAE, δηλαδή τον κωδικοποιητή και τον αποκωδικοποιητή, με έναν μετασχηματιστή, προκειμένου να βελτιώσουμε την ανακατασκευή βίντεο ενσωματώνοντας στα διανύσματα χαρακτηριστικών τις πληροφορίες θέσης των καρέ. Τα σταθμισμένα χαρακτηριστικά των καρέ εισέρχονται στον μετασχηματιστή και η ανακατασκευασμένη ακολουθία τους, που αντιστοιχεί στο βίντεο εισόδου, τροφοδοτείται στον διαχωριστή, προκειμένου να ταξινομηθεί ως «πρωτότυπο» ή «περίληψη». Εφόσον αφαιρούμε το VAE, κατά τη διάρκεια της εκπαίδευσης δεν χρησιμοποιούμε το Prior Loss και ο μετασχηματιστής εκπαιδεύεται ως μέρος του συνοψιστή και του GAN. Η αρχιτεκτονική του SUM-GAN-ST φαίνεται στο σχήμα 6.

SUM-GAN-STSED

Στο SUM-GAN-STSED αντικαθιστούμε τον κωδικοποιητή της αρχιτεκτονικής με έναν κωδικοποιητή ακολουθίας μετασχηματιστή, ή *Transformer Sequence Encoder* (TSE), το τμήμα της μονάδας του μετασχηματιστή που αποτελεί τον κωδικοποιητή. Ο TSE έχει ως αποτέλεσμα μια πιο αποτελεσματική αναπαράσταση των χρονικών εξαρτήσεων για τα βίντεο μεγάλης εμβέλειας και ως αποτέλεσμα του διανύσματος κρυφής κατάστασης **e**, που οδηγεί στην αποτελεσματικότερη ανακατασκευή βίντεο. Για την υλοποίησή του χρησιμοποιούμε τον

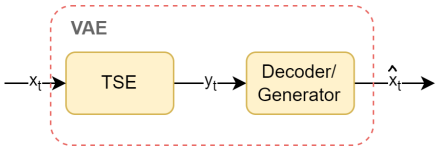

Figure 7. *Η αρχιτεκτονική SUM-GAN-STSED.*

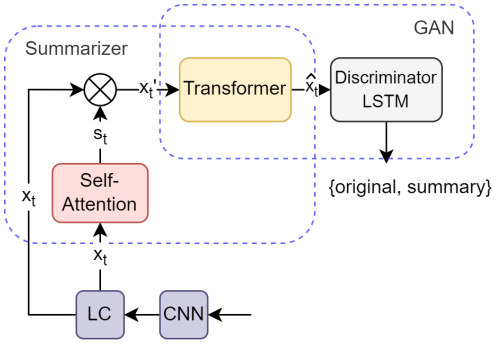

Figure 8. *Η αρχιτεκτονική SUM-GAN-SAT.*

κώδικα slp[2]. Το TSE είναι μια αρχιτεκτονική αλληλουχίας προς διάνυσμα, η οποία χρησιμοποιεί κωδικοποιήσεις θέσης για να εισάγει τις πληροφορίες σχετικής θέσης των διακριτών στοιχείων της ακολουθίας και να διατηρεί τη σειρά των καρέ. Η αρχιτεκτονική φαίνεται στο σχήμα 7. Για κάθε βίντεο, ο TSE λαμβάνει την έξοδο του επιλογέα καρέ πολλαπλασιασμένη με το διάνυσμα χαρακτηριστικών, την προωθεί σε ένα γραμμικό επίπεδο, υπολογίζει τις πληροφορίες θέσης και τις προσθέτει στον διάνυσμα. Το διάνυσμα που προκύπτει προωθείται στο τμήμα κωδικοποιητή του TSE και η έξοδος τροφοδοτείται στον αποκωδικοποιητή LSTM της αρχιτεκτονικής. Ο επιλογέας καρέ είναι ένα LSTM και η εκπαίδευση ακολουθεί εκείνη του βασικού μοντέλου SUM-GAN.

SUM-GAN-SAT

Τέλος, με κίνητρο την αποτελεσματικότητα των μονάδων που βασίζονται στην προσοχή, κατασκευάζουμε το SUM-GAN-SAT, στο οποίο αντικαθιστούμε τον επιλογέα καρέ LSTM με μια μονάδα αυτοπροσοχής και τον VAE με έναν μετασχηματιστή, όπως φαίνεται στο σχήμα. 8. Αυτό το μοντέλο συνδυάζει τις αρχιτεκτονικές SUM-GAN-AED και SUM-GAN-ST, καθώς επίσης και τα πλεονεκτήματά των μονάδων που βασίζονται στην προσοχή. Εδώ, ενσωματώνουμε την πιο ενδελεχή επιλογή καρέ που παρέχει η αυτοπροσοχή, με την καλύτερη μοντελοποίηση των πληροφοριών θέσης. Δεν χρησιμοποιούμε το Prior Loss κατά τη διάρκεια της εκπαίδευσης, η οποία ακολουθεί την αυτή των προηγούμενων μοντέλων.

Αποτελέσματα και Συζήτηση

Στον πίνακα 1 απεικονίζονται τα αποτελέσματα του F-score πρόσφατων μεθόδων πείληψης βίντεο χωρίς επίβλεψη της βιβλιογραφίας, σε σύγκριση με το μοντέλο μας SUM-GAN-AED. Όπως μπορούμε να δούμε, το προτεινόμενο μοντέλο μας ξεπέρασε τις σύγχρονες μεθόδους

[2]https://github.com/georgepar/slp

Model	SumMe	TVSum
SUM-GAN [2]	38.7	50.8
ACGAN [12]	46.0	58.5
Cycle-SUM [11]	46.8	57.6
SUM-GAN-sl [13]	46.8	**65.3**
SUM-GAN-AAE [24]	48.9	58.3
Proposed-B [30]	58.8	63.5
SUM-GAN-AED (Ours)	**64.85**	**63.18**

Table 1. *Συγκριτική αξιολόγηση της απόδοσης του μοντέλου μας SUM-GAN-AED, με σύγχρονες προσεγγίσεις περίληψης βίντεο χωρίς επίβλεψη, σε SumMe και TVSum (F-score(%)).*

στο SumMe, κατά ένα σημαντικό περιθώριο και στο TVSum η απόδοσή του είναι συγκρίσιμη με αυτήν των σύγχρονων μεθόδων.

Η αποτελεσματικότητα του SUM-GAN-AED συγκρίνεται με εκείνη των πειραματικών μοντέλων που αναπτύχθηκαν ως μέρος αυτής της μελέτης και τα οποία χρησιμοποιούν αρχιτεκτονικές προσοχής και μετασχηματιστών, στον Πίνακα 2. Παρέχουμε επίσης τα αποτελέσματα των βασικών μοντέλων [2] και [13], καθώς και την απόδοσή τους όταν τα εκπαιδεύουμε τοπικά, στα μοντέλα SUM-GAN reproduced και SUM-GAN-sl reproduced, αντίστοιχα. Και οι δύο πίνακες δείχνουν ότι η εισαγωγή του μηχανισμού προσοχής, είτε με αυτοπροσοχή είτε με μετασχηματιστές, οδηγεί σε βελτίωση της απόδοσης στο SumMe και στο TVSum, καθώς επίσης και στο COGNIMUSE, αποτέλεσμα που επιβεβαιώνει το αρχικό μας κίνητρο να την χρησιμοποιήσουμε.

Μεταξύ όλων των πειραματικών μοντέλων, το SUM-GAN-AED μας δίνει τα καλύτερα συνολικά αποτελέσματα, λαμβάνοντας υπόψη όλα τα σύνολα δεδομένων, γεγονός που επίσης δείχνει ότι η κορύφωση της απόδοσης δεν εστιάζεται σε συγκεκριμένο σύνολο δεδομένων, αλλά είναι γενική. Η προσθήκη του μηχανισμού αυτοπροσοχής στο στάδιο επιλογής των καρέ που θα εισαχθούν στην περίληψη, στην αρχιτεκτονική που βασίζεται σε μετασχηματιστή SUM-GAN-ST, οδηγεί σε βελτιωμένη απόδοση, όπως αποδεικνύεται από το SUM-GAN-SAT. Αυτό το αποτέλεσμα υπογραμμίζει τη σημασία της μονάδας αυτοπροσοχής όσον αφορά τη συνολική απόδοση, καθώς και την αξία της μοντελοποίησης της χρονικής πληροφορίας των καρέ, κατά τον υπολογισμό των σκορ τους. Θα πρέπει να σημειωθεί ότι οι πειραματικές αρχιτεκτονικές που βασίζονται σε μετασχηματιστές αποδίδουν σταθερά καλά στα σύνολα δεδομένων και ακόμη ξεπερνούν πολλές σύγχρονες προσεγγίσεις χωρίς επίβλεψη.

Γενικά, οι βαθμολογίες του TVSum είναι υψηλότερες από αυτές του SumMe για καθένα από τα μοντέλα, εκτός από το SUM-GAN-SEAD. Μια πιθανή εξήγηση είναι η διαφορά του μεγέθους των συνόλων δεδομένων. Το TVSum έχει συνολικό αριθμό καρέ ίσο με 23510 και το SumMe έχει 7336 καρέ συνολικά. Αυτό επιτρέπει στο TVSum να επιτύχει αποτελεσματικότερη εκπαίδευση, επιτυγχάνοντας έτσι καλύτερη απόδοση στη συντριπτική πλειοψηφία των προσεγγίσεων.

Μπορούμε να παρατηρήσουμε από τον Πίνακα 2, ότι η καλύτερη μέθοδος στο TVSum (SUM-GAN-sl, f-score $= 65,3\%$) είναι ιδιαίτερα προσαρμοσμένη σε αυτό το σύνολο δεδομένων, καθώς η απόδοση που παρουσιάζει στο SumMe (f-score $= 46,8\%$) κατατάσσεται

Model	SumMe	TVSum	COGNIMUSE
SUM-GAN reported	38.7	50.8	-
SUM-GAN reproduced	46.92	51.19	51.24
SUM-GAN-sl reported	46.8	**65.3**	-
SUM-GAN-sl reproduced	48.04	64.78	50.5
SUM-GAN-LC	57.99	61.74	49.99
SUM-GAN-STD	54.15	63.82	51.38
SUM-GAN-ST	56.00	60.53	50.73
SUM-GAN-STSED	61.30	62.73	52.8
SUM-GAN-SEAD	61.89	61.66	52.55
SUM-GAN-SAT	61.38	62.41	49.81
SUM-GAN-AED	**64.85**	**63.18**	**55.49**

Table 2. *Σύγκριση της απόδοσης του βασικού και των προτεινόμενων μοντέλων σε SumMe, TVSum (F-score(%)) και COGNIMUSE (AUC).*

δεύτερη από το τέλος (SUM-GAN, f-score $= 38,7\%$). Το ίδιο μοτίβο παρατηρείται για την απόδοση του SUM-GAN-sl reproduced, γεγονός που αναμέναμε.

Μέχρι στιγμής, έχουμε παρατηρήσει τα θετικά αποτελέσματα στη βελτίωση της απόδοσης, όταν χρησιμοποιούμε μηχανισμούς προσοχής για την αυτόματη παραγωγή περίληψης βίντεο. Συγκεκριμένα, τα αποτελέσματα υπογραμμίζουν για ακόμα μία φορά τη θετική συμβολή του μηχανισμού προσοχής, καθώς και του μετασχηματιστή στην ενίσχυση της ικανότητας του συνοψιστή να εντοπίζει τα πιο σημαντικά τμήματα ενός βίντεο και στην αποτελεσματική καθοδήγηση της εκπαίδευσης της αντιπαραθετικής συνιστώσας της αρχιτεκτονικής. Υποστηρίζουμε, ότι παρά τη μη βέλτιστη απόδοση στο TVSum, τα μοντέλα που βασίζονται σε μηχανισμούς προσοχής και μετασχηματιστές, ξεπερνούν αυτά που βασίζονται μόνο σε LSTM, καθώς η βέλτιστη απόδοση δεν εστιάζεται σε συγκεκριμένο σύνολο δεδομένων, αλλά τα μοντέλα πετυχαίνουν υψηλές ακρίβειες σε όλα τα σύνολα δεδομένων.

Συμπεράσματα

Στην παρούσα εργασία προτείνουμε ένα νέο πλαίσιο για την περίληψη βίντεο χωρίς επίβλεψη, που βασίζεται σε ένα Παραγωγικό Ανταγωνιστικό Δίκτυο. Βασιζόμενοι στο μοντέλο SUM-GAN, χρησιμοποιούμε τον μηχανισμό προσοχής, εισάγοντας μονάδες αυτο-προσοχής και μετασχηματιστών στην αρχιτεκτονική μας, προκειμένου να αξιοποιήσουμε τις εξαρτήσεις μεγάλης εμβέλειας, να προσαρμόσουμε τα μοντέλα σε διαφορετικά μήκη ακολουθίας που δεν συναντώνται στην εκπαίδευση και να ενσωματώσουμε τις πληροφορίες θέσης των καρέ στα διανύσματα χαρακτηριστικών.

Τα πειράματά μας δείχνουν ότι η χρήση της αυτοπροσοχής για την επιλογή καρέ, ακολουθούμενη από LSTM για την κωδικοποίηση και αποκωδικοποίηση, οδηγεί στα καλύτερα συνολικά αποτελέσματα για την περίληψη βίντεο. Επιπλέον, οι πειραματικές αρχιτεκτονικές που βασίζονται σε μετασχηματιστές αποδίδουν σταθερά καλά στα σύνολα δεδομένων και ξεπερνούν, ακόμη, πολλές υπερσύγχρονες προσεγγίσεις χωρίς επίβλεψη. Η ποσοτική αξιολόγηση σε δύο ευρέως διαδεδομένα σύνολα δεδομένων αναφοράς (TVSum, SumMe) και η περαιτέρω

αξιολόγηση στο συνολο δεδομένων COGNIMUSE, επιβεβαιώνει την αποτελεσματικότητα της προσέγγισής μας.

Chapter 1

Introduction

1.1 Motivation

The amount of digital information that users receive and consume everyday, comes in different modalities and overwhelming quantity. It is therefore crucial that systems are developed in such direction, so as to aid the users in their interaction with such a dense amount of information as efficiently as possible. Video content is classified as the number one source of information for the majority of the Internet users. From educational content to social media posts and adds, video is in high demand from the consumers. Our brains are hard-wired to enjoy storytelling as a narrative, compared to other forms of media and video presents a great way to tell a story and engage, as well as connect emotionally.

Video summarization is the task of generating a short, concise synopsis of a video, that conveys the important parts of the original, full-length video, encapsulates the flow of the story and enables the user to directly access the most important segments of the video. The objective is for the produced summary to retain only the significant parts and contain as little unnecessary content as possible [14]. Video summarization aims to fulfill the ever-increasing aforementioned users' needs and its applications, apart from efficient browsing and retrieval of videos, include the summarization of surveillance videos, medical videos, videos captured by Unmanned Aerial Vehicles (UAV) [15], the effective indexing and promotion of organizations' media assets, etc.

In the task of video summarization, many deep learning models have been developed, as the relevant research area remains very active and new approaches are presented in a yearly basis. The deep learning-based video summarization methods significantly outperform more traditional approaches, such as techniques that rely on weighted fusion, sparse subset selection or data clustering algorithms and represent the current state-of-the-art. The video summary is the result of the composition of a set of representative key-frames, in which case it is called a story-board or static video summary, or of a set of key-fragments or key-shots, which is called a video skim. In both cases, the representative key-frames or key-fragments are arranged together in chronological order. Furthermore, depending on the type of data that a method uses to extract the video summary, there are unimodal and multimodal approaches. Unimodal approaches utilize only the visual information of the video to extract the frame features, whereas multimodal approaches exploit the available audio or/and textual metadata as well.

In this work, motivated by the recent increase of the need to efficiently handle video media, content and information, we are eager to explore the challenges and opportunities presented when building artificial intelligence models for automatic video summarization. To this end, not only do we study video summarization as a deep learning task, but we also build an end-to-end unimodal system based on a generative adversarial network and attention mechanisms, to summarize videos, that tackles the task as a key-segment selection problem.

1.2 Objectives and Contributions

The main objective of this work is to develop an end-to-end video summarization model, which given a video as an input, will generate the corresponding summary as a set of video key-segments. Based on this, we build on a simple generative adversarial network, SUM-GAN [2], which constitutes the foundation of our models, expanding it with the integration of attention mechanisms, such as self-attention or transformers, in different parts of the architecture.

In particular, our main contributions are as follows:

- We investigate the efficacy of the integration of attention mechanisms and transformers in different parts of the SUM-GAN architecture, which is based on a generative adversarial network. Namely we we experiment with using a transformer as the frame selector (SUM-GAN-SAT), the encoder (SUM-GAN-STD, SUM-GAN-STSED) and the encoder-decoder (SUM-GAN-SAT, SUM-GAN-ST).
- Based on the above results, we propose the use of a self-attention mechanism for the frame selection of the model, while retaining the Long Short-Term Memory (LSTM) architecture for the encoder and the decoder (SUM-GAN-AED).
- We evaluate our models on two benchmark datasets SumMe [16] and TVSum [17] and we achieve state-of-the-art performance on SumMe and competitive performance on TVSum.
- We extend our evaluation to one more dataset, which we build from the database COGNIMUSE [18] and on which or model also achieves competitive performance.

1.3 Thesis Outline

In Chapter 2, Machine Learning, we provide the theoretical foundations, in order for the reader to familiarize themselves with the foundations of this work. This theoretical background is considered necessary to understand the rest of this dissertation. We first present an overview of the basics of Machine Learning, detailing fundamental Machine Learning Approaches and Standard Leaning Tasks, as well as some principal concepts. Next, we document the fundamentals of Deep Learning, analyzing Recurrent Neural Networks, Sequence-to-sequence Models and Attention Mechanisms, all of which are important, as they are the basis of our work.

In Chapter 3 we present the necessary Video Summarization theory background, that is needed to understand this work. After presenting the main Video Summarization Types, we analyze the most popular Video Summarization Approaches, such as Feature-based Video Summarization, which is the basis of our method.

In Chapter 4 we present our approach and the method we followed to build our Video Summarization model. We first briefly discuss the recent bibliography on the subject and we analyze the baseline model, which acted as the foundation of our proposed models. Then we present in detail the architectures we propose and on which we conduct our experiments, as well as the final proposed model. We discuss the training process and the hyperparameters we chose and we compare the results of all the models. We also describe the datasets that were used and the steps that were followed in order to build the dataset COGNIMUSE, as well as the evaluation metrics that were applied. Finally, we analyze the qualitative evaluation and discuss the outcomes.

Finally, in Chapter 5, we discuss our research efforts and draw conclusions, present the advantages and limitations, as well as the future work that could enhance the efficacy of our approach.

Chapter 2

Machine Learning

In this chapter we will lay the foundations upon which we have built our work. We dive into the fundamentals of Machine Learning and we present the main theoretical background regarding Machine Learning Approaches, Learning Tasks and core concepts, emphasizing subsequently, on the analysis of Deep Learning.

2.1 Introduction to Machine Learning

In this era of information explosion that we are currently experiencing, Machine Learning is the tool that allows us to handle enormous amounts of data, using significantly improved algorithms, and substantially more powerful computer hardware. Machine Learning deals with the design of systems that can learn rules from data, adapt to changes, and improve performance with experience. In addition to being one of the initial dreams of Computer Science, Machine Learning has become crucial as computers are expected to solve increasingly more complex problems and become even more integrated into our daily lives. In 1959, Arthur Samuel, a computer scientist, pioneered the study of artificial intelligence and described machine learning as "the study that gives computers the ability to learn without being explicitly programmed." A more technical definition given by Tom M. Mitchell in 1997: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [31].

These tasks that can be accomplished with Machine learning algorithms, include a wide variety of applications, that from an engineering point of view, are too difficult to solve with fixed programs written and designed by human beings. Some of the applications of Machine Learning include computer vision, speech recognition and traffic prediction. In these areas of study, we are not capable of developing traditional algorithms and computational methods to find solutions for the needed tasks. Thus, Machine Learning allows us to find solutions for such issues, otherwise infeasible to deal with. A core objective of a Machine Leaning algorithm is to generalize from its experience [32]. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that

enables it to produce sufficiently accurate predictions in new cases. In the next sections we will briefly analyze the core Machine Learning approaches.

2.2 Machine Learning Approaches

There are generally four basic Machine Learning approaches. These are supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. Supervised learning can be thought of as a family of algorithms that learn a function from input samples to target values, given a set of data, for which the target responses are known known as "labels". Unsupervised learning models learn the structure of the input data, without any relevant information provided provided in advance, there aren't any labels. In semi-supervised learning, models do not have labels for every sample, and thus, it is a combination of supervised and unsupervised learning. As stated by Zhu et al. [33], the goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. Finally, reinforcement learning deals with learning in sequential decision making problems in which there is limited feedback. In this setting, the model is learning what to do - how to map situations to actions - so as to maximize a numerical reward [34], defined by a reward function.

2.2.1 Supervised Learning

Supervised learning entails learning a mapping between a set of input variables $\mathcal{X}$ and an output variable $\mathcal{Y}$ and applying this mapping to predict the outputs for unseen data [35]. Supervised learning is the most important methodology in machine learning and it also has a central importance in the processing of multimedia data. During training, both $\mathcal{X}$ and their corresponding labels $\mathcal{Y}$ are provided. At inference, we expect that the mapping function successfully predicts the output for every given $\mathcal{X}$ provided from the same distribution as the training sample, without the label $\mathcal{Y}$ given. The main supervised learning tasks are regression and classification, which are further analyzed in subsequent sections.

2.2.2 Unsupervised Learning

Unsupervised learning is a family of algorithms that learn to infer patterns, without given target values for each learning example. The machine simply receives inputs $x1, x2, ...$, but obtains neither supervised target outputs, nor rewards from its environment. It may seem somewhat mysterious to imagine what the machine could possibly learn given that it doesn't get any feedback from its environment. However, it is possible to develop a formal framework for unsupervised learning based on the notion that the machine's goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc. [36]. In a sense, unsupervised learning can be thought of, as finding patterns in the data above and beyond what would be considered pure unstructured noise. Two very simple

classic examples of unsupervised learning are clustering and dimensionality reduction. We discuss these in section 2.3.

2.2.3 Semi-supervised Learning

Semi-supervised learning is an approach to Machine Learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning and supervised learning. It is a special instance of weak supervision. In this kind of setting there are the input variables $\mathcal{X}$ and an output variable $\mathcal{Y}$ for only a subset of the samples. Semi-supervised learning has tremendous practical value, as in many tasks, there is a paucity of labeled data. The labels $\mathcal{Y}$ may be difficult to obtain because they require human annotators, special devices, or expensive and slow experiments.It can potentially utilize both labeled and unlabeled data to achieve better performance than supervised learning. [33]

2.2.4 Reinforcement Learning

The technique of reinforcement learning (Sutton and Barto, 1998 [37]) is concerned with the problem of finding suitable actions to take in a given situation in order to maximize a reward. In reinforcement learning, there is a learning agent interacting with its environment to achieve a goal. The agent performs actions based on observations, and receives a reward from the environment. The behavior of the agent depends on a function that maps the observations of the environment to actions. The machine uses trial-and-error, in order to learn. It starts with random trials and having as aim to optimize a reward, it progresses to advanced techniques and abilities. One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between exploration and exploitation. To obtain a high reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that have not been selected before. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favor those that appear to be best. The exploration–exploitation dilemma has been intensively studied by mathematicians for many decades.[32]

2.3 Standard Learning Tasks

In the following sections we are going to briefly analyze the most common machine learning tasks. Starting with classification, we are going to present the fundamentals of the approach, moving on with regression, linear and logistic and clustering.

2.3.1 Classification

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given number of input data. Having the input samples, the goal is to train a model to classify them to a class label, according to some features. This model is called a classifier. In this type of task, the model is asked to specify which of k categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbf{R}_n \rightarrow \{1, ..., k\}$. When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y.

In the case of $k = 2$ there are only two classes and we refer to it as a binary classification task, otherwise it is known as multi-class classification. A further categorization that can be made is between generative and discriminative classification models. In generative models, the classifier tries to model the distribution of the data i.e., what are the features of the class. In short, it models how a particular class would generate input data. Whereas discriminative classifiers learn what the features in the input are most useful to distinguish between the various possible classes. They focus on optimizing an objective function to best discern between the classes, thus implicitly try to learn the decision boundary for the model.

Bayes Classifier A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task and it is based on the Bayes theorem. According to Bayes Theorem, the posterior probability of a sample to belong to a specific class, equals to the product of the likelihood of this sample's generation in this class and the class prior probability, divided by the predictor prior probability:

$$P(y_j|x) = \frac{P(x|y_j)P(y_j)}{P(x)}, \text{ for a class } c \tag{2.1}$$

We can easily understand that prior probability $P(x)$ is equal for every class, and it acts as a scaling factor, ensuing that the probability $P(y_j|x)$ is properly scaled. When we are interested in a crisp classification rule, that is, a rule that assigns each instance to exactly one class, then we can simply calculate the value of the numerator for each class and select that class for which this value is maximal. This rule is called the maximum posterior rule. The resulting "winning" class is also known as the maximum a posteriori (MAP) class, and it is calculated as $\hat{y}$ for the instance x as follows:

$$\hat{y} = \underbrace{\text{argmax}}_{y_j} \prod_{k=1}^{p} P(x_k|y_j)P(y_j) \tag{2.2}$$

A model that implements Eq. 2.2 is called a (simple) naive Bayes classifier. One of the main advantages of Naive Bayes Classifier is that it does not require a lot of training data to estimate the parameters [38].

Perceptron The perceptron (or McCulloch-Pitts neuron) is an algorithm for supervised learning of binary classifiers. It was introduced in 1957 by F. Rosenblatt[39]. It is the

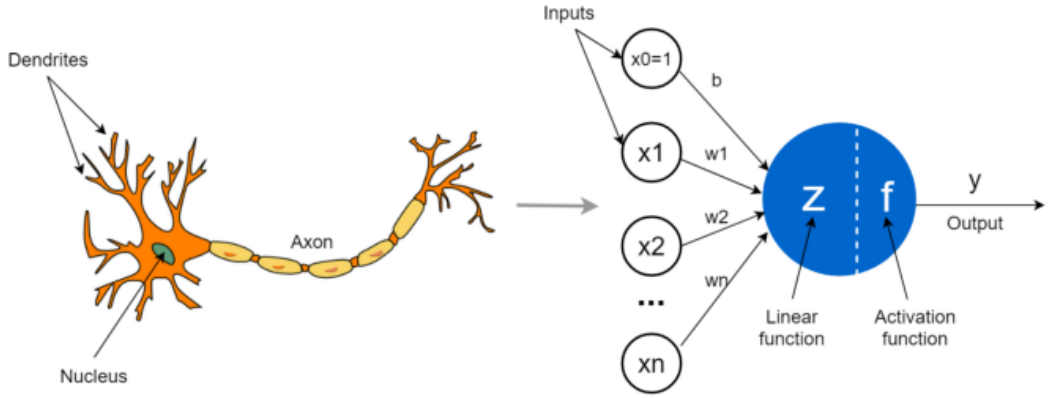

Figure 2.1. *Perceptron Structure. Source[3]*

simplest type of neural network and is also called a single layer neural network. It is a type of linear classifier, with a learnable weight vector. It can be denoted as a linear predictor function f that maps its input $\mathbf{x}$ vector to a binary output value $f(\mathbf{x})$:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{wx} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

where $\mathbf{w}$ is a vector of weights, $\mathbf{wx}$ is the dot product equal to $\sum_{i=1}^{m}$, and b is the bias. These parameters are updated through an iterative algorithm, with the goal of mapping the most inputs to the correct output. The perceptron is the building block of an Artificial Neural Network and is a biologically inspired computational model, patterned after the network of neurons present in the human brain. Figure 2.5 shows the comparison between a biological neuron and the perceptron. We model the neuron's firing rate with an activation function f, which represents the frequency of the spikes along the axon.A standard activation function is the sigmoid function σ, since it takes a real-valued input and squashes it to a range between 0 and 1.

In order to learn complex non-linear functions, architectures that combine several artificial neurons can be designed and implemented. Such architectures are called Multi-Layer Perceptrons (MLPs). An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. The advantage of the MLP is that it can distinguish data that is not linearly separable.

Support Vector Machines A support vector machine (SVM) is a computer algorithm that learns by example to assign labels to objects. In essence, an SVM is a mathematical entity, an algorithm (or recipe) for maximizing a particular mathematical function with respect to a given collection of data. The basic ideas behind the SVM algorithm, however, can be explained without ever reading an equation. Indeed, I claim that, to understand

the essence of SVM classification, one needs only to grasp four basic concepts: (i) the separating hyperplane, (ii) the maximum-margin hyperplane, (iii) the soft margin and (iv) the kernel function. [40] One of the advantages of the SVMs is that they can efficiently classify non-linearly separable data, using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. More specifically the SVM constructs hyperplanes, to perform classification in the higher-dimensional space.

Suppose we have a two-class classification problem using linear models of the form:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{2.4}$$

where $\phi(\mathbf{x})$ denotes a fixed feature-space transformation, and we have made the bias parameter b explicit. The training dataset comprises N input vectors $x_1, x_2, ..., x_N$ with corresponding target values $y_1, y_2, ..., y_N$ where $y_i\{1, 1\}$, and new data points x are classified according to the sign of $f(x)$. We shall assume for the moment that the training data set is linearly separable in feature space, so that by definition there exists at least one choice of the parameters w and b such that a function of the above form satisfies $f(x) > 0$ for points having $y_i = +1$ and $f(x) < 0$ for points having $y_i = 1$, so that $y_i f(x) > 0$ for all training points.

2.3.2 Regression

Regression is used to identify the relationship between a dependent variable (output label) and one or more independent variables (input features) and is typically leveraged to make predictions about future outcomes. The label here is a value in a continuous space, which the model is called to predict.

Linear Regression As the name suggests, linear regression models have the goal of finding the optimal line that best describes the data points, a process described as "fitting“ the data. The key property of linear regression is that it is a linear function of its parameters w. In its simplest form, it is also a linear function with respect to the input variables x, but a much more useful class of functions can be obtained through applying a fixed set of non-linear functions to the input, known as basis functions. It can be defined as:

$$y(x) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^\tau \phi(x) \tag{2.5}$$

where ϕ_j are the fixed nonlinear basis functions ($\phi_0 = 1$) and M is the total number of model parameters.

The error of estimating the curve for N data points can be described by:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} [t_n - w^T \phi(x_n)]^2 \tag{2.6}$$

also known as the sum of squares error (loss) function, with respect to the ground truth values t_n. Another metric called mean squared error (MSE) refers to the unbiased estimate

of error variance, which is the sum of squares divided by the number of degrees of freedom (usually N)[41]. The optimal parameters w are those that minimize equation ,2.6 which is called the least squares solution.

Logistic Regression Logistic regression (LR) is a linear classification model, which computes the probabilities for classification problems with two possible outcomes, by applying the logistic function to the output of a linear function f. The logistic function, also known as the sigmoid function, squeezes a vector into a range of (0, 1). For a binary classification problem, the probability of one of the classes for a feature vector $x \in \mathcal{R}^d$ is defined as:

$$P(y = 1|x) = \frac{1}{1 + e^{-f(x)}} \tag{2.7}$$

where f is a linear function with w_d parameters:

$$f(x) = w_0 + w_1 x_1 + ... + w_d x_d \tag{2.8}$$

Since we examine problems with only two classes, the probability of the other class can be computed as:

$$p(y = 0|x) = 1 - P(y = 1|x) \tag{2.9}$$

The parameters of the linear function are computed by minimizing the cross-entropy loss J, which is defined as:

$$J(w) = [y \log(P(y = 0|x)) + (1 - y) \log(P(y = 1|x))] \tag{2.10}$$

2.3.3 Clustering

Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A cluster is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. Besides the term data clustering as synonyms like cluster analysis, automatic classification, numerical taxonomy, botrology and typological analysis. [42]

Data clustering algorithms can be hierarchical or partitional. Hierarchical algorithms find successive clusters using previously established clusters, whereas partitional algorithms determine all clusters at time. Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. A key step in a hierarchical clustering is to select a distance measure. A simple measure is Manhattan distance, equal to the sum of absolute distances for each variable. Other measures include the Euclidean distance function. Partitioning algorithms are based on specifying an initial number of groups, and iteratively reallocating objects among groups to convergence. This

algorithm typically determines all clusters at once. Most applications adopt one of two popular heuristic methods like k-means algorithm or k-medoids algorithm.

2.3.4 Further Tasks

Some other well known machine learning tasks are transcription, machine translation, anomaly detection and synthesis & sampling and we are going to briefly describe each one.

In transcription, the machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form. For example, in optical character recognition, the computer program is shown a photograph containing an image of text and is asked to return this text in the form of a sequence of characters (e.g., in ASCII or Unicode format). Another example is speech recognition, where the model is provided an audio waveform and emits a sequence of characters or word ID codes describing the words that were spoken in the audio recording. Deep learning is a crucial component of modern speech recognition systems used at major companies, including Microsoft, IBM and Google, Hintonet al.[43].

In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language. This is commonly applied to natural languages, such as translating from English to French.[32]

In anomaly detection the computer program sifts through a set of events or objects and flags some of them as being unusual or atypical, base on Hinton et al.[44]. An example of an anomaly detection task is credit card fraud detection.

In synthesis & sampling, the machine learning algorithm is asked to generate new examples that are similar to those in the training data.[45] Synthesis and sampling via machine learning can be useful for media applications when generating large volumes of content by hand would be expensive, boring, or require too much time. In some cases, we want the sampling or synthesis procedure to generate a specific kind of output given the input. For example, in a speech synthesis task, we provide a written sentence and ask the program to emit an audio waveform containing a spoken version of that sentence. This is a kind of structured output task, but with the added qualification that there is no single correct output for each input, and we explicitly desire a large amount of variation in the output, in order for the output to seem more natural and realistic.[32]

2.4 Concepts

In the following sections we are going to describe some key Machine Learning concepts, that are fundamental in the methods we used to approach Video Summarization.

2.4.1 Feed Forward Neural Networks

Artificial neural networks are inspired by the early models of sensory processing by the brain, as stated by Krogh et al [4]. An artificial neural network can be created by simulating a network of model neurons in a computer. By applying algorithms that mimic

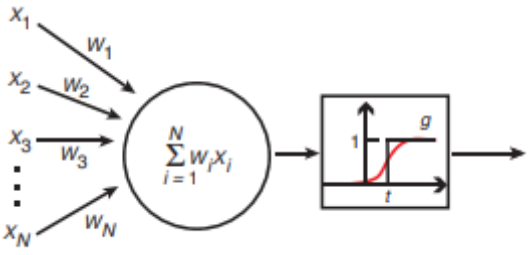

Figure 2.2. *Graphical representation of the McCulloch-Pitts model neuron or threshold unit. Source:[4]*

the processes of real neurons, we can make the network 'learn' to solve many types of problems. A model neuron is referred to as a threshold unit and its function is illustrated in Figure 2.2. It receives input from a number of other units or external sources, weighs each input and adds them up. If the total input is above a threshold, the output of the unit is one; otherwise it is zero. Therefore, the output changes from 0 to 1 when the total weighted sum of inputs is equal to the threshold. The points in input space satisfying this condition define a so called, hyperplane. In two dimensions, a hyperplane is a line, whereas in three dimensions, it is a normal plane. Points on one side of the hyperplane are classified as 0 and those on the other side as 1. It means that a classification problem can be solved by a threshold unit if the two classes can be separated by a hyperplane. Such problems, are said to be linearly separable and the perceptron, that we introduced in Section 2.3.1 is used to model them.

Feed Forward Neural Networks (FFNs) was the first and simplest type of Artificial Neural Network devised. It can be thought therefore as the basis of Deep Learning. Another name for FFNNs is Multilayer Perceptrons (MLPs) and it reveals that they consist of multiple stacked layers of Perceptron units. These Perceptron units are connected without any feedback loops, in a Directed Acyclic Graph. The advantage of using perceptrons, lies within the fact that if we introduce non-linearities through non-linear activation functions, which are described in the next section, we are able to distinguish between non-linearly separable data. Multi-layer networks use a variety of learning techniques, the most popular being back-propagation. A feed forward network consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. The input layer takes as input the data, the hidden layers process the outputs of the previous layers and the output layer provides the final output. The flow of information from the input to the output is called forward propagation. Typically the units of a single layer are not connected to each other and the number of layers of the model determines the depth of the model. An example of a FFN can be seen in figure 2.3.

2.4.2 Activation Function

The activation function refers to the equation that determines the output of each neuron given its input or, in other words, it defines how the weighted sum of the input is transformed into an output. It serves as a mathematical gate between the neuron's input

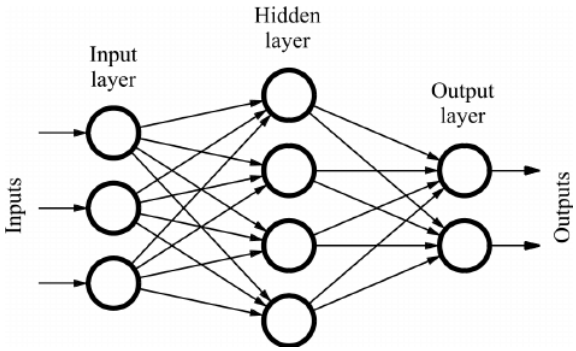

Figure 2.3. *Feed Forward Neural Network. Source[5]*

or set of inputs, and the output that will be transmitted to the next layer. In its simplest form, it can be a binary function that turns the neuron on and off, depending on the input. However, activation functions are essential for non-linearities, as only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called non-linearities. It can also help normalize the output to a range between -1 and 1, or transform the input signals into output signals. Some of the most popular functions are presented next.

Linear Activation Function A linear function is defined as a straight line, where the activation is proportional to the input i.e. the weighted sum from neurons. Hence, it is mathematically defined as:

$$f(x) = ax + b \tag{2.11}$$

The linear activation function has a lot of drawbacks and is not usually used in modern artificial neural networks. The final layer of the Neural Network will be acting as a linear function of the first layer.

Sigmoid Function The sigmoid or logistic function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.12}$$

The sigmoid function is differentiable, defined for all real input values and has a nonnegative derivative at each point. It takes a real-valued number and outputs a real number bounded in the range [0, 1]. In particular, large negative numbers become 0 and large positive numbers become 1.

Tanh or Hyperbolic Tangent Tanh is defined as follows:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.13}$$

As an activation function, tanh is mostly used to model inputs that have strongly negative and positive values as it is zero-centered. Its outputs range between -1 and 1 and is basically a rescaled sigmoid function. The tanh can be seen in figure 2.5.

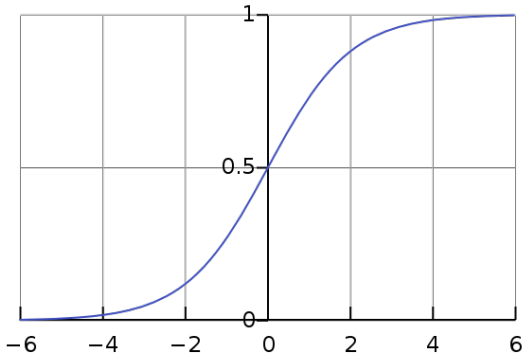

Figure 2.4. *The logistic sigmoid function.*

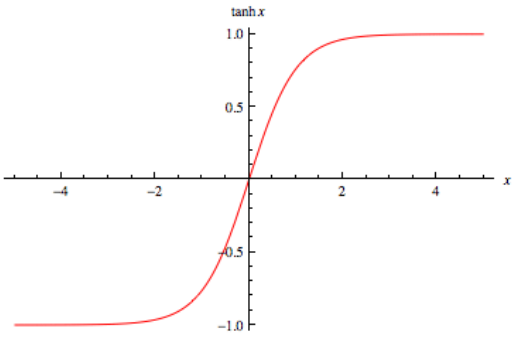

Figure 2.5. *The tanh function.*

Rectified Linear Unit (ReLU) ReLU is a non-linear function and is the most commonly used activation function in neural networks. It is a simple calculation that returns the value of the input, or 0, if the input value is less than 0. Thus, it can be defined as:

$$f(x) = max(x, 0) \tag{2.14}$$

This activation function simply thresholds the input at zero, keeping only positive inputs. It is otherwise known as the ramp function. It is one of the most commonly used activation functions. It has many advantages, some of them are the fact that it doesn't include any expensive operations, meaning exponential operations, it accelerates convergence in a lot of cases, and it avoids the vanishing gradients problem. ReLU activation function has also a few variants that alleviate some of its problems:

- **Leaky ReLU**: a linear variant of the ReLU that attempts to fix the problem of "dying ReLU". It is defined as:

$$f(z) = \begin{cases} z \text{ if } z > 0, \\ az \text{ otherwise.} \end{cases} \tag{2.15}$$

  where α is a small constant (commonly 0.01).

- **ELU**: Exponential linear units try to make the mean activations closer to zero, which speeds up learning. It has been shown that ELUs can obtain higher classification

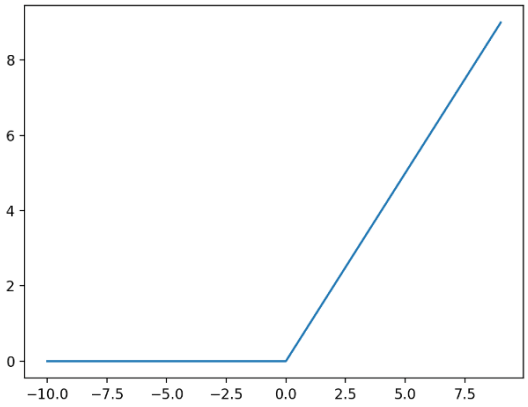

Figure 2.6. *The ReLU function.*

accuracy than ReLUs. This activation function is defined as:

$$f(x) = \begin{cases} x \text{ if } x > 0, \\ a(e^x - 1) \text{ otherwise.} \end{cases} \tag{2.16}$$

2.4.3 Loss Function

The goal of any Supervised Learning algorithm is to return a function $f()$ which accurately matches the input examples to the corresponding labels. The Loss Function is a function that measures the difference, or loss, between a predicted label and a true label, as stated by Mohri et al. [46]. Denoting the set of all labels as $\mathcal{Y}$ and the set of possible predictions as $\mathcal{Y}'$, a loss function $\mathcal{L}$ is a mapping $\mathcal{L} : \mathcal{Y} \times \mathcal{Y}' \to \mathbb{R}_+$. In most cases, $\mathcal{Y}' = \mathcal{Y}$ and the loss function is bounded, but these conditions do not always hold.

Given a train set $(x_{1:n}, y_{1:n})$, a cost function L per sample and a function $f(x; \theta)$, the total loss is defined as the average loss on all training data:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x; \theta), y_i)$$

Where the goal is to find the optimal parameters θ that minimize the total error:

$$\hat{\theta} = arg_\theta \mathcal{L}(\theta) = arg_\theta min \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x; \theta), y_i)$$

A different loss function should be selected, depending on the task or the dataset, since classification models (binary or multi-label) and regression models use a different method to calculate the loss. A few standard cost functions are presented below:

Mean Square Error (MSE): MSE calculates the mean squared prediction error:

$$\mathcal{J}(\partial) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - P_i)^2$$

Where the prediction error is the difference between the ground-truth value Y_i and the

predicted value (P_i) for an instance, and ∂ is the parameter vector of the network. MSE is primarily used with regression models.

Mean Absolute Error (MAE): MAE calculates the mean of the absolute prediction error:

$$\mathcal{J}(\partial) = \frac{1}{n}\sum_{i=1}^{n}|Y_i - P_i|$$

Where Y_i is the true value and P_i is the predicted value for an instance, and ∂ is the parameter vector of the network.

Cross Entropy: Cross-entropy loss function uses the concept of cross-entropy. Cross-entropy is mathematically defined as:

$$H(p,q) = -\sum_{k} p_k log(q_k)$$

Where p and q are the true and the predicted probability distributions, respectively, the more the two distributions differ, the higher the value of the cross-entropy. The cross-entropy loss function is widely used in classification problems. Based on the definition of cross-entropy, the goal of the Cross-entropy loss function is to minimize the cross-entropy between the model's distribution and the distribution of the given data.

2.4.4 Generalization, Overfitting and Underfitting

The ability to categorize correctly new examples that differ from those used for training is known as *generalization* [32]. In practical applications, the variability of the input vectors will be such that the training data can comprise only a tiny fraction of all possible input vectors, and so generalization is a central goal in pattern recognition. In any case, generalization evaluates a model's ability to process new data and generate accurate predictions after being trained on a fixed training set. It refers to the model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.

In general, the best predictor on a training sample may not be the best overall. A predictor chosen from a very complex family of functions can essentially memorize the data, but generalization is distinct from the memorization of the training labels [46]. The trade-off between the sample size and complexity plays a critical role in generalization. When the sample size is relatively small, choosing from a too complex a family of functions may lead to poor generalization, which is also known as overfitting. It occurs when the gap between the training error and test error is too large. On the other hand, with a too simple family of functions, it may not be possible to achieve a sufficient accuracy, which is known as underfitting. It occurs when the model cannot obtain a sufficiently low error value on the training set.

2.4.5 Reguralization, Dropout

Overcoming the problem of overfitting is crucial to improve generalization, while training a model. One technique that is often used to control the over-fitting phenomenon in such cases is that of regularization, which involves adding a penalty term to the loss function in order to discourage the coefficients from reaching large values [31]. Regularization is one of the central concerns of the field of machine learning, rivalled in its importance only by optimization. The no free lunch theorem has made it clear that there is no best machine learning algorithm, and, in particular, no best form of regularization[47]. Instead we must choose a form of regularization that is well-suited to the particular task we want to solve. Regularization is achieved by implying a term in the loss function, that penalized the size of the model. The form of the Loss Function is then:

$$\hat{\theta} = \underbrace{argmin}_{\theta} \mathcal{L}(\theta) = \underbrace{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) + \lambda R(\theta)$$

The regularization term considers the parameter values and scores their complexity. We then look for values that have both a low loss and low complexity. What regularization inherently intends to do is penalize complex models and favor simpler ones. λ is a value that must be set manually, based on the classification performance on a development set (called hyperparameter). The Regularizers R measure the norms of the parameter matrices and opt for solutions with low norms. The two most common regularization norms are L_1 and L_2, and the most common technique is dropout.

L_1 **Regularization:** The L1 regularizer, or lasso, encourages sparse solutions or models with many zero value parameters. It punishes uniformly low and high values and intends to decrease all non-zero parameter values towards zero.

$$R_L(\mathbf{W}) = \| \mathbf{W} \|_1 = \sum_{i,j} |W_{[i,j]}| \tag{2.17}$$

L_2 **regularization:** R takes the form of the standard Euclidean norm (L_2 -norm) of the parameters, trying to keep the sum of the squares of the parameter values low. Large model weights $\mathbf{W}_{[i,j]}$ will be penalized, since they are considered "unlikely". L_2 is also referred to as weight decay. High weights are severely penalized, but weights with small values are only negligibly affected.

Dropout: An effective regularization technique, that offers a very computationally cheap and remarkably effective regularization method to reduce overfitting and improve the generalization error in deep neural networks of all kinds. During the training of the model, some number of layer outputs are randomly ignored or "dropped out". This has the effect of making the layer look like and be treated like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs. This conceptualization suggests that perhaps

dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust.

2.4.6 Gradient Descent

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update [48].

Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta) \tag{2.18}$$

where η is the learning rate, a hyperparameter that controls the extent to which the model parameters are adjusted concerning the loss gradient and $J()$ is the loss function. We update our parameters in the direction of the gradients with the learning rate determining how big of an update we perform. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces. However, as we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that do not fit in memory. Batch gradient descent also does not allow us to update our model online, i.e. with new examples on-the-fly.

Stochastic Gradient Descent in contrast performs a parameter update for each training example x_i and label y_i:

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta; x_i; y_i) \tag{2.19}$$

Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. Stochastic Gradient Descent does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online. At the same time, it does not always guarantee convergence. Opting for a small learning rate may lead to slow convergence, while selecting a very large one might cause the model to fail to converge.

2.4.7 Backpropagation

In order to use Gradient Descent, it is important to compute the gradient of the loss function with respect to the weights of the network. Every neural network can be illustrated as a directed graph where each neuron corresponds to a node and each weight to an edge. The backpropagation algorithm [49, 50] computes the gradient of the loss function for each input-output pair, with respect to each weight. It is the backbone of training neural networks. Computing the gradient for a network is not a trivial step, especially for large and complex networks. The backpropagation algorithm can efficiently compute the gradients for all the parts of the network. It works by computing the gradient of the function with

respect to each weight, using the chain rule. It computes the gradient one layer at a time, iterating backward from the last layer and caching intermediate results to avoid redundant calculations.

2.5 Deep Learning

Deep learning (DL) is a part of Machine Learning and is a set of learning methods that imitate the workings of human brain in processing data. The fundamental blocks of deep learning networks are artificial neural networks stacked together in a number levels and form different architectures. Algorithms in deep learning extract high-level features from raw data, by propagating the input through the consecutive levels of the network. Every level serves as a function that learns extract a representation for the input data. Deep learning has been applied to numerous fields of study, including computer vision, speech recognition, natural language processing, bioinformatics and medical image analysis. The capability of Deep Learning algorithms to find solutions to supervised and unsupervised is an important benefit because unlabeled data are more abundant than the labeled data.

2.5.1 Feedforward Neural Networks

A Feed Forward Neural Network is an artificial neural network in which the connections between the nodes do not form a cycle. In these architectures, also often called Multilayer Perceptrons (MLPs), the information is only processed in one direction from the input nodes, through the hidden nodes to the output nodes. In each node of an FFNN the weighted sum of its inputs is usually computed, followed by a non-linear activation function. In its simplest form, the network consists of three layers: the input layer, the hidden layer, and the output layer. Except for the input nodes, each node is a neuron that uses a non-linear activation function. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. In order to learn complex non-linear functions, architectures that combine several artificial neurons can be designed and implemented. Such architectures are called Multilayer Perceptrons and are constructed by stacking multiple FFNNs together. That way it is possible to distinguish data that is not linearly separable. In Figure 2.7 we can see a deep neural network (such as an MLP or an FFNN). A deep neural network consists of more than one hidden layer.

Each neural network is composed of the following layers:

1. **Input Layer:** Accepts the input data and does not provide any further computations. The nodes of this layer forward the information to the hidden layer.

2. **Hidden Layer:** At least one hidden layer processes the inputs obtained by the previous layer, extracting the required features from the input data. Further hidden layers extract higher-level features.

3. **Output Layer:** It is the last layer that receives the processed data and generates the output of the model.

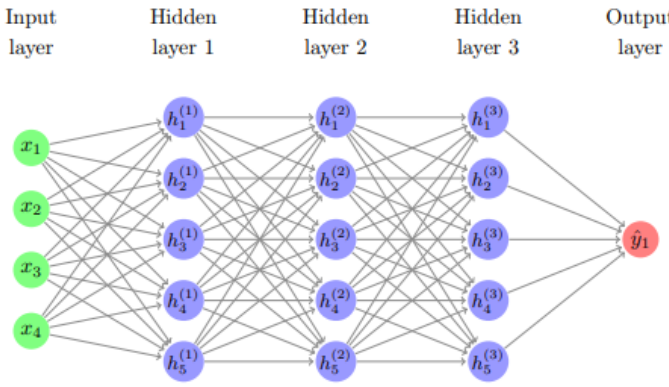

Figure 2.7. *Deep Neural Network architecture. Source: [6]*

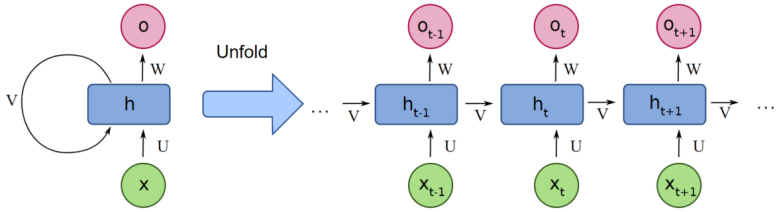

Figure 2.8. *The unfolded structure of an RNN. Source: medium.com*

2.5.2 Recurrent Neural Networks

The Recurrent Neural Network (RNN) are a family of neural networks for processing sequential data. An RNN is a neural network that is specialized for processing a sequence of values $\mathbf{x}_t$. Recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length. Unlike feedforward neural networks, that operate under the assumption that all training instances are independent, RNNs produce their output taking into account previously presented information. This allows them to exhibit temporal dynamic behavior and this is the reason why RNNs are used to model sequential data (for example text or audio). They are called recurrent because of their ability to process a sequence, performing the same task for every element.

RNNs are especially useful with sequential data because each neuron can use its internal memory to maintain information about the previous input. This memory mechanism of RNNs is implemented using an internal hidden layer that produces a hidden state, which remembers all information about what has been calculated. The mechanism is very important for tasks as natural language generation and speech recognition, where the model needs to remember the previous words in the sentence so as to understand the context or generate a new word. It makes RNNs applicable to tasks that require to remember the history of previous inputs and outputs. We can see the unfolded structure of a recurrent neural network is depicted in Figure 2.8.

At each timestep t the RNN receives an input vector x_t t and a hidden state from the previous timestep h_{t-1} and produces a new hidden state h_t and an output o_t. The hidden state at each timestep is updated based on the previous hidden state and the input vector,

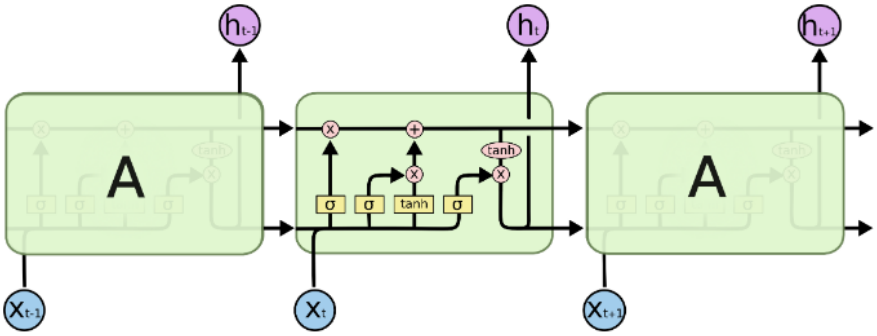

Figure 2.9. *The repeating module in an LSTM. Source: colah.github.io*

while the output depends on the current hidden state. The procedure is continued for all timesteps in the given sequence. The hidden state h_t and the output o_t for each timestep t are formulated as:

$$\begin{aligned} h_t &= f_t(V_t h_{t-1} + U_h x_t + b_h) \\ o_t &= f_o(W_o h_t + b_o) \end{aligned} \tag{2.20}$$

where f_h, b_h, and f_o, b_o are activation functions and biases for the hidden state and the output respectively, while U_h (input-to-hidden weights), V_h (hidden-to-hidden) and W_o (hidden-to-output), are learnable weight matrices.

While RNNs were very promising in handling sequential data, in practice they fall short when it comes to large contexts. As the sequence of information gets bigger and bigger, the RNN cannot continue to 'remember' all the crucial information. Here come the Long Short-Term Memory networks.

Long Short-Term Memory (LSTM): A subcategory of Recurrent Neural Networks (RNNs) described above, are the LSTMs, originally proposed by Hochreiter and Schmidhuber in 1997 [51]. They were proposed in order to overcome the shortcoming of RNNs when processing long sequences.

LSTM networks are a special kind of RNN, capable of learning long-term dependencies. They are explicitly designed to avoid the long-term dependency and vanishing gradient problems in RNNs through additional cell states and gates. In particular, when training a RNN using backpropagation, the gradients which are back-propagated can “vanish” (that is, they can tend to zero) or “explode” (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers, a phenomenon called as "gradient vanishing". They are able to effectively learn long-term dependencies and have largely replaces the simple RNN architecture. Intuitively, they solve the vanishing gradient problem through additional additive components, and forget gate activations, that allow the gradients to flow through the network without vanishing as quickly. Adding the LSTM to the network is like adding a memory unit inside the network that can remember context from the very beginning of the input. The LSTM architecture is depicted in Figure 2.9.

Given a sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t, ..., \mathbf{x}_n$ of vectors of an input sequence of length n, for vector $\mathbf{x}_t$, with inputs $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$, $\mathbf{h}_t$ and $\mathbf{c}_t$ are computed as follows:

$$
\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{u}_t &= tanh(\mathbf{W}_u \mathbf{x}_t + \mathbf{U}_u \mathbf{h}_{t-1} + \mathbf{b}_u) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{u}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t)
\end{aligned}
\tag{2.21}
$$

Forget Gate (f_t): This gate decides what information should be thrown away or kept. Information from the previous hidden state h_{t-1} and information from the current input x_t is passed through the sigmoid activation function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

Input Gate (f_{it}): The previous hidden state and current input are passed into a sigmoid function that decides which values will be updated by forcing the values to be between 0 and 1. The hidden state and current input are also passed to the *tanh* function to squish values between 1 and 1 (u_t). Finally, the *tanh* output is multiplied with the sigmoid output ($i_t \odot u_t$). The sigmoid output will filter the important information of *tanh*.

Cell State (c_t): To compute the next cell state, firstly the current cell state c_t gets point-wise multiplied by the result of the forget gate. This results in dropping information from the cell state that is not that important. Then, a point-wise addition is applied between previous result and the output from the input gate, that updates the cell state to new values that the neural network finds relevant.

Output Gate (o_t): The output gate determines the next hidden state. As the hidden state contains information from previous inputs, it is also used for predictions. After, passing the previous hidden state and the current input into a sigmoid function, the newly modified cell state is passed to the *tanh* function. We multiply the *tanh* output with the sigmoid output ($o_t \odot tanh(c_t)$) to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

2.5.3 Sequence-to Sequence Models

Sequence-to-sequence (seq2seq) is a family of machine learning approaches used for natural language processing, that was introduced by [52]. Seq2seq models generate an output sequence given an input sequence and the two sequences may have different lengths and not an explicit one to-one correspondence. Since the two sequences do not have the same structure, using simply a sequential architecture, like the RNN we described above, is not sufficient.

Seq2seq is based on an Encoder-Decoder framework. The encoder encodes the input sequence into a hidden (contextualized) representation, while the decoder takes this representation as input to generate the final output. In the original implementation, RNNs (specifically deep LSTMs) are used as the encoder and the decoder, while the encoded representation is a vector of fixed dimensionality.

The input sequence is denoted as $x_1, x_2, ..., x_n$ and the output sequence as $y_1, y_2, ..., y_m$

and the fixed size vector as c (context vector), we can formally express the task of generating the output sequence given the input, by the conditional probability:

$$P(y1, ..., y_m|x_1, ..., x_n) = \prod_{i=1}^{m} P(y_i|c, y_1, ..., y_{i-1}) \tag{2.22}$$

In this equation, each $p(y_i|c, y_1, ..., y_{i-1})$ distribution is represented with a softmax over all the words in the vocabulary. The encoder RNN processes the input sequence one token at a time and the final hidden state is used as the context vector c. The hidden state of the decoder RNN is initialized with c. Each sentence ends with a special end-of-sentence symbol "<EOS>", which enables the model to define a distribution over sequences of all possible lengths. The encoder RNN listens to the input tokens until it gets a special <EOS> token, and then the decoder RNN takes over and starts generating tokens, also finishing with its own <EOS> token.

2.5.4 Attention Mechanism

The Attention Mechanism was first introduced by [7], in order to address the bottleneck problem that arises with the use of a fixed-length encoding vector in the aforementioned architecture, where the decoder would have limited access to the information provided by the input. The attention mechanism develops a dynamic context vector by combining all the encoder hidden states, instead of discarding the intermediate hidden states of the encoder and using its final state as the context vector. Attention has become enormously popular, as an essential component of neural architectures, for many applications in Natural Language Processing, Speech, and Computer Vision. It allows the decoder to access the entire encoded input sequence and it can essentially, be interpreted as the ability to focus on relevant tokens of the input by computing weights of importance for their representations. This is the the central idea of the Attention Mechanism, which is depicted in Figure 2.10. It induces attention weights α over the input sequence to prioritize the positions where relevant information is present for generating the next output token.

We denote the input sequence as $x_1, x_2, ..., x_n$ and the output sequence as $y_1, y_2, ..., y_m$. The encoder is recurrent neural network, with a forward hidden state $\overrightarrow{\mathbf{h}_i}$ and a backward one $\overleftarrow{\mathbf{h}_i}$. A simple concatenation of two represents the encoder state. The motivation is to include both the preceding and following words in the annotation of one word. So it is: $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}^T; \overleftarrow{\mathbf{h}_i}^T]^T, i = 1, 2, ..., n$. At timestep i the decoder is an unidirectional RNN with hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ at timestep t. The context vector is a sum of hidden states of the input sequence, weighted by alignment scores:

$$\mathbf{c}_t = \sum_{i=1}^{n} a_{t,i}\mathbf{h}_i \tag{2.23}$$

where

$$a_{t,i} = align(y_t, x_i) = \frac{exp(score(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^{n} exp(score(\mathbf{s}_{t-1}, \mathbf{h}_{i'}))} \tag{2.24}$$

The alignment model assigns a score $a_{t,i}$ to the pair of input at position i and output at

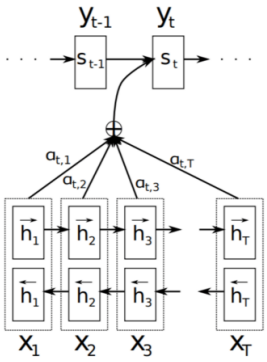

Figure 2.10. *The Attention Mechanism. Source: [7]*

position t, (y_t, x_i), based on how well they match. The set of $a_{t,i}$ are weights defining how much of each source hidden state should be considered for each output. In [7], the alignment score a is parametrized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model. The score function is therefore in the following form, given that tanh is used as the non-linear activation function:

$$score(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^T tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i]) \tag{2.25}$$

where both $\mathbf{v}_a$ and $\mathbf{W}_a$ are weight matrices to be learned in the alignment model. This type of attention mechanism is called additive attention. However this is not the only choice for the alignment function. There are many other choices, such as $score(\mathbf{s}_t, \mathbf{h}_i) = cosine[\mathbf{s}_t, \mathbf{h}_i]$ (content-based attention [53]) or $a_{t,i} = softmax(\mathbf{W}_a \mathbf{s}_t)$ (location-based attention [54]).

Another type of attention that we use in our work is **Self-attention** [55]. Self-attention, also known as intra-attention, is an attention mechanism Attention can be applied in a single sequence when there is no additional information and it allows it to relate different positions of itself in order to compute a representation of the same sequence. Self-attention is very effective in machine reading, abstractive summarization, or image description generation. To compute a specific output, we need the corresponding input and everything preceded that.

The simplest form of comparison between elements in a self-attention layer is a dot product. In every step, there is an element focus, i.e. the element whose similarity with the other tokens that is currently being computed. After the calculation of the scores of every set, a softmax layer is used that indicates the proportional relevance of each input to the target element. Finally, given these scores, the final output can be computed, by taking the average of the inputs that influence this output weighted by the scores generated by softmax. Formally, we have :

$$\begin{aligned} &score(x_i, x_j) = x_i \cdot x_j \\ &a_{i,j} = softmax(score(x_i, x_j)) = \frac{exp(score(x_i, x_j))}{\sum_{k=1}^{i} exp(score(x_i, x_k))}, \forall j \leq i \\ &y_i = \sum_{j \leq i} a_{ij} x_j \end{aligned} \tag{2.26}$$

Every input embedding (x_i) can be one of:

1. **query** The target element of the process. This element is then compared to every preceding input.

2. **key** Every preceding input that is compared to the current target element.

3. **value** The element that is used to compute the final output.

According to this, we can define the key, query, value definitions that are the key concepts of the attention mechanism.

In order to incorporate the above in the attention mechanisms, three matrices are introduced : $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$. The process is parallelized and therefore the matrices' multiplication take place at the same time. Having the input sequence into a single matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, we can produce matrices $\mathbf{Q} \in \mathbb{R}^{N \times d}$, $\mathbf{K} \in \mathbb{R}^{N \times d}$ and $\mathbf{V} \in \mathbb{R}^{N \times d}$ by:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}}; \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}}; \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

Self-attention is computed as:

$$SelfAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{2.27}$$

2.5.5 Transformers

Despite the ability of LSTMs to mitigate the loss of distant information due to the recurrence in RNNs, the underlying problem remains. Passing information forward through an extended series of recurrent connections leads to a loss of relevant information and to difficulties in training. Moreover, the inherently sequential nature of recurrent networks inhibits the use of parallel computational resources. These Transformers considerations led to the development of Transformers [1].

The Transformer is a deep learning model introduced by Vaswani et al. [8], used primarily in Natural Language Processing. It is an encoder-decoder architecture. The encoder and the decoder consist of N stacked layers, where the output of each layer is the input to the next. The layers are identical, with different weights. Each encoder layer consists of a self-attention and a fully connected feedforward sublayer interleaved by a residual connection and layer normalization. The decoder has the same structure, with the addition of a cross-attention sublayer between the self-attention and the feed-forward. The self-attention sublayer is capable of creating a contextualized representation of the sequence, by analyzing the dependency between tokens of the sequence. The cross-attention sublayer is responsible for analyzing the dependency between the input and the output sequences. The self-attention in the encoder is "bidirectional", while the self-attention of the decoder is "unidirectional". The output of the last decoder layer is finally converted to token probabilities, using a linear transformation and a softmax. The transformer architecture is presented in Figure 2.11.

The two Feed Forward Networks are fully connected and are linear transformations with a ReLU activation function in between:

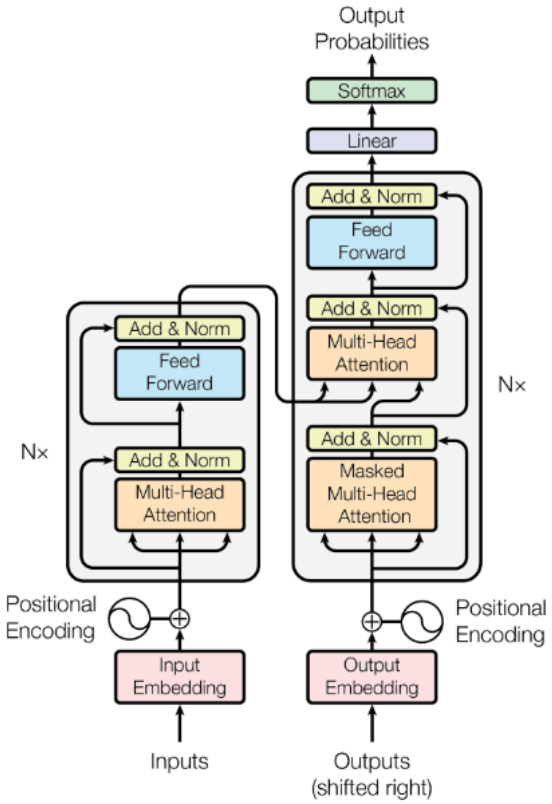

Figure 2.11. *The Transformer architecture. [8]*

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.28}$$

The layers of the Transformer encoder apply the same linear transformations to all the words in the input sequence, but each layer employs different weight (W_1, W_2) and bias (b_1, b_2) parameters to do so.

Positional Encoding The positional embeddings of the Transformer are used to give a sense of order in the sequences. They are ultimately a mapping of the sequence index to a vector. The information of absolute and relative positions of the tokens is encoded in their vector representations. This is done via summation of a position-representative vector to each input embedding, called the positional encoding. This vectors' function over the inputs must both be periodic and have variable frequency, to model both relative and absolute distance. The following sinusoidal function covers these requirements, and additionally gives the advantage of being able to scale to unseen lengths of sequences.

$$\begin{aligned} PE(pos, 2i) &= sin(pos/1000^{2i/d}) \\ PE(pos, 2i+1) &= cos(pos/1000^{2i/d}) \end{aligned} \tag{2.29}$$

Scaled dot-product Attention The Transformer implements a scaled dot-product attention, which follows the procedure of the general attention mechanism that we described in the previous section. As the name suggests, the scaled dot-product attention first com-

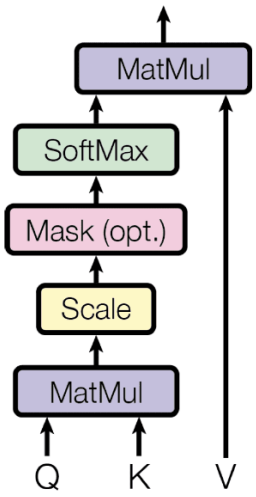

Figure 2.12. *Scaled-dot product attention. [8]*

putes a dot product for each query, q, with all of the keys, k. It subsequently divides each result by $\sqrt{d_k}$ and proceeds to apply a softmax function. In doing so, it obtains the weights that are used to scale the values, v. It is depicted in Figure 2.12. The computations performed by the scaled dot-product attention can be efficiently applied to the entire set of queries simultaneously. In order to do so, the matrices **Q,K** and **V**, are supplied as inputs to the attention function:

$$attention(\mathbf{Q},\mathbf{K},\mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{V}^T}{\sqrt{d_k}})\mathbf{V} \tag{2.30}$$

where $\sqrt{d_k}$ is the dimension of keys. The scaling factor $\sqrt{d_k}$ was introduced to counteract the effect of having the dot products grow large in magnitude for large values of d_k, where the application of the softmax function would then return extremely small gradients that would lead to the infamous vanishing gradients problem. The scaling factor, therefore, serves to pull the results generated by the dot product multiplication down, preventing this problem.

Multi-Head Attention The multi-head attention mechanism, that was proposed by [8], linearly projects the queries, keys, and values h times, using a different learned projection each time. The single attention mechanism is then applied to each of these h projections in parallel to produce h outputs, which, in turn, are concatenated and projected again to produce a final result.

The idea behind multi-head attention is to allow the attention function to extract information from different representation subspaces, which would otherwise be impossible with a single attention head [56]. The multi-head attention is given by:

$$MultiHead(\mathbf{Q},\mathbf{K},\mathbf{V}) = [head_1; ...; head_h]\mathbf{W}^O \tag{2.31}$$

where $head_t = Attention(\mathbf{Q}\mathbf{W}_t^Q, \mathbf{K}\mathbf{W}_t^K, \mathbf{V}\mathbf{W}_t^V)$ In the original paper it hold that $h = 8 and d_q = d_k = d_v = d/h = 64$. All the above weight matrices are learned by the model.

Chapter 3

Video Summarization

3.1 Introduction

The amount of video data that is produced on a daily basis is growing at an exponential rate. Given this growth, the assistance that users need, to browse extensive collections of videos and to extract key information from them, is becoming more immediate day by day. Video summarization aims to provide a short visual summary of an original, full-length video, that encapsulates the flow of the story and the most important segments of the original video. The goal is for the produced summary to retain only the significant parts of the video and to be non-redundant, with as little unnecessary content as possible [14]. The produced summary is usually composed of a set of representative video frames (a.k.a. video key-frames), or video fragments (a.k.a. video key-fragments) that have been stitched in chronological order to form a shorter video. Existing video summarization techniques explore a plethora of properties that a good summary should capture, designing criteria that the algorithm should prioritize when deciding which subset of frames (or subshots) to select [57]. In this chapter, we present a brief review of video summarization types and recent approaches.

3.2 Video Summarization Types

3.2.1 Dynamic and Static Video Summarization

There are two types of Video Summarization techniques when considering the type of the produced summary. Most video representation and summarization approaches that have appeared in the literature rely on static arrangements of key frames [58]. Specifically, a set of key frames is selected from each video shot and spatially arranged in a variety of pictorial summary forms. This is called a *Static Video Summary* or a Story Board. Such compact representations of video provide viewers with a global picture of the entire video content on a single screen. However, a common drawback of static video summaries are that they do not preserve the time-evolving dynamic nature of video content. Furthermore, these static representations are not natural to non-experts and can be hard to grasp on a single screen, particularly when the underlying video is complex.

In recent literature, the focus has shifted to generating summaries as shorter videos by processing both the visual and audio content. This is called *Dynamic Video Summarization*

or Video Skimming, which improves the information conveyed by the summary, as the generated shorter videos, called skims, consist of video segments and corresponding audio information [59]. The fundamental idea of video skim which is a short video composed of informative scenes from the original video presented to the user to be able to receive an abstract of the video story, but in video format [60]. The key difference between static and dynamic summaries is the presence of motion and audio information in the latter. Some of the key benefits of video skimming/dynamic video summarization, according to [59] are:

1. Conveying the plot of the video in shorter time.
2. Reduction in transmission time for videos browsed over the Internet.
3. Increases storage space utilization, by storing the video in its summarized form.
4. Assimilation of information conveyed through multiple videos belonging to a topic.

One advantage of video skims over static sets of frames is the ability to include audio and motion elements that offer a more natural story narration and potentially enhance the expressiveness and the amount of information conveyed by the video summary. Besides, it is often more entertaining and interesting for the viewer to watch a skim rather than a slide show of frames [61].

3.2.2 Video Summarization Training Strategies

Concerning the adopted training strategy, the existing deep-learning-based video summarization algorithms can be coarsely categorized in the following categories:

Supervised Video Summarization Supervised approaches rely on datasets with human labeled ground-truth annotations, either in the form of video summaries or in the form of frame-level importance scores, based on which they try to discover the underlying criterion for video frame/fragment selection and video summarization [14]. Models trained with supervision learn the transformation that produces summaries similar to those manually produced. Supervised video summarization does not bypass the need for annotated datasets, the number of which are limited [62] and their production expensive.

Early deep-learning-based approaches cast summarization as a structured prediction problem and try to make estimates about the frames' importance by modeling their temporal dependency. During the training phase the Summarizer gets as input the sequence of the video frames and the available ground-truth data that indicate the importance of each frame according to the users' preferences. These data are then used to model the dependencies among the video frames in time and estimate the frames' importance [63]. Some techniques, besides the temporal information, also pay attention to the spatial structure of the video and also model the spatiotemporal dependencies [64].

Unsupervised Video Summarization Unsupervised video summarization methods learn to summarize videos without the need for ground-truth summaries or user anno-

tations. Early unsupervised video summarization approaches use machine learning algorithms to analyze and cluster unlabeled data sets. [65] is one of the earliest unsupervised video summarization methods. It extracts color features from the video and then performs k-medoid clustering to acquire the key-frames [66]. Recent works ([2, 67, 11]) learn summarization by fooling a discriminator when trying to discriminate the original video from a summary-based reconstruction of it [14]. Without any guidance, most existing unsupervised approaches rely on the rule that a representative summary ought to assist the viewer to infer the original video content. In this context, these techniques utilize GANs to learn how to create a video summary that allows a good reconstruction of the original video, which is the approach we also follow in this work.

Other unsupervised video summarization methods learn summarization by targeting specific desired properties for the summary, such as diversity, representativeness, uniformity, non-redundancy and preservation of spatiotemporal patterns. Also, building object-oriented summaries by modeling the key-motion of important visual objects is a technique that many works use [14]. For example, [68] focuses on finding the important objects of the video and their key-motions and produces a summary that preserves the underlying semantic and motion information of the video. Even though unsupervised video summarization bypasses the need for annotated data, a drawback of this strategy is the requirement of large amounts of data, in order to train the models effectively.

Reinforcement Learning for Video Summarization Reinforcement Learning in Video Summarization is based on the use of hand-crafted rewards about specific properties of the produced summary. These rewards usually aim to increase the representativeness, diversity [69] and uniformity [70] of the summary, retain the spatiotemporal patterns of the video, or secure a good summary-based reconstruction of the video [71]. Recently, deep reinforcement learning has been explored for video summarization, where the summarizer creates a summary by predicting frame-level importance scores and the evaluator quantifies the existence of the desired characteristics with the help of the reward functions.

3.2.3 Unimodal and Multimodal Video Summarization

Unimodal video summarization approaches utilize only the visual modality of the videos for feature extraction and learn to summarize in a supervised or unsupervised manner. On the other hand, multimodal methods exploit the the available textual or audio metadata and learn semantic/category-driven summarization in a supervised way by increasing the relevance between the semantics of the summary and the semantics of the associated metadata or video category [14]. For example, [72] uses action classifiers that have been trained with video-level labels, to perform action-driven video fragmentation and labeling. This method extracts a fixed number of key-frames and utilizes reinforcement learning to select the frames with the highest categorization accuracy, performing category-driven summarization.

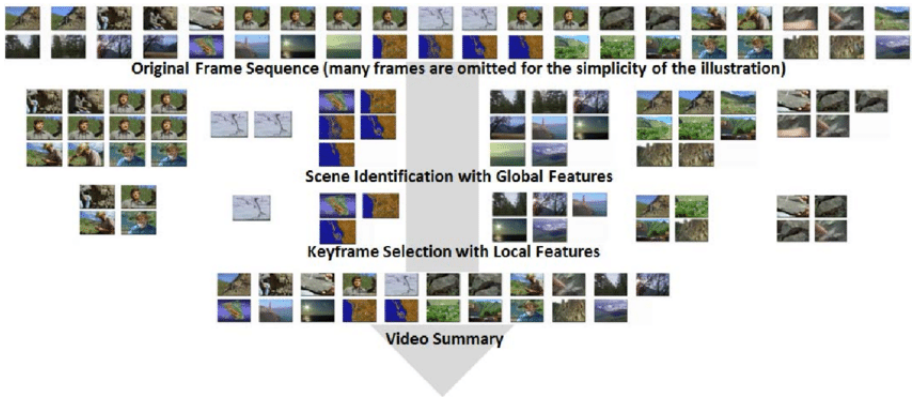

Figure 3.1. *Feature-based video summarization. Source: [9]*

3.3 Video Summarization Approaches

Multiple techniques have been developed for video summarization, which can be classifies into four major categories, based on their properties and characteristics, that are presented subsequently.

3.3.1 Feature-Based Video Summarization

Video modality contains information that is related to many features, such as color, motion, audio, visual, events, objects etc. The original video content can be represented as an aggregation of these various features. The goal of feature-based video summarization is to process these features, in order to select only a subset of the original video frames that will represent the summary. The feature extraction and aggregation is among the most important steps in this method and the selection process can be applied in different levels of detail. Usually, the video is firstly divided into shots or scenes, that represent segments, and then the most important key-segments are determined and constitute the summary. The overview of this approach is depicted in Figure 3.1. Several feature-based video summarization techniques exist, that include the utilization of features that represent events, objects, color and motion, plus attention-based techniques [73]. These approaches are detailed below.

3.3.1.1 Event-Based Video Summarization

Event-based approaches are useful for the identification of normal and abnormal events, that are part of a video and which are considered interesting and important for the summary. For example, tracking and recognizing sudden changes appearing in the environment like robbery scenes, by using detection models to observe suspicious/abnormal features is a form of event-detection video summarization.

Existing work on abnormal event detection can be roughly categorized into trajectory analysis and volume matching [74]. Trajectory analysis-based abnormal event detection approaches, track the targets in the videos, stabilize the positions into trajectories, and then recognize whether they are abnormal events or not. For example, [75] introduces a motion

descriptor based on optical flow measurements in a spatiotemporal volume for the tracked human targets, and uses an associated similarity measure in a nearest-neighbor framework to recognize the event. Volume matching-based abnormal event detection approaches apply a spatiotemporal volume localization scheme to search for the position of abnormal events. An example is [76], who propose using the volumetric features of the video for event detection.

3.3.1.2 Object-Based Video Summarization

In object-based Video Summarization Techniques we are interested in detecting specific objects from the video, such as cars, people, animals etc. The objective is to collect all the frames which include the desired objects, thus an object detector is required to analyze each scene [77]. Zhang et al. [68] developed a method that focuses on the preservation in the summary of the underlying fine-grained semantic and motion information of the video. They find the important objects of the video and their key-motions and the video is represented by creating super-segmented object motion clips. These clips are given to the summarizer, which memorizes past states of object motions by continuously updating a recurrent auto-encoder network, that reconstructs the clips and guides the training of the summarizer.

3.3.1.3 Color-Based Video Summarization

Color-based video summarization uses the color features of the video frames to select the key-frames. In most works the video sequence is clustered in a predefined number of shots based on color histogram, computed for every frame [78]. The frames of the original video that are similar to each other, are clustered in together into a matrix using color histograms in the HSV color space. A Delauna Triangulation diagram is built followed by edge detection and extraction using the Delaunay diagram. Finally, the extracted edges are used to generate the clusters and the frame that is closest to the centroid of a cluster is selected as a key-frame.

3.3.1.4 Attention-Based Video Summarization

Attention-based video summarization focuses on identifying the parts of the video that capture most the user's interest. In this method, the system assigns higher scores to the key frames or shots that exist in an interesting region and that will eventually constitute the summary. There are various ways to pinpoint the parts of the video hold most of the users' interest [73]. For example, [79] achieve highlight detection by motion attention modeling. Representing the video in a complete undirected graph, it is partitioned into video clusters, which form a directed temporal graph, in which a shortest path algorithm is applied, in order to detect the video scenes. The attention values of each scene are computed and attached to the scenes, clusters, shots, and subshots in a temporal graph, that can inherently describe the evolution and perceptual importance of a video.

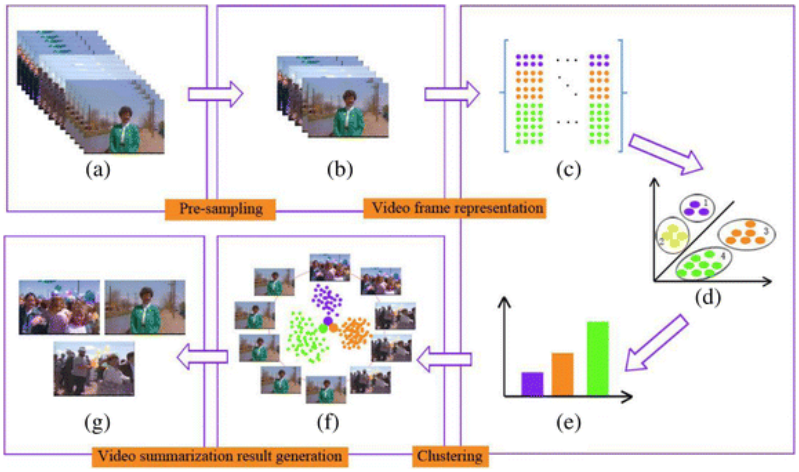

Figure 3.2. *Clustering-based Video Summarization. Source: [10]*

3.3.2 Clustering-Based Video Summarization

Clustering-based video summarization detects similar activities or properties within video frames. The idea beyond these approaches is to split the frames of a given video into different groups such that frames that belong to the same group are more similar among themselves. After extracting the most suitable frame features, the key-frames are detected based on the cluster centroids. The approach can be seen in Figure 3.2. This method is also useful when the objective is to eliminate frames that have irregular trends and are considered outliers. The clustering methods we will focus on are K-Means Clustering and Spectral Clustering.

3.3.2.1 K-Means Clustering

K-Means clustering is a method of vector quantization, that aims to partition n observations into K clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster [80]. In video summarization K-Means clustering is used for the selection of the key-frames or key-segments of the video. [81] utilizes HOG descriptors of Gabor maps of the input frames, a high level representation of which, is created using sparse auto-encoders. A video summary is generated from the candidate frames that are obtained by K-Means clustering. K-Means clustering is not suitable for videos whose scenes are static, because it can create redundancy.

3.3.2.2 Spectral Clustering

A clustering technique that has been increasingly growing recently is the spectral clustering [82], due to the fact that it produces more satisfactory results than those obtained by classic clustering algorithms. Its application in video summarization is limited but there have been some works that utilize spectral clustering for key-frame selection [83] and shot boundary detection [84]. In multivariate statistics spectral clustering techniques utilize the eigenvalues of the similarity matrix of the data to perform dimensionality reduction, before clustering in fewer dimensions. It is a technique based on graph theory, where the

approach is used to identify communities of nodes in a graph based on the edges connecting them. In this setting, [82] proposes a method that after the feature extraction, detects the shot boundaries by firstly estimating the number of shots, based on the number k of the clusters used. Following the shots number estimation, a spectral clustering algorithm is executed for the keyframe extraction stage, based on the descriptor feature vectors and an affinity matrix is constructed, based on its eigenvectors the K-means algorithm is run to cluster the frames according to the shots to which they are associated.

3.3.3 Sparse Dictionary Learning

Dictionary learning is a sparse encoding schema. Sparse learning is a representation learning method whose objective is to find a sparse representation of the input data, in the form of a linear combination of the basic elements of the data. We refer to the columns of the full row-rank $m \times n$ matrix $A = [a_1, ..., a_n] \in \mathbb{R}^{m \times n}, n >> m$, as a dictionary and they are assumed to be a set of vectors able to provide a highly succinct representation for most statistically representative signal vectors $y \in \mathbb{R}^m$, where $y = Ax$ [85]. We refer to a sparse solution x, as a sparse coding representation of the signal instantiation, y.

In video summarization, this technique is used to obtain frames which have the highest weights in reconstruction of a video data. The extracted key-frames form a dictionary and enable optimal reconstruction of the input video from the selected dictionary. Thus, the video summary is framed as the set of key-frames that can linearly reconstruct the full-length video. In general, adjacent frames share similar visual content. As a result, candidate key-frames occur in blocks in sparse dictionary based representation. When sparse selection is performed upon such block sparsity in video summarization, temporally neighboring frames are encapsulated as a 'frame block' and the most representative are selected so that neighboring frames cannot be selected as key-frames at the same time [86].

An example is portrayed in [87], who summarize a video into a few key objects by selecting representative object proposals generated from video frames. This representative selection problem is formulated as a sparse dictionary selection problem, i.e. choosing a few representatives object proposals to reconstruct the whole proposal pool. Each of the object proposal can be represented by a feature vector and dictionary learning is used to find the representatives of the video.

Chapter 4

Proposed Method

4.1 Problem Definition

In this work, we tackle video summarization to produce short dynamic summaries for videos, as a key-segment selection problem. Adopting an unsupervised approach, we utilize the visual features of the video frames in order to build a key-frame selection mechanism, which will produce a summary that will optimally represent the input video. Our goal is for the produced summary to encapsulate the flow of the story and the most important segments of the video, retaining only the most significant parts and containing as little unnecessary content as possible.

Several methods have been proposed to tackle video summarization using information extracted from the audio, video and text modalities. Early approaches rely on the statistical processing of low-level video, audio and text features for assessing frame similarity or performing clustering-based key-frame selection [88], while the detection of the salient parts of the video is achieved using motion descriptors, color histograms and eigen-features. Given the recent growth of neural network architectures, many deep learning based video summarization frameworks have been proposed over the last years.

Deep-learning based video summarization algorithms typically represent visual content as feature vector encodings of video frames extracted from Convolutional Neural Networks (CNNs) [14]. One of the challenges in video summarization is learning the complex temporal dependencies among the video frames. Early temporal modeling approaches used Long Short-Term Memory (LSTM) units [51], or in general, sequence-to-sequence models such as Recurrent Neural Networks (RNNs) [89, 25]. The introduction of transformers allowed for parallel computation, as well as, better modeling of the long-range temporal dependencies among the video frames [27]. Generative Adversarial Networks (GANs) [90] have also been used in video summarization algorithms. In [2], an adversarial framework is proposed consisting of a summarizer and a discriminator, both of which were based on LSTMs. GAN-based video summarization algorithms have been shown to produce state-of-the-art results, as presented in [14]. Recently, the utilization of attention mechanisms has appeared to be very effective in identifying the important parts of the videos [14, 91, 24, 12], a central task in video summarization.

In our work, we adopt a GAN-based approach and build upon the SUM-GAN model proposed in [2]. Furthermore, motivated by the limited memorization, long-range and

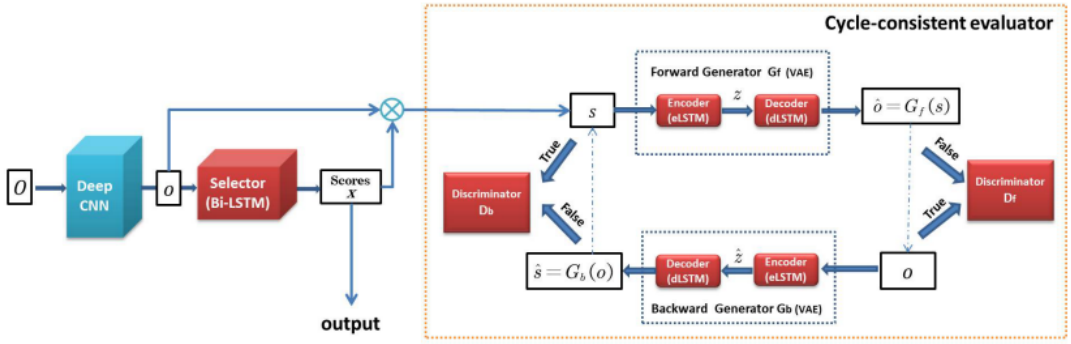

Figure 4.1. *Cycle-SUM architecture. Source: [11]*

temporal modeling capacity of the LSTM, as well as the success of the multi-head attention and transformer architectures in overcoming these drawbacks [27, 26, 28], we extend SUM-GAN by integrating attention mechanisms in several parts of the architecture. We perform an ablation study to identify the importance of better temporal modeling in the frame selection, encoder and decoder of SUM-GAN, experimenting on each of the resulting models. Based on the above results, in SUM-GAN-AED, we propose the use of a self-attention mechanism as the frame-selector, while we retain the LSTM architecture for the encoder and the decoder. All the models and the experiments are presented in detail in the following sections.

We assess the performance of our models on two benchmark datasets, SumMe [16] and TVSum [17] and we also build a third dataset, COGNIMUSE, from the COGNIMUSE database [18], which is used for further model evaluation. Finally, we analyze our findings, compare them with other state-of-the-art approaches and present our conclusions.

4.2 Related Work

There are several works which use Generative Adversarial Networks (GANs), in order to produce video summaries in an unsupervised setting. Yuan et al. [11] proposed an approach that aims to maximize the mutual information between the summary and the video, using a trainable couple of discriminators and a cycle-consistent adversarial learning objective. The frame selector is a bi-directional LSTM and builds a video summary by modeling the temporal dependency among the video frames. This summary is then forwarded to the evaluator which is composed of two GANs; the forward GAN is used to learn how to reconstruct the original video from the video summary, and the backward GAN tries to learn how to perform the backward reconstruction from the original to the summary video. The consistency between the output of such cycle learning is used as a measure that quantifies information preservation between the original video and the generated summary. Using this measure, the evaluator guides the frame selector to identify the most informative frames and form the video summary. The model is called Cycle-SUM and they evaluate it on two benchmark datasets, TVSum [17] and SumMe [16]. The overview of the architecture is illustrated in 4.1.

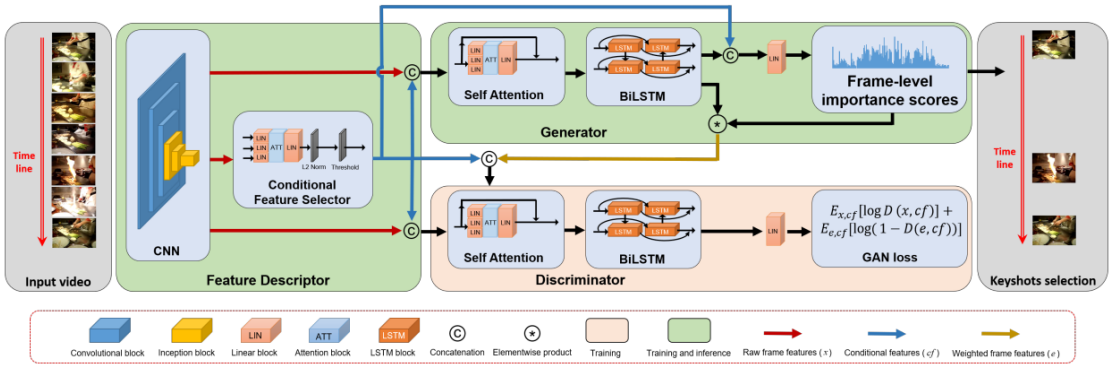

Figure 4.2. *An overview of the Attentive Conditional GAN framework. Source: [12]*

He et al. [12] propose a novel video summarization framework, with attentive conditional GANs. The proposed baseline GAN model is extended to the conditional GAN model by utilizing conditional information provided by a feature selector to focus on more important frames of the input video. The framework consists of a generator and a discriminator, as illustrated in 4.2. The generator tries to predict the frame-level importance scores and produces weighted frame features (i.e., e in 4.2). Then, raw frame features (i.e., x in 4.2) and weighted frame features are treated as real and fake inputs for the discriminator to distinguish. A multi-head self-attention module is introduced to capture long-range temporal dependencies throughout the whole video sequence, as a complementary guidance to the Bidirectional LSTM (BiLSTM). A conditional feature selector is used to guide the GAN model to focus on more important temporal regions of the whole video frames (i.e., cf in 4.2). The model is evaluated on the SumMe[16] and TVSum[17] datasets.

Apostolidis et al. [67], propose an architecture that embeds an Actor-Critic model into a GAN and formulates the selection of important video fragments (that will be used to form the summary) as a sequence generation task. The Actor and the Critic take part in a game that incrementally leads to the selection of the key video fragments. Their choices at each step of the game result in a set of rewards from the Discriminator. The designed training workflow allows the Actor and Critic to discover a space of actions and automatically learn a value function (Critic) and a policy for key-fragment selection (Actor). This method establishes a link between GANs and reinforcement learning approaches, as it uses the Discriminator's feedback to train the summarizer. Figure 4.3 shows the architecture of the proposed AC-SUM-GAN model. The model is, also, evaluated on the SumMe and TVSum datasets.

Building on this model and on a public available implementation of a variation of [2] that includes a linear compression layer to reduce the number of learned parameters and applies an incremental approach for training the different components of the architecture, Apostolidis et al. [13] propose a stepwise, label-based learning process to improve the training efficiency of the adversarial part of the model. The proposed variation of the SUM-GAN model is shown on figure 4.4 and it is called SUM-GAN-sl. The 3-step incremental training approach updates specific parts of the network in each step. In particular, differently to the immediate update of the entire model based on the computed losses after

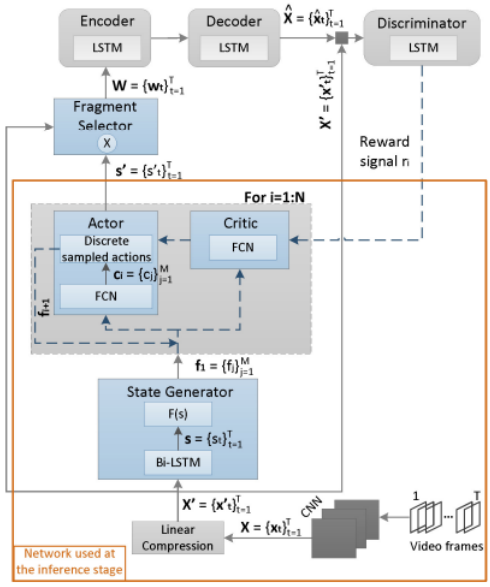

Figure 4.3. *The AC-SUM-GAN architecture.*

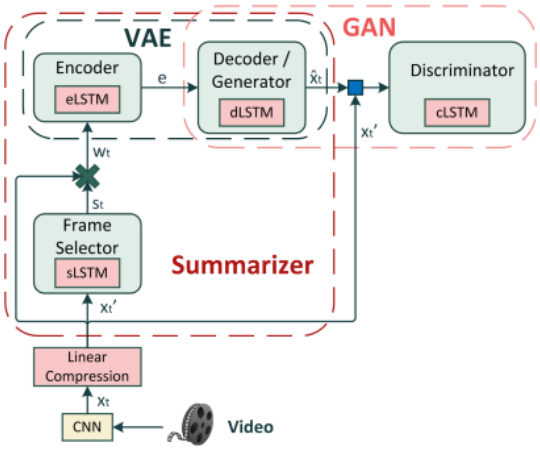

Figure 4.4. *The SUM-GAN-sl variation of the SUM-GAN model. Source:[13]*

a single forward pass of the architecture, the implemented process performs three passes in each epoch, updating different losses during each one. In the same way, the training of the discriminator is performed in a stepwise manner. This incremental process enables a more fine-grained computation of the discriminator's gradients compared with the training policy used in SUM-GAN, and helps the discriminator develop higher discrimination efficiency, thus performing better during the classification.

4.3 Baseline Model

4.3.1 Main Components

The basic generative adversarial framework [2], is illustrated in Figure 4.5. The frame selector, encoder and decoder jointly constitute the Summarizer, while the decoder (gen-

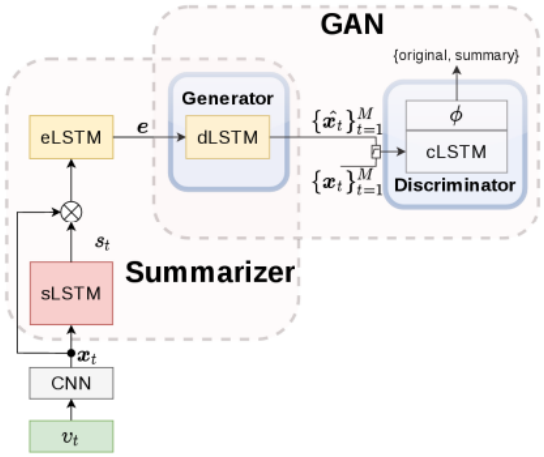

Figure 4.5. *Main components of the model's architecture. Source: [2]*

erator) along with the discriminator constitute the Generative Adversarial Network. The encoder and the decoder also form a Variational Auto-Encoder (VAE), which aids training by producing an underlying representation of the video and introducing an additional frame scores vector [20]. The approach suggests a keyframe selection mechanism, that minimizes the distance between the features of the original videos and the videos that result as a reconstruction from the predicted summaries. The summarizer and the discriminator are trained adversarially in an unsupervised manner, until the discriminator is not able to distinguish between the reconstructed videos from summaries and the original videos.

More precisely, the frame selector is a bi-directional LSTM and the encoder and decoder are LSTMs. Let $\mathbf{X} \in \mathbb{R}^{M \times N}$ be the frame features of the input video, derived from the CNN, where M is the number of frame features and N the number of frames. $\mathbf{x}_i \in \mathbb{R}^M$, $i \in [1, N]$, is the feature vector that describes the i^{th} frame. At each instance i, the selector is fed with $\mathbf{x}_i$ and outputs a normalized importance scores vector $\mathbf{s} \in \mathbb{R}^N$, where $\mathbf{s}_i \in [0, 1]$. The weighted frame features, $\mathbf{x}_i\mathbf{s}_i$, correspond to the summary and are fed into the encoder which results in a hidden state vector, $\mathbf{e} \in \mathbb{R}^H$. The decoder takes $\mathbf{e}$ as input and reconstructs a sequence of features representing the input video $\hat{\mathbf{x}} \in \mathbb{R}^N$. Finally, $\hat{\mathbf{x}}$ is forwarded to the LSTM based discriminator, which aims to classify it as 'original' or 'summary'.

Proposed by Kingma et al. [20], the VAE defines a posterior distribution over the observed data, given an unobserved latent variable. Let $e \sim p_e(e)$ be a prior over the unobserved latent variable, and x be the observed data. We can interpret $\mathbf{e}$ as the encoding of $\mathbf{x}$ and define $q(\mathbf{e}|x)$ as the probability of observing e given $\mathbf{x}$. It is typical to set $e \sim p_e(e)$ as the standard normal distribution. Similarly, $p(x|e)$ identifies the conditional generative distribution for x, given e. The VAE is trained by minimizing the negative log-likelihood of the data distribution:

$$-log\frac{p(x|e)p(e)}{q(e|x)} = \underbrace{-log(p(x|e))}_{\mathcal{L}_{reconst}} + \underbrace{\mathcal{D}_{KL}(q(e|x)||p(e))}_{\mathcal{L}_{prior}} \tag{4.1}$$

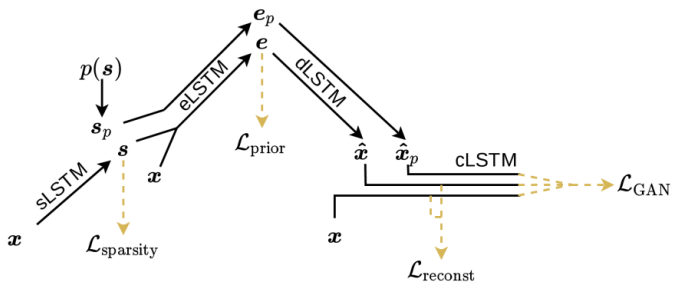

Figure 4.6. *The four loss functions used in the model training. Source:[2]*

Mahasseni et al. [2] is the first that combines an LSTM-based key-frame selector with a Variational Auto-Encoder (VAE) and a trainable discriminator.

4.3.2 Training Approach

The training of the model specifies the summarizer's parameters, $\{\theta_s, \theta_\epsilon, \theta_d\}$, that correspond to the frame selector, encoder and decoder, and the GAN's parameters, $\{\theta_d, \theta_c\}$, which define the decoder and classifier. As it is illustrated in figure 4.6, it is defined by four loss functions: the Loss of GAN, $\mathcal{L}_{GAN}$, the Reconstruction Loss, $\mathcal{L}_{reconst}$, the Regularization Loss $\mathcal{L}_{sparsity}$ and the Prior Loss, $\mathcal{L}_{prior}$.

The key idea behind the proposed generative-adversarial training by [2], is to introduce an additional frame selector $\mathbf{s}_p$, governed by a prior distribution (e.g., uniform distribution), $\mathbf{s}_p \sim p(\mathbf{s}_p)$, which is achieved through the variational autoencoder. When we sample the input video frames with $\mathbf{s}_p$ we get a subset which is passed to encoder, producing the representation $\mathbf{e}_p$. Given $\mathbf{e}_p$, the decoder reconstructs a video sequence $\hat{\mathbf{x}}_p$, which is used to regularize learning of the discriminator, such that the classifier is highly accurate on recognizing $\hat{\mathbf{x}}_p$ as the 'summary' class, but that it confuses $\hat{\mathbf{x}}$ as the 'original' class. Mahasseni et al. [2], similar to the training of the GAN models proposed by Goodfellow et al. [90] and Larsen et al. [92], formulates an adversarial learning algorithm that iteratively optimizes the following three objectives:

1. To learn $\theta_s, \theta_\epsilon$, minimize $\mathcal{L}_{reconst} + \mathcal{L}_{prior} + \mathcal{L}_{sparsity}$.
2. To learn θ_d we minimize $\mathcal{L}_{reconst} + \mathcal{L}_{GAN}$.
3. To learn θ_c we maximize $\mathcal{L}_{GAN}$.

Let us now define each of the above loss functions. The sparsity loss is used for regularization and penalizes having a large number of key frames selected in the summary. It is denoted as:

$$\mathcal{L}_{sparsity} = \| \frac{1}{N} \sum_{t=1}^{N} s_t - \sigma \|_2 \tag{4.2}$$

where M is the total number of video frames and σ is the summary rate, an input hyper-parameter representing a percentage of frames that are to be selected in the summary.

The Prior Loss is defined as:

$$\mathcal{L}_{prior} = \mathcal{D}_{KL}(q(\mathbf{e}|\mathbf{x})||\mathcal{N}(0,1))) \tag{4.3}$$

where $\mathcal{D}_{KL}$ is the Kullback–Leibler divergence and it denotes a measure of how one probability is different from a second.

The Reconstruction Loss is denoted as:

$$\mathcal{L}_{reconst} = \mathbb{E}[-\log p(\phi(\mathbf{x})|\mathbf{e})] \tag{4.4}$$

where expectation $\mathbb{E}$ is approximated as the empirical mean of training examples, $\phi(\mathbf{x})$ is the output of the last hidden layer of the discriminator and $p(\phi(\mathbf{x})|\mathbf{e}) \propto exp(- \parallel \phi(\mathbf{x}) - \phi(\hat{\mathbf{x}}) \parallel^2)$. The Reconstruction Loss ensures that the summary maintains the main information of the original video and minimizes the information loss between $\mathbf{x}$ and $\hat{\mathbf{x}}$. The standard practice in learning encoder-decoder networks is to use the Euclidean distance between the input and decoded output, for estimating the reconstruction error, however recent studies demonstrate the drawbacks of this method [92]. That is why the Reconstruction Loss is formulated as an expectation of a log-likelihood $logp((\phi(\mathbf{x})|\mathbf{e}))$ and it is calculated based on the last hidden layer of the discriminator LSTM.

The Adversarial Loss is denoted as:

$$\mathcal{L}_{GAN} = log(LSTM(\mathbf{x})) + log(1 - LSTM(\hat{\mathbf{x}})) + log(1 - LSTM(\hat{\mathbf{x}}_p)) \tag{4.5}$$

where LSTM$(\cdot)$ is the binary soft-max output of the classifier and it represents the discriminator's confidence when classifying the original video, the generated summary and the uniform summary.

Given the above definitions, the parameters $\theta_s, \theta_\epsilon, \theta_d$ and θ_c are updated using the Stohastic Gradient Variational Bayes estimation [20], [92] adapted for recurrent networks.

4.3.3 Testing Approach

After training, the components responsible for generating a summary for an unseen video are the linear compression layer and the frame selector. In particular, the CNN features of the video frames pass through the aforementioned components and an importance score is computed for each frame. Then, having the video fragmented using the KTS [93] algorithm, segment-level importance scores are calculated by averaging the importance scores of each segment's frames. Finally, the summary is created by selecting the segments that maximize the total importance score, provided that the summary length does not exceed 15% of the original video duration, a convention adopted by several video summarization approaches [25, 63, 17]. This latter step is performed by solving the following optimization problem:

$$max \sum_{i=1}^{N} a_i b_i, s.t. \sum_{i=1}^{N} a_i l_i \leq 0.15L, a_i \in \{0,1\} \tag{4.6}$$

where N is the number of segments, L is the length of the original video and 0.15 defines the upper limit for the summary duration. Given the i^{th} segment of the video, a_i is a binary value that indicates whether the segment is selected or not, b_i is the computed segment-level importance score, and l_i is the length of the segment. The latter is the 0/1 Knapsack problem [94] which is defined as follows: given a set of n items and a *knapsack*, with

$$p_j = \text{profit of item j},$$
$$w_j = \text{weight of item j},$$
$$\text{c} = \text{capacity of the knapsack},$$

select a subset of the items so as to:

$$\text{maximize } z = \sum_{j=1}^{n} p_j x_j \tag{4.7}$$

$$\text{subject to } \sum_{j=1}^{n} w_j x_j \leq c, \tag{4.8}$$

$$x_j = 0 \text{ or } 1, j \in N = \{1, ..., n\} \tag{4.9}$$

where

$$x_j = \begin{cases} 1 & \text{if item j is selected} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

Here our "*knapsack*" is the video summary and our items are the segments of the whole video. We want to maximize the segment (frame groups) weight that will fit in our summary, while keeping its length 15% of the size of the video.

4.3.4 SUM-GAN

Our implementation of the baseline model is SUM-GAN and it's built using the code implementation[1] of [2]. The model architecture is depicted in figure 4.5 and it is implemented using Python 3.7 [21] and PyTorch 1.0.1 [22]. Building on this code, we removed the linear compression layer that is placed before the frame selector and the stable GAN training this version uses; that is for the first 15 steps of each epoch, the discriminator's parameters are fixed. While trying to reproduce the exact architecture that is described in Mahasseni et al. [2], we encountered the following inconsistency: the original paper mentions that the hidden size of the encoder and decoder LSTMs is 2048, however the original input size of $\mathbf{x}_t$, before any processing, is 1024, according to the output of pool5 layer of the GoogLeNet network[23]. The classifier LSTM takes as input both $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$, which leads us to an inconsistent input size. To avoid this conflict, we decided to reduce the hidden size of both the encoder and decoder LSTMs to 1024. This is the only change that we have made, on the implementation of [2] as it is described in the paper to the best of our knowledge. Ultimately, each component of the network is comprised of a 2-layer LSTM, with 1024 hidden units in each layer.

[1] https://github.com/j-min/Adversarial$_V ideo_S ummary$

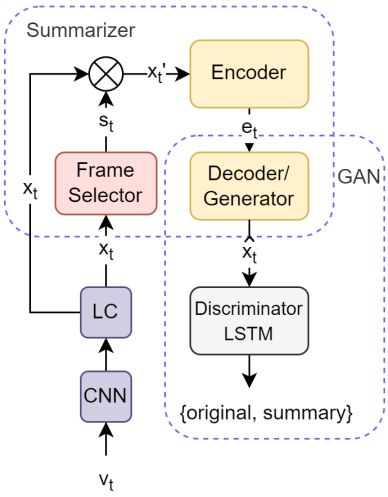

Figure 4.7. *The SUM-GAN-LC model architecture.*

4.3.5 SUM-GAN-LC

In SUM-GAN-LC, we add a linear compression layer to the architecture that is depicted in 4.5, that reduces the number of trainable parameters, as it is shown in 4.7. Reducing the size of the input vectors, in return leads to a performance improvement, as it is noted by Apostolidis et al. [13]. The linear compression layer significantly reduces the amount of trained parameters and it advances the model's training capacity in case of small datasets (as for SumMe), while a lower impact is observed in the case of larger datasets (as for TVSum). However, it is possible that the amount of training samples in the case of TVSum is adequate for learning a larger set of parameters [13]. Each component of the architecture is comprised of a 2-layer LSTM, with 500 hidden units in each layer.

4.4 Proposed Models

In this section we describe in detail our proposed model, SUM-GAN-AED, as well as the variants of it. We present a Video Summarization model that is based on the SUM-GAN architecture [2]. We build on a publicly available implementation of a variation of this work and we conduct the experiments described in the following subsections. The training and testing of our model and its variants follow the baseline. In addition to this, we train and test our models on three datasets, SumMe [16], TVSum [17] and COGNIMUSE [18], for each of the experiments.

4.4.1 SUM-GAN-AED

Our approach introduces an attention-based frame selector in the GAN-based architecture. The model is inspired by [24] and [25]. The self-attention layer ranges over the entire video sequence and efficiently captures long-term temporal dependencies, as opposed to using an LSTM that fails to capture such dependencies effectively[26]. This leads to faster computation and a more representative frame selection scores vector [26]. The architecture of the proposed SUM-GAN-AED model is depicted in Figure 4.8.

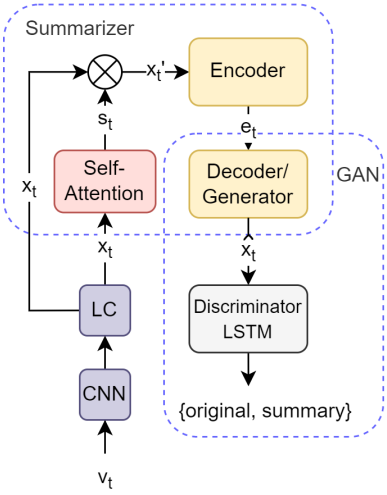

Figure 4.8. *The architecture of the SUM-GAN-AED model.*

We used the self-attention class of the slp framework [2] for our self-attention module. The module has an attention size (hidden size) of 500 and an input size of 1024. Following the linear compression layer, the compressed frame features sequence, enters the self-attention module, as depicted in 4.8. The key, query and value are computed and the attention scores are calculated using single-head scaled dot product attention. Finally, the attention scores are multiplied with the values. The scaled dot attention scores are defined as:

$$a = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V} \tag{4.11}$$

where Q is the queries tensor, K is the keys tensor and V is the values tensor and d is the model dimension. Then the frame scores multiplied by the frame features continue as a sequence to the VAE and the rest of the training and testing continues as described in section 4.3.

4.4.2 Model Variants

Continuing, we further investigate the effectiveness of adding self-attention layers in the SUM-GAN model, as it pertains to capturing long-term temporal dependencies in the input video. Specifically, we experimented with replacing the LSTMs in the SUM-GAN-AED architecture with transformers at various places. The following changes are proposed: replacing the LSTM in the encoder with various flavors of a transformer (SUM-GAN-STD, SUM-GAN-STSED) and replacing the LSTMs in the encoder and the decoder with a transformer (SUM-GAN-SAT, SUM-GAN-ST). For the SUM-GAN-STD, SUM-GAN-STSED and SUM-GAN-ST models we also use a bi-directional LSTM instead of a self-attention module as the frame selector. The evaluation of the various LSTM, self-attention and transformer variations serves as an ablation study that highlights which modules in the architecture benefit the most from the improvement of the temporal dependencies modeling. The proposed models are detailed next.

[2] https://github.com/georgepar/slp

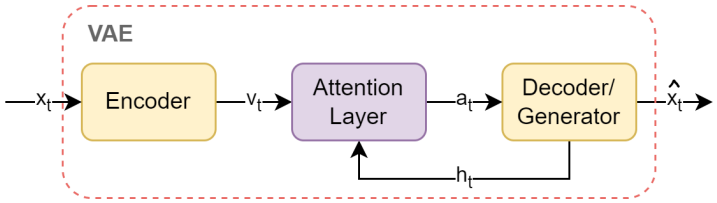

Figure 4.9. *The VAE architecture of our SUM-GAN-SEAD model.*

4.4.2.1 SUM-GAN-SEAD

In SUM-GAN-SEAD we introduce an attention mechanism in the Variational Autoencoder module of the architecture, as it is depicted in figure 4.9. The idea behind this is to implement a gradual decision making approach that bases the selection of data from a data sequence, on the previously seen ones. This model was inspired by Apostolidis et al. [24]. The former extended the VAE in the original SUM-GAN-sl model, with variational attention. In particular, the attention weights of each frame were considered as random variables and a latent space was computed by the VAE for these values, too, so the decoder updates its hidden states based on both latent spaces. This employed an encoder-attention-decoder network to tackle video summarization, thus again the attention mechanism allows the decoder to selectively focus on only a subset of inputs, by increasing their attention weights.

In our implementation we integrated the attention mechanism after the encoder and before the encoder, as well. The attention component receives the encoder output $\mathbf{v}_t$ and the previous hidden state of the decoder $\mathbf{h}_t$, it calculates the attention energy vector $\mathbf{e}_t$ and finally applies a soft-max function to normalize the attention energies producing the attention weight vector $\mathbf{a}_t$. The attention weight vector is then fed into the decoder, which combines it with its output from the previous frame, so as to incrementally reconstruct the video. The attention weights are calculated based on the score function:

$$\mathbf{e}_t^i = \mathbf{v}_i^* \mathbf{W}_a h_{t-1} \tag{4.12}$$

where $\mathbf{v}_i^*$ is s the transposed encoder output for the i^{th} video frame, $\mathbf{h}_{t-1}$ is the hidden state for the decoder for $t-1$, $\mathbf{W}_a$ is a learnable parameter and $\mathbf{e}_t^i$ is the relevance score (scalar value) before normalization [24]. The final attention weights $\mathbf{a}_t^i$ are computed based on the following normalization:

$$\mathbf{a}_t^i = \frac{\exp\left(\mathbf{e}_t^i\right)}{\sum_{j=1}^{n} \exp\left(\mathbf{e}_t^j\right)} \tag{4.13}$$

4.4.2.2 SUM-GAN-STD

In SUM-GAN-STD, we swap the encoder LSTM of [2] for a Transformer [8]. The frame selector is a bi-directional LSTM and the decoder and discriminator are LSTMs as in [2]. The use of a transformer was inspired by a number of works that have utilized the transformer architecture for the task of video summarization [27, 28] and by the Transformer's

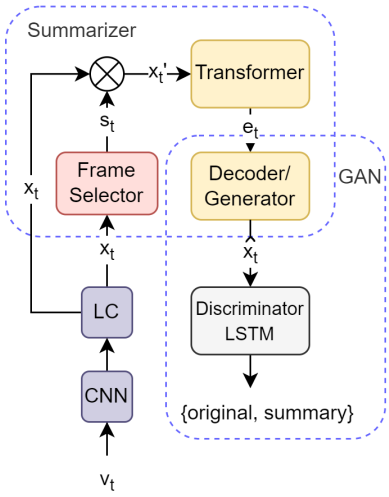

Figure 4.10. *The SUM-GAN-STD architecture.*

ability to address the shortcomings of the recurrent machine translation systems, since it relies on self-attention [8]. As it is explained by Narasimhan et al. [29] the transformer can also capture the global dependencies among a sequence, which leads to better modeled relationships among the frames.

In our model we used the PyTorch implementation of the Transformer model [22]. The SUM-GAN-STD architecture is shown in figure 4.10. Following the linear compression layer and the frame selector, the weighted feature vector is forwarded to the Transformer, whose output is fed to the decoder and then to the discriminator LSTM. The output of the transformer is the concatenated output of each of the Multi-Head Attention heads and is formulated as:

$$\mathbf{y}_t = \text{ concat(head1, head2,..., headN) } \mathbf{W}_o \tag{4.14}$$

where

$$head_i = Attention(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{K}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}}) \tag{4.15}$$

for each attention head of the transformer, where $\mathbf{K}, \mathbf{V}, \mathbf{Q}$ are the key, value and query matrices.

4.4.2.3 SUM-GAN-ST

In SUM-GAN-ST we replace the Variational Auto-Encoder, i.e. the encoder and the decoder, with a transformer, in order to enhance the video reconstruction by integrating the positional information of the frames. The sequence-to-sequence architecture is suitable for video summarization. The weighted frame features enter the transformer and the reconstructed frame sequence that corresponds to the input video is fed into the discriminator, in order to be classified as 'original' or 'summary'. Since we remove the VAE, during training we do not utilize the Prior Loss; the transformer is trained as part of the summarizer and the GAN. The architecture of SUM-GAN-ST can be seen in Figure 4.11.

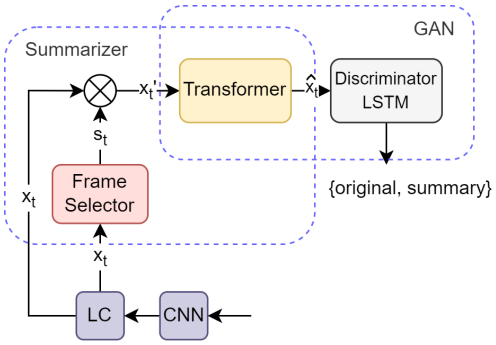

Figure 4.11. *The SUM-GAN-ST architecture.*

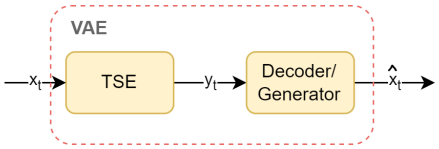

Figure 4.12. *The SUM-GAN-STSED architecture.*

4.4.2.4 SUM-GAN-STSED

In SUM-GAN-STSED we replace the encoder of the architecture with a *Transformer Sequence Encoder* (TSE), the part of the transformer module that constitutes the encoder. The intuition remains, a more effective representation of the long-range temporal video dependencies and as a result of the hidden state vector **e**, which leads to a better video reconstruction. For its implementation, we use the slp framework[3]. The TSE is a sequence-to-vector architecture, which uses positional encodings to insert relative position information of the sequence tokens and keep track of the ordering of the frames. The architecture is shown in Figure 4.12. For each video, TSE takes the output of the frame selector multiplied by the features vector, forwards it to a linear layer, computes the positional embeddings and adds them to the tensor. The resulting vector is forwarded to the encoder part of the TSE and the output is fed to the decoder LSTM of the architecture. The frame selector is an LSTM, and the training follows the description in Section 4.3.

For the positional encodings, sine and cosine functions of different frequencies are used. More specifically for even positions it is denoted as:

$$\text{PosEncoder}(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{4.16}$$

and for odd positions it is denoted as:

$$\text{PosEncoder}(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{4.17}$$

where pos is the frame position, i is the embedding index and d is the model dimension.

[3]https://github.com/georgepar/slp

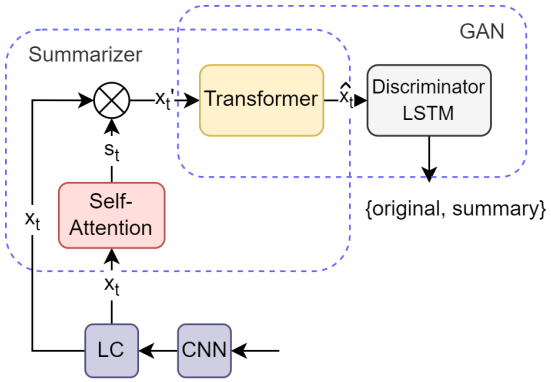

Figure 4.13. *The SUM-GAN-SAT architecture.*

4.4.2.5 SUM-GAN-SAT

Finally, motivated by the effectiveness of the attention based modules, we build SUM-GAN-SAT, in which we swap the LSTM frame selector with a self-attention module and the Variational Auto-Encoder with a transformer, as it is depicted in figure 4.13. This model combines the SUM-GAN-AED and SUM-GAN-ST architectures and their advantages in integrating attention-based modules in the GAN-based architecture. Here, we incorporate the more thorough frame selection that self-attention provides, with the integration of the positional encodings information during the video reconstruction phase. We do not use Prior Loss during training, which follows the training of the preceding models.

4.5 Datasets

In this section we will introduce the datasets we will be using to train and test our models. As it is stated in [14], the main datasets that prevail in video summarization bibliography are SumMe [16], Title-based Video Summarization (TVSum) [17], OVP and Youtube [65]. In the present work we base our experiments and results on the datasets SumMe and TVSum, along with the COGNIMUSE, which we build from the COGNIMUSE database [18] and we use as our third dataset.

4.5.1 SumMe and TVSum

The SumMe dataset, introduced by Gygli et al. [16], consists of 25 videos covering holidays, events and sports, captured from both first-person and third-person view. They are raw or minimally edited user videos and they have a high compressibility compared to already edited videos. The length of the videos ranges from about 1 to 6 minutes. Each video has been annotated by 15 to 18 users in the form of key-fragments and thus is associated to multiple fragment-level user summaries that have a length between 5% and 15% of the initial video duration. Moreover, besides the aforementioned user summaries, a single ground-truth summary in the form of frame-level importance scores (calculated as an average of the key-fragment user summaries per frame) is also provided.

TVSum, created by Song et al. [17], consists of 50 videos, 1 to 11 minutes long, and their shot-level importance scores, obtained via crowdsourcing. It contains videos

from 10 categories of the TRECVid Multimedia Event Detection (MED) task [19], 5 per category, that were obtained from YouTube, using the category as a search query term. The collected videos represent various genres, including news, how-to's, documentaries, and user-generated content (vlog, egocentric). TVSum videos have been annotated by 20 users in the form of shot and frame-level importance scores (ranging from 1 to 5). Similar to SumMe, a single ground-truth summary in the form of frame-level importance scores (computed after averaging all users' scores) is provided for each video of the dataset.

4.5.2 COGNIMUSE

The COGNIMUSE database, built by Zlaintsi et al. [18], is a video-oriented database multimodally annotated with sensory and semantic saliency, audio and visual events, cross-media relations, as well as emotion. It was created as a framework that would assist the training and evaluation of event detection and summarization algorithms, regarding their accuracy in detecting salient events as well as for content analysis, with respect to the included annotation schemes. The dataset consists of half-hour continuous segments (with the final shot/scene included) from seven Hollywood movies (three and a half hours in total), which are "A Beautiful Mind" (BMI), "Chicago" (CHI), "Crash" (CRA), "The Departed" (DEP), "Gladiator" (GLA), "Lord of the Rings-the Return of the King" (LOR), and the animation movie "Finding Nemo" (FNE). They include basic concepts, such as the main character/s, the desire, and the conflict as well as typical features such as music, vivid color variations, audio and visual effects, speed of action, etc., which are used as a powerful tool for developing the plot, leading therefore to effective summaries. The seven movie segments are annotated with sensory and semantic saliency, audio-visual events and emotion. These seven movie segments comprise the foundation for the COGNIMUSE dataset that is used in our study.

4.6 Evaluation Metrics

For a fair comparison with the state of the art, the key-shot-based metric used by Mahasseni et al. [2] is used for the evaluation of the results, on the TVSum and SumMe datasets. Let A be the generated key-shots and B the user-annotated keyshots. The precision and recall are defined based on the amount of temporal overlap between A and B as follows:

$$Precision = \frac{\text{overlap duration of } A \text{ and } B}{\text{duration of } A} \tag{4.18}$$

$$Recall = \frac{\text{overlap duration of } A \text{ and } B}{\text{duration of } B} \tag{4.19}$$

F-score Video F-score is introduced in [95], in three different settings: canonical, augmented and transfer. After that many other studies followed, benchmarking their approaches using F-score. The F-score formula is detailed below.

The harmonic mean F-score can be obtained by:

$$F - Score = \frac{2P \times R}{P + R} \times 100\% \tag{4.20}$$

where P and R represent the precision and recall respectively.

We follow the method presented in [95], to convert frame-level scores to key-frame and key-shot summaries in TVSum and SumMe. To generate key shots for datasets which only provide key frame scores, the videos are initially temporally segmented into disjoint intervals using KTS [93], as previously stated. The resulting intervals are ranked based on their importance scores, where the importance score of an interval is equal to the average score of the frames in that interval. A subset of intervals is selected from the ranked intervals as keyshots, such that the total duration of the generated key shots are less than 15% of the duration of the original video. This is conducted for both predicted and ground truth frame level scores, for TVSum and SumMe, and the f-score is computed, based on the two resulting summaries.

For the evaluation on the dataset COGNIMUSE, we follow the evaluation method proposed in [96]. Since COGNIMUSE dataset doesn't include ground truth frame scores, the summarization task is approached as a two-class classification problem, where multiple thresholds are applied to the estimated frame-wise importance scores, in order to obtain results for various compression rates and thus produce summaries of various lengths. Then, the AUC metric is computed, derived from the ROC curve that results by applying the different thresholds.

4.7 Experiments

4.7.1 Data Pre-processing

In order for the above datasets to be used in the training and evaluation of our models, all the videos were downsampled to 2 fps and for a fair comparison with other works, for example [2], we used the output of pool5 layer of GoogLeNet [23] trained on ImageNet [97], for representing the visual content of the video frames. The Hierarchical Data Format (HDF5) of the datasets, was used as an input to our models. For the datasets SumMe and TVSum the HDF5 files were publicly available. Each HDF5 file contains a group for each video of the datasets, and each group contains the keys that are listed and described in table 4.1.

For the dataset COGNIMUSE we build the HDF5 file, following the method that is described as follows. To begin with, the COGNIMUSE HDF5 contains seven groups, one for each video. Each group, contains 7 keys: n_frames, features, the frame rate of each video, frames per second (fps), change_points, n_steps, picks, and the ground truth summary of each video, gt_summary. We created a script using Python [21] to extract information for each one of the aforementioned keys. We first downsampled the movie segments to 2 fps, from 25 fps, using FFmpeg [98], a framework that is able to decode, encode, transcode, mux, demux, stream, filter and play anything that humans and

Key Name	Description
n_frames	number of frames in original video
n_steps	number of subsampled frames
features	2D-array with shape (n_steps, feature-dimension)
gtscore	1D-array with shape (n_steps), stores ground truth importance scores
user_summary	2D-array with shape (num_users, n_frames), each row is a binary vector
change_points	2D-array with shape (num_segments, 2), each row stores indices of a segment
n_frame_per_seg	1D-array with shape (num_segments), indicates number of frames in each segment
picks	positions of subsampled frames in original video
gtsummary	1D-array with shape (n_steps), ground truth summary provided
video_name	original video name, only available for SumMe dataset

Table 4.1. *HDF5 file keys description.*

machines have created. To extract the representation of the frame features, we first convert each video to video frames, we extract its features and its change points, and we create the HDF5 file with the ground truth summary, as well as the rest of the keys.

To convert each video to video frames we used opencv [99], a library of Python bindings designed to solve computer vision problems, and to extract the frame features, which represent the visual information of the video frames, we used GoogLeNet [23], a convolutional neural network that is 22-layers deep, pretrained on ImageNet [97] dataset. In order to get the change points, which represent the video segments, the videos are temporally segmented into disjoint intervals using Kernel Temporal Segmentation (KTS). KTS, proposed by [93], is a kernel-based method that splits the video into a set of non-intersecting temporal segments. Change point detection usually focuses on constant one dimensional signals corrupted by noise, and the goal is to detect the jumps in the signal. It is able to statistically discriminate between jumps due to noise and jumps due to the underlying signal. Given the matrix of frame-to-frame similarities defined through a positive definite kernel, the algorithm outputs a set of optimal "change points" that correspond to the boundaries of temporal segments.

More precisely, let the video be a sequence of frame features $x_i \in \mathbf{X}, i = 0, 1, ..., n-1$, and let $K : \mathbf{X} \times \mathbf{X} \to \mathbb{R}$ be a kernel function between frame features. Let $\mathcal{H}$ be the feature space of the kernel $K(.,.)$. Denote $\phi : \mathbf{X} \to \mathcal{H}$, the associated feature map, and $\| \cdot \|_{\mathcal{H}}$ the norm in our feature space $\mathcal{H}$. We minimize the following objective

$$\underbrace{\text{Minimize}}_{m;t_0,...,t_{m-1}]} J_{m,n} := L_{m,n} + Cg(m,n) \tag{4.21}$$

where m is the number of change points and $g(m,n) = m(\log(n/m)+1)$ a penalty term that penalizes segmentations with too many segments. $L_{m,n}$ is defined from the within-segment

Hyperparameter	SUM-GAN	sl	LC	SEAD
number of epochs	50	100	50	50
input size	1024	1024	1024	1024
hidden size	1024	500	500	500
regularization factor	0.3	0.1	0.3	0.15
learning rate	0.0001	0.0001	0.0001	0.0001
discriminator lr	0.00001	0.00001	0.00001	0.0001

Table 4.2. *Model Hyperparameters for models: SUM-GAN, SUM-GAN-sl, SUM-GAN-LC and SUM-GAN-SEAD.*

kernel variances $v_{t_i,t_{i+1}}$:

$$L_{m,n} = \sum_{i=0}^{m} v_{t_i,t_{i+1}}, \;\; v_{t_i,t_{i+1}} = \sum_{t=t_i}^{t_{i+1}-1} \parallel \phi(x_t) - \mu_i \parallel_{\mathcal{H}}^2, \;\; \mu_i = \frac{\sum_{t=t_i}^{t_{i+1}-1} \phi(x_t)}{t_{i+1} - t_i} \tag{4.22}$$

and it measures the overall within segment variance. Increasing the number of segments decreases $L_{m,n}$, but increases the model complexity. This objective yields a trade-off between under- and over-segmentation. Following this, we create the HDF5 file, using the hdf5 library of Python h5py [100], with the rest of the features the insertion of which is trivial.

4.7.2 Experimental Setup

In all the experiments we follow the standard 5-fold cross validation approach, where 80% of the videos are used for training and 20% of the videos are used for testing. The results we report refer to the average performance over the 5 runs and we implemented every model using PyTorch [22]. The linear compression layer reduces the size of these feature vectors from 1024 to 500. Each component of the architecture comprises a 2-layer LSTM with 500 hidden units in each layer, while the frame selector LSTM is bi-directional. The training is online and based on the Adam optimizer and the learning rate for all components but the discriminator is 10^{-4}; for the latter it equals to 10^{-5}. The regularization factor for the sparsity loss is $\sigma = 0.3$. All the experiments are conducted on Google Colaboratory platform, using the integrated GPU with CUDA. In table 4.2 and 4.3 we summarize the main hyperparameters more specifically, for each model we train. The number of layers of each LSTM module used in our architectures is 2 and it remains the same for each model, thus it is not reported in the table.

4.7.3 Results and Comparison

Table 4.4 depicts the F-score results of the recent state-of-the-art unsupervised video summarization methods, in comparison with our model SUM-GAN-AED. As we can see, the proposed framework surpassed the state-of-the-art methods on SumMe, by a significant margin and on TVSum its performance is comparable to the current state-of-the-art.

Hyperparameter	AED	STD	TSE	ST
number of epochs	50	50	50	50
input size	1024	1024	1024	1024
hidden size	500	500	500	500
regularization factor	0.3	0.3	0.3	0.3
learning rate	0.0001	0.0001	0.0001	0.0001
discriminator lr	0.0001	0.00001	0.00001	0.00001
dropout	0.1	0.1	0.1	0.1
dimension	-	500	500	500
heads	-	5	5	5
encoder layers	-	2	6	2
decoder layers	-	2	6	2

Table 4.3. *Model Hyperparameters for models: SUM-GAN-AED, SUM-GAN-STD, SUM-GAN-TSE, SUM-GAN-ST and SUM-GAN-SAT.*

Model	SumMe	TVSum
SUM-GAN [2]	38.7	50.8
ACGAN [12]	46.0	58.5
Cycle-SUM [11]	46.8	57.6
SUM-GAN-sl [13]	46.8	**65.3**
SUM-GAN-AAE [24]	48.9	58.3
Proposed-B [30]	58.8	63.5
SUM-GAN-AED (Ours)	**64.85**	**63.18**

Table 4.4. *Comparative performance evaluation of our SUM-GAN-AED model, with state-of-the-art unsupervised key-frame extraction approaches, on SumMe and TVSum (F-score(%)).*

Model	SumMe	TVSum	COGNIMUSE
SUM-GAN reported	38.7	50.8	-
SUM-GAN reproduced	46.92	51.19	51.24
SUM-GAN-sl reported	46.8	**65.3**	-
SUM-GAN-sl reproduced	48.04	64.78	50.5
SUM-GAN-LC	57.99	61.74	49.99
SUM-GAN-STD	54.15	63.82	51.38
SUM-GAN-ST	56.00	60.53	50.73
SUM-GAN-STSED	61.30	62.73	52.8
SUM-GAN-SEAD	61.89	61.66	52.55
SUM-GAN-SAT	61.38	62.41	49.81
SUM-GAN-AED	**64.85**	**63.18**	**55.49**

Table 4.5. *Performance comparison of our baseline and proposed models on SumMe, TVSum (F-score(%)) and COGNIMUSE (AUC).*

The efficacy of SUM-GAN-AED is compared to that of the experimental models, developed as part of this study, that utilize attention and transformer architectures in Table 4.5. We, also, provide the results of the baseline models of Mahasseni et al. [2] and Apostolidis et al. [13], as well as their performance when we trained and tested them locally, in SUM-GAN reproduced and SUM-GAN-sl reproduced, respectively. Both tables showcase that the introduction of the attention mechanism, either with self-attention or a transformer, leads to an improvement of the performance on SumMe and TVSum, as well as COGNIMUSE, a result which confirms our initial motivation to utilize attention.

Between all experimental models, SUM-GAN-AED yields the best overall results, while taking in consideration all the datasets, which also shows that the performance peak is not dataset focused. The addition of the self-attention mechanism at the frame selection stage in the transformer-based architecture SUM-GAN-ST, leads to improved performance as evidenced by SUM-GAN-SAT. This result underlines the importance of the self-attention module in terms of overall performance, as well as the value of modeling the temporal information of the video frames, when computing their scores. It should be noted that the experimental transformer-based architectures perform consistently well on the evaluation datasets and outperform many state-of-the-art unsupervised approaches.

In general the scores of TVSum are higher than those of SumMe in each of the models, except SUM-GAN-SEAD. A contribution to this pattern would be the difference of the size of the datasets. TVSum has a total number of frames equal to 23510 and SumMe has 7336 frames in total. This allows TVSum to achieve better training, thus achieving a better performance score in the vast majority of the approaches.

We can remark from Table 4.5, that the best method in TVSum (SUM-GAN-sl reported, f-score = 65.3%) is highly adapted to this dataset, as the performance it exhibits on SumMe (f-score = 46.8%) is second to worst (SUM-GAN reported, f-score = 38.7%). The same pattern is observed for the performance of SUM-GAN-sl reproduced, which is to be expected.

So far, we have observed the positive results in improving performance, when we uti-

lize attention mechanisms for the Video Summarization task. In particular this outcome highlights the positive contribution of the attention mechanism, as well as the transformer in enhancing the summarizer's ability to identify the most important frames, and in effectively guiding the learning of the adversarial component of the architecture. We would argue that despite the not optimal performance on TVSum, the models that are based on attention mechanisms and transformers, outperform the ones based only on LSTMs, since the performance is not dataset focused, but rather they achieve high accuracy scores on all the datasets.

Chapter 5

Conclusions

5.1 Discussion

In this Diploma Thesis we explore the task of unsupervised Video Summarization. Motivated by the need to efficiently browse and retrieve dense amounts of information, we focus on Video Summarization as a key-segment selection problem, using the visual information of the video to model the video frames and rank the individual segments.

To this end, we develop an end-to-end video summarization model, which given an input video, generates the corresponding summary as a set of video key-segments. We propose a novel framework for unsupervised video summarization based on a Generative Adversarial Network. Building on the SUM-GAN model [2], we utilize the attention mechanism, with the introduction of self-attention and transformer modules into our framework, in order to capture long-range dependencies, adapt to sequence lengths not encountered in training and embed the positional information of the video frames. We evaluate this model with further experiments, building models that integrate the attention mechanism in different parts of their architectures, which act as an ablation study. We investigate, the efficacy of the attention mechanism as the frame selector, encoder or encoder-decoder of the architecture, on improving the performance of the model and modeling more efficiently the temporal dependencies, as well as correlations of the data.

We also build a dataset that can be used for video summarization, derived from the database COGNIMUSE. We evaluate our models on this dataset, in addition to the two benchmark datasets on which we train and test each of our proposed architectures.

The results of our experiments indicate that the use of self-attention for frame selection, followed by LSTMs for encoding and decoding, enhances the performance of the basic GAN-based video summarization framework, as we achieve results that perform consistently well on the evaluation datasets and outperform many of the state-of-the-art unsupervised video summarization approaches.

5.2 Future Work

Reaching the end of this thesis, we would like to propose some ideas that can be explored in the future. Given the current state of the art in automated video summarization, we believe that progress should target primarily the development of unsupervised deep learning

methods, which are able to be trained effectively without the need for collecting human annotations.

We encourage the following points to be explored in future work:

- We could target the addition of further criteria concerning different attributes of the generated summary, such as visual diversity, uniformity [70], which can be done by introducing additional rewards that relate to these.
- With respect to the utilized data modality, it would be very beneficial to take advantage the audio modality of the video which could be a very valuable source of information as well. It would be a lot easier to identify the most critical parts of a movie or a video when we take into account the audio of the data. To add to that, textual metadata of a video could also be used to further enhance the frame selection process.
- It would also be very interesting to intervene with with the summary production process, in order to force a desired outcome by applying some user rules, for example create a summary that contains all the cars, or the people, etc. To this direction, the summary could take into account user text queries.
- Another future objective would be to train the models in sufficiently big datasets. Because annotated data are expensive and hard to obtain, one alternative is to train large datasets of data that are not paired, but have common domains or storylines. Another, would be to adapt current datasets meant for other computer vision tasks, to the requirements of video summarization.
- Finally, we can further improve the performance of video summarization models, by improving the evaluation protocols to achieve a more accurate future works comparison. The evaluation methods could develop to also include qualitive summary evaluation from the user.

Bibliography

[1] Daniel Jurafsky και James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1stη έκδοσηη, 2000.

[2] Behrooz Mahasseni, Michael Lam και Sinisa Todorovic. *Unsupervised Video Summarization With Adversarial LSTM Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[3] Rukshan Pramoditha. *The Concept of Artificial Neurons (Perceptrons) in Neural Networks*, 2021.

[4] Anders Krogh. *What are artificial neural networks? Nature Biotechnology*, 26(2):195–7, 2008.

[5] DeepAI. *What is a Feed Forward Neural Network?*

[6] Pérez-Enciso και Laura Zingaretti. *A Guide for Using Deep Learning for Complex Trait Genomic Prediction. Genes*, 10:553, 2019.

[7] Dzmitry Bahdanau, Kyung Hyun Cho και Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser και Illia Polosukhin. *Attention Is All You Need*, 2017.

[9] Genliang Guan, Zhiyong Wang, Kaimin Yu, Shaohui Mei, Mingyi He και David Dagan Feng Feng. *Video Summarization with Global and Local Features*. σελίδες 570–575, 2012.

[10] Jiaxin Wu, Sheng Hua Zhong, Jianmin Jiang και Yunyun Yang. *A Novel Clustering Method for Static Video Summarization. Multimedia Tools Appl.*, 76(7):9625–9641, 2017.

[11] Li Yuan, Francis E. H. Tay, Ping Li, Li Zhou και Jiashi Feng. *Cycle-SUM: Cycle-consistent Adversarial LSTM Networks for Unsupervised Video Summarization. CoRR*, abs/1904.08265, 2019.

[12] Xufeng He, Yang Hua, Tao Song, Zongpu Zhang, Zhengui Xue, Ruhui Ma, Neil Robertson και Haibing Guan. *Unsupervised Video Summarization with Attentive*

Conditional Generative Adversarial Networks. Proceedings of the 27th ACM International Conference on Multimedia, MM '19, σελίδα 2296–2304, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Evlampios Apostolidis, Alexandros I. Metsai, Eleni Adamantidou, Vasileios Mezaris και Ioannis Patras. *A Stepwise, Label-Based Approach for Improving the Adversarial Training in Unsupervised Video Summarization. Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, AI4TV '19, σελίδα 17–25, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Evlampios E. Apostolidis, Eleni Adamantidou, Alexandros I. Metsai, Vasileios Mezaris και Ioannis Patras. *Video Summarization Using Deep Neural Networks: A Survey. CoRR*, abs/2101.06072, 2021.

[15] Hoang Trinh, Jun Li, Sachiko Miyazawa, Juan Moreno και Sharath Pankanti. *Efficient UAV video event summarization. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, σελίδες 2226–2229, 2012.

[16] Michael Gygli, Helmut Grabner, Hayko Riemenschneider και Luc Van Gool. *Creating Summaries from User Videos. Computer Vision – ECCV 2014*David Fleet, Tomas Pajdla, Bernt Schiele και Tinne Tuytelaars, επιμελητές, σελίδες 505–520, Cham, 2014. Springer International Publishing.

[17] Yale Song, Jordi Vallmitjana, Amanda Stent και Alejandro Jaimes. *TVSum: Summarizing Web Videos Using Titles. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] Zlatintsi A., Koutras P. και Evangelopoulos G.et al. *COGNIMUSE: a multimodal video database annotated with saliency, events, semantics and emotion with application to summarization. Journal on Image and Video Processing*, vol. 54 (2017).

[19] George Awad, Asad A. Butt, Keith Curtis, Jonathan Fiscus, Afzal Godil, Yooyoung Lee, Andrew Delgado, Jesse Zhang, Eliot Godard, Baptiste Chocot, Lukas Diduch, Jeffrey Liu, Yvette Graham, Gareth J. F. Jones, και Georges Quénot. *Evaluating Multiple Video Understanding and Retrieval Tasks at TRECVID 2021. Proceedings of TRECVID 2021.* NIST, USA, 2021.

[20] Diederik P Kingma και Max Welling. *Auto-Encoding Variational Bayes*, 2013.

[21] Guido Van Rossum και Fred L Drake Jr. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai και Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library. Ad-*

vances in Neural Information Processing Systems 32, σελίδες 8024–8035. Curran Associates, Inc., 2019.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke και Andrew Rabinovich. *Going Deeper with Convolutions. CoRR*, abs/1409.4842, 2014.

[24] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I. Metsai, Vasileios Mezaris και Ioannis Patras. *Unsupervised Video Summarization via Attention-Driven Adversarial Learning. MultiMedia Modeling*Yong Man Ro, Wen Huang Cheng, Junmo Kim, Wei Ta Chu, Peng Cui, Jung Woo Choi, Min Chun Hu και Wesley De Neve, επιμελητές, σελίδες 492–504, Cham, 2020. Springer International Publishing.

[25] Zhong Ji, Kailin Xiong, Yanwei Pang και Xuelong Li. *Video Summarization With Attention-Based Encoder–Decoder Networks. IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1709–1717, 2020.

[26] Manjot Bilkhu, Siyang Wang και Tushar Dobhal. *Attention is all you need for Videos: Self-attention based Video Summarization using Universal Transformers. CoRR*, abs/1906.02792, 2019.

[27] Bin Zhao, Maoguo Gong και Xuelong Li. *Hierarchical Multimodal Transformer to Summarize Videos. CoRR*, abs/2109.10559, 2021.

[28] Jeiyoon Park, Kiho Kwoun, Chanhee Lee και Heuiseok Lim. *Multimodal Frame-Scoring Transformer for Video Summarization*, 2022.

[29] Medhini Narasimhan, Anna Rohrbach και Trevor Darrell. *CLIP-It! Language-Guided Video Summarization. CoRR*, abs/2107.00650, 2021.

[30] Michail Kaseris, Ioannis Mademlis και Ioannis Pitas. *Exploiting Caption Diversity for Unsupervised Video Summarization. ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, σελίδες 1650–1654, 2022.

[31] Ian J. Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[32] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[33] Xiaojin Zhu, Andrew B. Goldberg, Ronald Brachman και Thomas Dietterich. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, 2009.

[34] Richard S. Sutton και Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[35] Pádraig Cunningham, Matthieu Cord και Sarah Jane Delany. *Supervised Learning*, σελίδες 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[36] Zoubin Ghahramani. *Unsupervised Learning*, σελίδες 72–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[37] Richard S. Sutton και Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, secondη έκδοσηη, 2018.

[38] Shoba Ranganathan, Michael Gribskov, Kenta Nakai και Christian Schönbach, επιμελητές. *Encyclopedia of Bioinformatics and Computational Biology - Volume 2*. Elsevier, 2019.

[39] F. Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. Psychological Review*, σελίδες 65–386, 1958.

[40] William Stafford Noble. *What is a support vector machine? Nature Biotechnology*, 24:1565–1567, 2006.

[41] Douglas C. Montgomery, Elizabeth A. Peck και G. Geoffrey Vining. *Introduction to linear regression analysis*. Wiley series in probability and statistics. Wiley, New York, NY [u.a.], 3. edη έκδοσηη, 2001.

[42] T. Soni Madhulatha. *An Overview on Clustering Methods. CoRR*, abs/1205.1117, 2012.

[43] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever και Ruslan Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors. CoRR*, abs/1207.0580, 2012.

[44] Varun Chandola, Arindam Banerjee και Vipin Kumar. *Anomaly Detection: A Survey. ACM Comput. Surv.*, 41(3), 2009.

[45] Connor W. Coley, William H. Green και Klavs F. Jensen. *Machine Learning in Computer-Aided Synthesis Planning. Accounts of Chemical Research*, 51(5):1281–1289, 2018. PMID: 29715002.

[46] Mehryar Mohri, Afshin Rostamizadeh και Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[47] D.H. Wolpert και W.G. Macready. *No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[48] Sebastian Ruder. *An overview of gradient descent optimization algorithms. CoRR*, abs/1609.04747, 2016.

[49] Yann Lecun, Leon Bottou, Y. Bengio και Patrick Haffner. *Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE*, 86:2278 – 2324, 1998.

[50] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning internal representations by error propagation*. 1986.

[51] Sepp Hochreiter και Jürgen Schmidhuber. *Long Short-Term Memory. Neural Computation*, 9(8):1735–1780, 1997.

[52] Ilya Sutskever, Oriol Vinyals και Quoc V. Le. *Sequence to Sequence Learning with Neural Networks. CoRR*, abs/1409.3215, 2014.

[53] Alex Graves, Greg Wayne και Ivo Danihelka. *Neural Turing Machines. CoRR*, abs/1410.5401, 2014.

[54] Thang Luong, Hieu Pham και Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, σελίδες 1412–1421, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[55] Jianpeng Cheng, Li Dong και Mirella Lapata. *Long Short-Term Memory-Networks for Machine Reading. CoRR*, abs/1601.06733, 2016.

[56] Aston Zhang, Zachary C. Lipton, Mu Li και Alexander J. Smola. *Dive into Deep Learning. arXiv preprint arXiv:2106.11342*, 2021.

[57] Boqing Gong, Wei Lun Chao, Kristen Grauman και Fei Sha. *Diverse Sequential Subset Selection for Supervised Video Summarization. Advances in Neural Information Processing Systems*Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence και K.Q. Weinberger, επιμελητές, τόμος 27. Curran Associates, Inc., 2014.

[58] Jeho Nam και Ahmed H. Tewfik. *Dynamic Video Summarization and Visualization. Proceedings of the Seventh ACM International Conference on Multimedia (Part 2)*, MULTIMEDIA '99, σελίδα 53–56, New York, NY, USA, 1999. Association for Computing Machinery.

[59] Vivekraj V. K., Debashis Sen και Balasubramanian Raman. *Video Skimming: Taxonomy and Comprehensive Survey. ACM Comput. Surv.*, 52(5), 2019.

[60] Michael S. Lew, Nicu Sebe, Chabane Djeraba και Ramesh Jain. *Content-Based Multimedia Information Retrieval: State of the Art and Challenges. ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.

[61] Ying Li, Tong Zhang και Daniel Tretter. *An overview of video abstraction techniques.* 2001.

[62] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso και Paolo Remagnino. *Summarizing Videos with Attention. CoRR*, abs/1812.01969, 2018.

[63] Ke Zhang, Wei Lun Chao, Fei Sha και Kristen Grauman. *Video Summarization with Long Short-Term Memory. Computer Vision – ECCV 2016*Bastian Leibe, Jiri Matas, Nicu Sebe και Max Welling, επιμελητές, σελίδες 766–782, Cham, 2016. Springer International Publishing.

[64] Shamit Lal, Shivam Duggal και Indu Sreedevi. *Online Video Summarization: Predicting Future to Better Summarize Present. 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, σελίδες 471–480, 2019.

[65] Sandra Eliza Fontes de Avila, Ana Paula Brandão Lopes, Antonio da Luz και Arnaldo de Albuquerque Araújo. *VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method. Pattern Recognition Letters*, 32(1):56–68, 2011. Image Processing, Computer Vision and Pattern Recognition in Latin America.

[66] Hussain Kanafani, Junaid Ahmed Ghauri, Sherzod Hakimov και Ralph Ewerth. *Unsupervised Video Summarization via Multi-source Features. CoRR*, abs/2105.12532, 2021.

[67] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I. Metsai, Vasileios Mezaris και Ioannis Patras. *AC-SUM-GAN: Connecting Actor-Critic and Generative Adversarial Networks for Unsupervised Video Summarization. IEEE Transactions on Circuits and Systems for Video Technology*, 31(8):3278–3292, 2021.

[68] Yujia Zhang, Xiaodan Liang, Dingwen Zhang, Min Tan και Eric P. Xing. *Unsupervised Object-Level Video Summarization with Online Motion Auto-Encoder. Pattern Recogn. Lett.*, 130(C):376–385, 2020.

[69] Kaiyang Zhou, Yu Qiao και Tao Xiang. *Deep Reinforcement Learning for Unsupervised Video Summarization With Diversity-Representativeness Reward. Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[70] Gokhan Yaliniz και Nazli Ikizler-Cinbis. *Using Independently Recurrent Networks for Reinforcement Learning Based Unsupervised Video Summarization. Multimedia Tools Appl.*, 80(12):17827–17847, 2021.

[71] Bin Zhao, Xuelong Li και Xiaoqiang Lu. *Property-Constrained Dual Learning for Video Summarization. IEEE Transactions on Neural Networks and Learning Systems*, PP:1–12, 2019.

[72] Jie Lei, Qiao Luan, Xinhui Song, Xiao Liu, Dapeng Tao και Mingli Song. *Action Parsing-Driven Video Summarization Based on Reinforcement Learning. IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):2126–2137, 2019.

[73] Hafiz Burhan Haq, M Asif και Maaz Bin. *Video Summarization Techniques: A Review. International Journal of Scientific Technology Research*, 9:146–153, 2021.

[74] Xinhui Song, Li Sun, Jie Lei, Dapeng Tao, Guanhong Yuan και Mingli Song. *Event-based large scale surveillance video summarization. Neurocomputing*, 187:66–74, 2016. Recent Developments on Deep Big Vision.

[75] Alexei A. Efros, Alexander C. Berg, Greg Mori και Jitendra Malik. *Recognizing Action at a Distance. IEEE International Conference on Computer Vision*, σελίδες 726–733, Nice, France, 2003.

[76] Yan Ke, Rahul Sukthankar και Martial Hebert. *Volumetric Features for Video Event Detection. Int. J. Comput. Vision*, 88(3):339–362, 2010.

[77] Shih Ting Lin, Yuan Hsin Liao, Yu Tsao και Shao Yi Chien. *Object-based on-line video summarization for internet of video things. 2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, σελίδες 1–4, 2017.

[78] Muhammad Asim, Noor Almaadeed, Somaya Al-maadeed, Ahmed Bouridane και Azeddine Beghdadi. *A Key Frame Based Video Summarization using Color Features. 2018 Colour and Visual Computing Symposium (CVCS)*, σελίδες 1–6, 2018.

[79] Chong Wah Ngo, Yu Fei Ma και Hong Jiang Zhang. *Video summarization and scene detection by graph modeling. IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):296–305, 2005.

[80] J. MacQueen. *Some methods for classification and analysis of multivariate observations.* Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[81] Jesna Mohan, Madhu S. Nair, Sabu M. Thampi και El Sayed M. El-Alfy. *Domain Independent Static Video Summarization Using Sparse Autoencoders and K-Means Clustering. J. Intell. Fuzzy Syst.*, 36(3):1945–1955, 2019.

[82] Marcos Vinicius Mussel Cirne και Helio Pedrini. *A Video Summarization Method Based on Spectral Clustering. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*José Ruiz-Shulcloper και GabriellaSanniti di Baja, επιμελητές, σελίδες 479–486, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[83] Vasileios Chasanis, Aristidis Likas και Nikolaos Galatsanos. *Video Rushes Summarization Using Spectral Clustering and Sequence Alignment. Proceedings of the 2nd ACM TRECVid Video Summarization Workshop*, TVS '08, σελίδα 75–79, New York, NY, USA, 2008. Association for Computing Machinery.

[84] Uros Damnjanovic, Ebroul Izquierdo και Marcin Grzegorzek. *Shot boundary detection using spectral clustering. 2007 15th European Signal Processing Conference*, σελίδες 1779–1783, 2007.

[85] Kenneth Kreutz-Delgado, Joseph F. Murray, Bhaskar D. Rao, Kjersti Engan, Te Won Lee και Terrence J. Sejnowski. *Dictionary Learning Algorithms for Sparse Representation. Neural Computation*, 15(2):349–396, 2003.

[86] Mingyang Ma, Shaohui Mei, Shuai Wan, Junhui Hou, Zhiyong Wang και David Dagan Feng. *Video summarization via block sparse dictionary selection. Neurocomputing*, 378:197–209, 2020.

[87] Jingjing Meng, Hongxing Wang, Junsong Yuan και Yap Peng Tan. *From Keyframes to Key Objects: Video Summarization by Representative Object Proposal Selection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, σελίδες 1039–1048, 2016.

[88] R. M. Jiang, A. H. Sadka και D. Crookes. *Advances in Video Summarization and Skimming*. Springer Berlin Heidelberg, 2009.

[89] M. Sanabria, F. Precioso και T. Menguy. *Hierarchical Multimodal Attention for Deep Video Summarization. 2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[90] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville και Yoshua Bengio. *Generative Adversarial Nets. Advances in Neural Information Processing Systems*Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence και K.Q. Weinberger, επιμελητές, τόμος 27. Curran Associates, Inc., 2014.

[91] Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo και In So Kweon. *Discriminative Feature Learning for Unsupervised Video Summarization. CoRR*, abs/1811.09791, 2018.

[92] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby και Ole Winther. *Autoencoding beyond pixels using a learned similarity metric. CoRR*, abs/1512.09300, 2015.

[93] Danila Potapov, Matthijs Douze, Zaïd Harchaoui και Cordelia Schmid. *Category-Specific Video Summarization. ECCV*, 2014.

[94] Hans Kellerer, Ulrich Pferschy και David Pisinger. *The Bounded Knapsack Problem*, σελίδες 185–209. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[95] Ke Zhang, Wei-Lun Chao, Fei Sha και Kristen Grauman. *Video Summarization with Long Short-term Memory. CoRR*, abs/1605.08110, 2016.

[96] Petros Koutras και Petros Maragos. *SUSiNet: See, Understand and Summarize it. CoRR*, abs/1812.00722, 2018.

[97] Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li και Li Fei-Fei. *Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition*, σελίδες 248–255. Ieee, 2009.

[98] Suramya Tomar. *Converting video formats with FFmpeg. Linux Journal*, 2006(146):10, 2006.

[99] G. Bradski. *The OpenCV Library. Dr. Dobb's Journal of Software Tools*, 2000.

[100] Andrew Collette. *Python and HDF5*. O'Reilly, 2013.