

# Κεφάλαιο 1

## Tube Proposal Network

### 1.1 Η αρχιτεκτονική του μοντέλου μας

Σε αυτό το κεφάλαιο, ασχολούμαστε κυρίως με το Tube Proposal Network (TPN), ένα από τα βασικά στοιχεία του μοντέλου μας ActionNet. Πριν ξεκινήσουμε την περιγραφή του, παρουσιάζουμε τη δομή όλου του μοντέλου μας. Προτείνουμε ένα δίκτυο παρόμοιο με αυτό των Hou, Chen, and Shah 2017. Η αρχιτεκτονική μας αποτελείται από τα ακόλουθα βασικά στοιχεία:

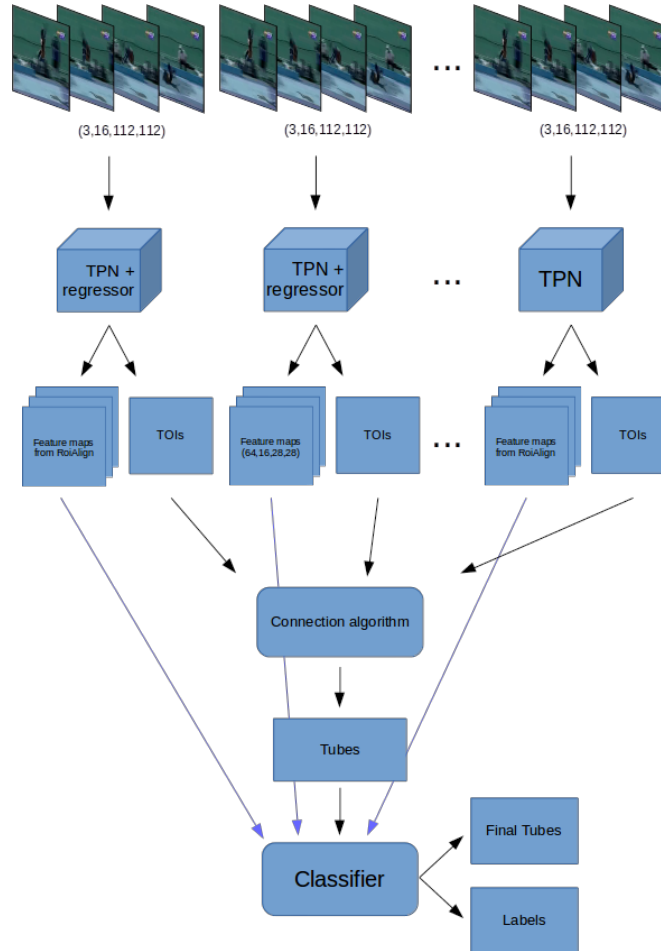
- Ένα τρισδιάστατο συνελικτικό μοντέλο (3D Convolutional Network), το οποίο χρησιμοποιείται για την εξαγωγή χαρακτηριστικών. Στην υλοποίησή μας χρησιμοποιούμε ένα δίκτυο 3D ResNet34, το οποίο λαμβάνεται από τους Hara, Kataoka, and Satoh 2018 και βασίζεται στα ResNet CNNs για ταξινόμηση εικόνων (He et al. 2015).
- Ένα TPN για την εξαγωγή υποψήφιων ToIs (βασιζόμενοι στην ιδέα που παρουσιάζουν οι Hou, Chen, and Shah 2017).
- Έναν ταξινομητή για την εύρεση της κλάσης των προτεινόμενων action video tubes.

Η βασική διαδικασία που ακολουθεί το ActionNet είναι:

1. Δεδομένου ενός βίντεο, το διαχωρίζουμε σε τμήματα βίντεο. Αυτά τα τμήματα βίντεο σε ορισμένες προσεγγίσεις επικαλύπτονται χρονικά και σε άλλες όχι.
2. Για κάθε τμήμα βίντεο, μετά την εκτέλεση της αλλαγής μεγέθους χωροχρονικά, τροφοδοούμε τα καρέ του στο ResNet34 για να εξάγουμε τους χωροχρονικούς χάρτες του. Αυτοί οι χάρτες ενεργοποίησης, στη συνέχεια, τροφοδοτούνται στο TPN για την πρόταση ακολουθιών πλαισίων που πιθανόν περιέχουν κάποια δράση τις θα ονομάσουμε Tubes of Interest (ToIs), όπως κάνουν και οι Hou, Chen, and Shah 2017.

3. Μετά την πρόταση ToIs για κάθε τμήμα βίντεο, χρησιμοποιώντας έναν αλγόριθμο σύνδεσης, το ActionNet βρίσκει τα τελικά υποψήφια action tubes. Αυτά τα action tubes δίνονται ως είσοδο σε έναν ταξινομητή για τον προσδιορισμό της κλάσης τους.

Ένα διάγραμμα του μοντέλου μας ActionNet εμφανίζεται στην εικόνα 1.1.



Σχήμα 1.1: Στρυςτυρε οφ τηε ωηολε νετωορκ

## 1.2 Εισαγωγή στο TPN

Ο κύριος σκοπός του TPN είναι να προτείνει **Tubes of Interest** (TOIs). Αυτά τα tubes είναι πιθανό να περιέχουν μια γνωστή δράση και αποτελούνται από μερικά δυσδιάστατα πλαίσια (1 για κάθε καρέ βίντεο). Το TPN είναι εμπνευσμένο

από το RPN που εισήχθη από το FasterRCNN (Ren et al. 2015), αλλά αντί για εικόνες, το TPN χρησιμοποιείται σε βίντεο όπως κάνουν και οι Hou, Chen, and Shah 2017. Σε πλήρη αντιστοιχία με το RPN, η δομή του TPN είναι παρόμοια με αυτή του RPN. Η μόνη διαφορά, είναι ότι το TPN χρησιμοποιεί 3D Convolutional Layers και 3D anchors αντί 2D.

Σχεδιάσαμε 2 κύριες δομές για το TPN. Κάθε προσέγγιση έχει διαφορετικό ορισμό για τα χρησιμοποιούμενα τρισδιάστατα anchors. Η υπόλοιπη δομή του TPN είναι κυρίως η ίδια με ορισμένες μικρές διαφορές στο στάδιο του regression.

## 1.3 Προετοιμασία πριν το TPN

### 1.3.1 Η προετοιμασία των δεδομένων

Πριν εισαχθεί ένα βίντεο στο ResNet και στο TPN για να εξαγάγουμε τα χαρακτηριστικά του και πιθανά ToIs, αυτό το βίντεο πρέπει να προεπεξεργαστεί. Η διαδικασία προεπεξεργασίας είναι η ίδια και για τις δύο προσεγγίσεις του TPN. Η αρχιτεκτονική μας λαμβάνει ως είσοδο μια ακολουθία από σταθερό αριθμό καρέ που έχουν σταθερό πλάτος και ύψος. Ωστόσο, κάθε βίντεο είναι πιθανόν να έχει διαφορετική ανάλυση. Αυτό δημιουργεί την ανάγκη να αλλάξουμε το μέγεθος κάθε καρέ και πλαισίου πριν εισαχθεί στο ActionNet. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, το πρώτο στοιχείο του δικτύου μας είναι ένα 3D ResNet που υλοποιήθηκε από τους Hara, Kataoka, and Satoh 2018. Αυτό το δίκτυο έχει σχεδιαστεί να δέχεται βίντεο με διαστάσεις (112, 112). Ως αποτέλεσμα, μεταβάλλουμε το μέγεθος κάθε καρέ από τα βίντεο των dataset σε (112, 112). Για να διατηρήσουμε την αναλογία διαστάσεων, προσθέτουμε μηδενικές τιμές είτε αριστερά και δεξιά, είτε πάνω και κάτω, ανάλογα με το ποια διάσταση είναι μεγαλύτερη. Στο σχήμα 1.2 μπορούμε να δούμε το αρχικό καρέ καθώς και το αναδιαμορφωμένο. Σε πλήρη αντιστοιχία, αλλάζουμε και το μέγεθος των πραγματικών πλαισίων οριοθέτησης αληθείας για κάθε καρέ (Τα σχήματα 1.2b και 1.2d το απεικονίζουν).

### 1.3.2 3D ResNet

Πριν από τη χρήση του Tube Proposal Network, εξάγουμε χωροχρονικά χαρακτηριστικά από το βίντεο. Για να γίνει αυτό, εξάγουμε τα 3 πρώτα στρώματα ενός προεκπαιδευμένου 3D ResNet34. Αυτό το μοντέλο είναι προεκπαιδευμένο στο Kinetics dataset (Kay et al. 2017) για διάρκεια του δείγματος ίση με 16 καρέ και μέγεθος δείγματος ίσο με (112, 122).

Αυτό το δίκτυο συνήθως χρησιμοποιείται για την ταξινόμηση ολόκληρου του βίντεο, οπότε μερικά από τα στρώματα του χρησιμοποιούν temporal stride ίσο με 2. Εμείς, όμως, θέτουμε το temporal stride ίσο με 1 γιατί δεν θέλουμε να χάσουμε χρονικές πληροφορίες κατά τη διάρκεια της διαδικασίας. Έτσι, η έξοδος του τρίτου στρώματος είναι ένα χάρτης χαρακτηριστικών με διαστάσεις (256, 16, 7, 7). Τροφοδοτούμε αυτό τον χάρτη χαρακτηριστικών στο TPN, το οποίο περιγράφεται στις ακόλουθες ενότητες.

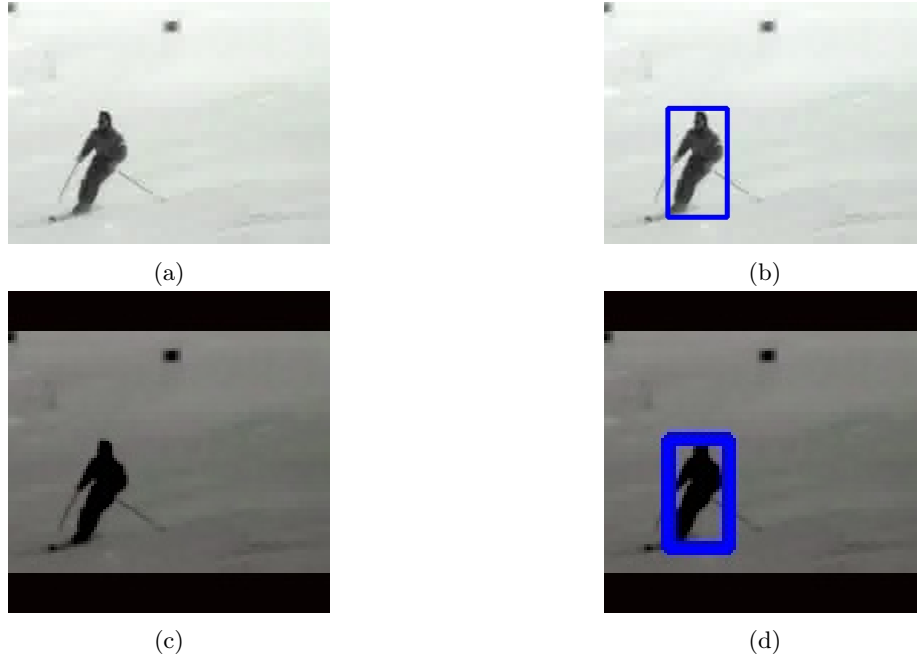


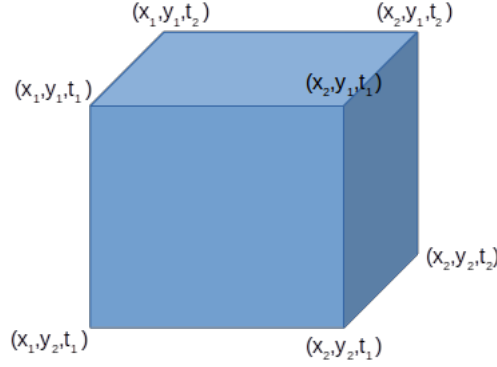
Figure 1.2: At (a), (b) frame is its original size and at (c), (d) same frame after preprocessing part

## 1.4 Τα τριασδιάστατα anchors ως 6-dim διανύσματα

### 1.4.1 Πρώτη περιγραφή

Ξεκινήσαμε να σχεδιάζουμε το TPN εμπνευσμένοι από την δουλειά των Hou, Chen, and Shah 2017. Έτσι, θεωρούμε κάθε anchor ως ένα τριασδιάστατο πλαίσιο το οποίο γράφεται ως  $(x_1, y_1, t_1, x_2, y_2, t_2)$  όπου  $x_1, y_1, t_1$  είναι οι πάνω μπροστά αριστερές διαστάσεις του κύβου και  $x_2, y_2, t_2$  είναι οι κάτω πίσω δεξιά όπως φαίνεται και στην εικόνα 1.3.

Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι εκτός από τις διαστάσεις x-y, η διάσταση του χρόνου είναι μεταβαλλόμενη. Ως αποτέλεσμα, τα προτεινόμενα TOIs δεν έχουν καθορισμένη χρονική διάρκεια. Αυτό θα μας βοηθήσει να ασχοληθούμε με τα μη-κομμένα (untrimmed) βίντεο, επειδή τα προτεινόμενα TOIs θα μπορούν να εξαιρέσουν backgroundχαρέ. Για αυτήν την προσέγγιση, χρησιμοποιούμε  $n = 4K = 60$  anchorsγια κάθε pixelστους χάρτες ενεργοποίησης του TPN. Έχουμε k anchorsγια κάθε διαφορετική διάρκεια anchor(5 κλίμακες των 1, 2, 4, 8, 16, 3 aspect ratios 1:1, 1:2, 2:1και 4 διάρκειες 16, 12, 8, 4 χαρέ). Σύμφωνα με τους Hou, Chen, and Shah 2017, τα anchors του δικτύου ορίζονται σύμφωνα με τα πιο συνηθισμένα anchors του συνόλου δεδομένων. Αυτό, ωστόσο,

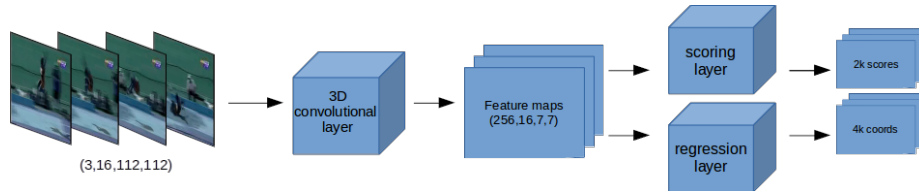


Σχήμα 1.3: An example of the anchor  $(x_1, y_1, t_1, x_2, y_2, t_2)$

δημιουργεί την ανάγκη επανασχεδιασμού του δικτύου για κάθε σύνολο δεδομένων. Στην προσέγγισή μας, χρησιμοποιούμε τα ίδια anchors και για τα δύο σύνολα δεδομένων, επειδή θέλουμε το δίκτυό μας να μην να βασίζεται στο σύνολο δεδομένων, αλλά να είναι σε θέση να γενικευσει για διάφορα σύνολα δεδομένων. Ως διάρκεια δειγματοληψίας, επιλέξαμε 16 καρέ ανά τμήμα βίντεο, επειδή η προ-εκπαιδευμένη έκδοση ResNet που χρησιμοποιούμε έχει εκπαιδευτεί για βίντεο κλιπ με αυτή τη διάρκεια. Έτσι, η δομή του TPN είναι:

- 1 3D Convolutional Layer με kernel size = 3, stride = 3 και padding = 1
- 1 classification layer που εξάγει  $2n$  scores για το αν υπάρχει ή όχι δράση για  $n$  tubes.
- 1 regression layer που εξάγει  $6n$  διαστάσεις  $(x_1, y_1, t_1, x_2, y_2, t_2)$  για  $n$  tubes.

Η δομή του TPN παρουσιάζεται στην Εικόνα 1.4. Το αποτέλεσμα του TPN είναι τα καλύτερα κουτιά, τα οποία είναι τα πιο πιθανά να περιέχουν κάποια δράση.

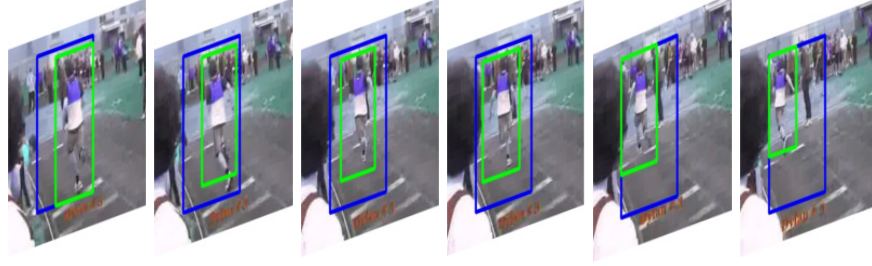


Σχήμα 1.4: Στρυςτρε οφ TPN

### 1.4.2 Training

Όπως προαναφέρθηκε, το TPN εξάγει ToIs ως 6-διάστατα διανύσματα. Για το λόγο αυτό, τροποποιήσαμε τα πραγματικά πλαίσια ανά καρέ σε πραγματικά Tubes .

Θεωρούμε δεδομένο ότι το άτομο που δρα, δεν μπορεί να κινηθεί πολύ σε 16 καρέ, γι' αυτό χρησιμοποιούμε τέτοιου είδους Tubes . Όπως φαίνεται στο σχήμα 1.5, αυτά τα τυβες είναι τρισδιάστατα κουτιά που περιλαμβάνουν όλα τα πραγματικά πλαίσια, τα οποία είναι διαφορετικά ανά καρέ.



Σχήμα 1.5: Το πραγματικό τυβε έχει μπλε χρώμα και το πραγματικό ανά καρέ πλαίσιο έχει χρώμα πράσινο

Για τη διαδικασία training, για κάθε βίντεο, επιλέγουμε τυχαία ένα μέρος του, το οποίο έχει διάρκεια 16 καρέ. Θεωρούμε ένα anchor ως πρώτο πλάνο, αν η βαθμολογία επικάλυψης του με το πραγματικό tube είναι μεγαλύτερη από 0.5 . Διαφορετικά, θεωρείται ως anchor φόντου. Χρησιμοποιούμε έναν scoring layer για να ταξινομήσουμε σωστά αυτά τα anchors και χρησιμοποιούμε την Cross Entropy Loss ως συνάρτηση κόστους (loss function). Έχουμε πολλά anchors για να προτείνουμε μια δράση, αλλά μικρό αριθμό δράσεων σε κάθε βίντεο, έτσι επιλέγουμε 256 anchors συνολικά για κάθε video. Ορίζουμε ότι ο μέγιστος αριθμός των anchors προσκηνίου να είναι 25% από τους 256 anchors και τα υπόλοιπα είναι anchors φόντου.

Η σωστή ταξινόμηση ενός anchor δεν είναι αρκετή για να προτείνουμε ToIs. Είναι, επίσης, απαραίτητο τα anchors να επικαλύπτονται όσο το δυνατόν περισσότερο με τα πραγματικά tubes. Αυτός είναι ο λόγος που χρησιμοποιούμε ένα επίπεδο παλινδρόμησης. Αυτό το layer «κινεί» τον κύβο στην περιοχή που πιστεύεται ότι είναι πιο κοντά στη δράση. Για συνάρτηση κόστους παλινδρόμησης χρησιμοποιούμε την συνάρτηση κόστους smooth-L1 όπως παρουσιάζεται από τους Girshick et al. 2013. Για να υπολογίσουμε τους στόχους παλινδρόμησης, χρησιμοποιούμε την pytorch εφαρμογή του FasterRCNN (Yang et al. 2017) για την παλινδρόμηση του πλαισίου και τροποποιούμε τον κώδικα επεκτείνοντας τον για 3 διαστάσεις. Έτσι έχουμε:

$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a, & t_z &= (z - z_a)/d_a, \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a), & t_d &= \log(d/d_a), \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, & t_z^* &= (z^* - z_a)/d_a, \\ t_w^* &= \log(w^*/w_a), & t_h^* &= \log(h^*/h_a), & t_d^* &= \log(d^*/d_a), \end{aligned}$$

όπου τα  $x, y, z, w, h, d$  υποδεικνύουν τις συντεταγμένες του κέντρου του τρισδιάστατου κουτιού καθώς επίσης το πλάτος, το ύψος και τη διάρκειά του. Οι μεταβλητές  $x, x_a$ , και  $x^*$  αφορούν το προβλεπόμενο πλαίσιο, το πλαίσιο του anchor

και το πραγματικό πλαίσιο αντίστοιχα (ομοίως για  $y, z, w, h, d$ ). Φυσικά, υπολογίζουμε την απώλεια παλινδρόμησης μόνο για τα anchors προσκηνίου και όχι αυτά του φόντου, συνεπώς στην χειρότερη θα υπολογίσουμε 64 στόχους για κάθε video.

Για να συνοψίσουμε στη διαδικασία training, εκπαιδεύουμε 2 layers για το TPN, scoring και regression. Η συνάρτηση κόστους περιλαμβάνει τα training losses που προκύπτουν απ' αυτά τα layers και ο τύπος της είναι:

$$L = \sum_i L_{cls}(p_i, p_i^*) + \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

όπου:

- $L_{cls}$  είναι η Cross Entropy loss που χρησιμοποιούμε για να εκπαιδεύσουμε τα anchors, με  $p_i$  είναι η προβλεπόμενη κλάση,  $p_i^*$  είναι η πραγματική κλάση και  $p_i, p_i^* \in \{0, 1\}$ .
- $L_{reg}$  είναι η συνάρτηση κόστους smooth-L1, η οποία πολλαπλασιάζεται με  $p_i^*$  προκειμένου να ενεργοποιείται όταν υπάρχει θετικό anchor ( $p_i^* = 1$ ) και να απενεργοποιείται για τα background anchors ( $p_i^* = 0$ ).

### 1.4.3 Validation

Η διαδικασία validation είναι κάπως παρόμοια με τη διαδικασία του training. Επιλέγουμε τυχαία 16 καρέ από ένα βίντεο επικύρωσης και εξετάζουμε αν υπάρχει τουλάχιστον 1 προτεινόμενο ToI που επικαλύπτει  $\geq 0,5$  για κάθε πραγματικό tube και παίρνουμε το recall score. Για να λάβουμε καλές προτάσεις, μετά τη λήψη των classification scores και των regression targets από το αντίστοιχα layers, χρησιμοποιούμε τον αλγόριθμο Non-Maximum Suppression(NMS). Έχουμε ορίσει το κατώφλι του NMS ίσο με 0,7 και κρατάμε τους πρώτους 150 κύβους με τη μεγαλύτερη βαθμολογία.

### 1.4.4 Modified Intersection over Union(mIoU)

Κατά τη διάρκεια του training, έχουμε πολλά anchors. Πρέπει να τα ταξινομήσουμε ως anchors προσκηνίου ή anchors παρασκηνίου. Τα anchors προσκηνίου είναι εκείνα που περιέχουν κάποια ενέργεια και, αντίστοιχα, του φόντου που δεν έχουν. Όπως παρουσιάστηκε προηγουμένως, το IoU για τα cuboids υπολογίζει το ποσοστό μεταξύ του όγκου της επικάλυψης και του όγκου των Ένωσης. Διαπισθητικά, αυτό το κριτήριο είναι καλό για την αξιολόγηση του βαθμού επικάλυψης 2 tube, αλλά έχει ένα μεγάλο μειονέκτημα: Θεωρεί ότι οι διαστάσεις x-y έχουν την ίδια σημασία με τη χρονική διάσταση, το οποίο δεν επιθυμούμε. Κι αυτό διότι πρώτον, μας ενδιαφέρει να είμαστε ακριβείς στη χρονική διάσταση, και στη συνέχεια μπορούμε να διορθώσουμε τον τομέα x-y. Ως αποτέλεσμα, αλλάζουμε τον τρόπο με τον οποίο υπολογίζουμε το Intersection over Union. Υπολογίζουμε ξεχωριστά το IoU στις διαστάσεις x-y (IoU-xy) και στην t διάσταση (IoU-t). Τέλος,

πολλαπλασιάζουμε αυτά τα δύο score για να πάρουμε το τελικό IoU. Συνεπώς ο τύπος για 2 tubes  $(x_1, y_1, t_1, x_2, y_2, t_2)$  και  $(x'_1, y'_1, t'_1, x'_2, y'_2, t'_2)$  είναι:

$$IoU_{xy} = \frac{\text{Area of Overlap in x-y}}{\text{Area of Union in x-y}}$$

$$IoU_t = \frac{\max(t_1, t'_1) - \min(t_2, t'_2)}{\min(t_1, t'_1) - \max(t_2, t'_2)}$$

$$IoU = IoU_{xy} \cdot IoU_t$$

Το παραπάνω κριτήριο μας βοηθά να εξισορροπήσουμε τις επιπτώσεις του χρόνου στο IoU score. Για παράδειγμα, ας εξετάσουμε 2 anchors:  $a = (22, 41, 1, 34, 70, 5)$  και  $b = (20, 45, 2, 32, 72, 5)$ . Αυτά τα 2 anchor στις διαστάσεις x-y έχουν βαθμολογία IoU ίσο με 0,61. Αλλά δεν είναι ακριβώς επικαλυπτόμενα στην διάσταση του χρόνου. Χρησιμοποιώντας την πρώτη προσέγγιση έχουμε 0,5057 IoU βαθμολογία ενώ η δεύτερη προσέγγιση μας δίνει 0,4889. Έτσι, το δεύτερο κριτήριο θα απέρριπτε αυτό το anchor, διότι υπάρχει μια διαφορά στην χρονική διάρκεια.

Για να επιβεβαιώσουμε την ιδέα μας, εκπαιδεύουμε το TPN χρησιμοποιώντας τόσο το IoU κριτήριο όσο και το mIoU για την επικάλυψη των tubes. Στο πίνακα 1.1 μπορούμε να δούμε την απόδοση σε κάθε περίπτωση και για τα δύο σύνολα δεδομένων, JHMDB και UCF. Το recall όριο για αυτή την περίπτωση είναι 0,5 και κατά την διάρκεια του validation χρησιμοποιούμε το κανονικό IoU για να καθορίσουμε αν 2 tubes επικαλύπτονται. Ο πίνακας 1.1 μας δείχνει ότι το

Dataset	Criterion	Recall(0.5)
JHMDB	IoU	0.70525
	mIoU	0.7052
UCF	IoU	0.4665
	mIoU	0.4829

Table 1.1: Recall results for both datasets using IoU and mIoU metrics

τροποποιημένο-IoU μας δίνει ελαφρώς καλύτερη απόδοση recall μόνο στο σύνολο δεδομένων UCF. Αυτό είναι λογικό, επειδή το σύνολο δεδομένων JHMDB χρησιμοποιεί κομμένα βίντεο, συνεπώς η χρονική διάρκεια δεν επηρεάζει πολύ. Έτσι, από τώρα και στο εξής, κατά τη διάρκεια του training χρησιμοποιούμε το mIoU ως επικαλυπτόμενη πολιτική βαθμολογίας.

#### 1.4.5 Βελτιώνοντας το TPN score

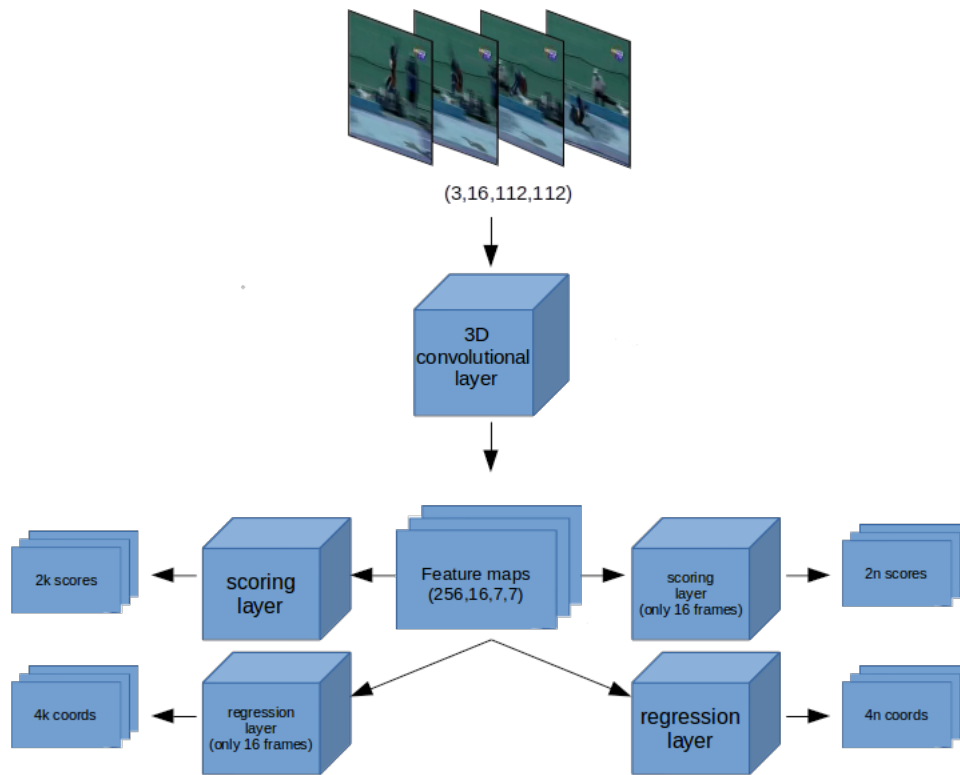
Μετά την πρώτη δοκιμή, μας ήρθε η ιδέα ότι σε ένα βίντεο που διαρκεί 16 καρέ, στον τομέα του χρόνου, όλα τα είδη των ενεργειών μπορούν να χωρίζονται στις ακόλουθες κατηγορίες:

1. Η ενέργεια ξεκινά από το  $n$ -ο πλαίσιο και ολοκληρώνεται μετά το 16ο καρέ του βίντεο που έχει υποβληθεί σε δειγματοληψία.



2. Η ενέργεια έχει ήδη ξεκινήσει πριν από το 1ο καρέ του βίντεο και τελειώνει στο  $n$  πλαίσιο.
3. Η ενέργεια έχει ήδη ξεκινήσει πριν από το 1ο καρέ του βίντεο και ολοκληρώνεται μετά το 16ο καρέ βίντεο.
4. Η ενέργεια ξεκινά και τελειώνει σε αυτά τα 16 καρέ του βίντεο.

Επιπλέον, παρατηρήσαμε ότι οι περισσότερες ενέργειες, στα σύνολα δεδομένων μας, διαρκούν περισσότερο από 16 καρέ. Έτσι, ήρθαμε με την ιδέα να προσθέσουμε 1 scoring layer και 1 regression Layer που θα προτείνει ToIs με σταθερή διάρκεια ίση με τη διάρκεια του δείγματος (16 καρέ) και θα λάβει υπόψη τις χωρικές πληροφορίες που παράγονται από τους χάρτες ενεργοποίησης. Η νέα δομή του TPN εμφανίζεται στην εικόνα 1.6. Αφού λάβουμε τις προτάσεις και από και από τα δύο scoring layers, τις ενώνουμε με ποσοστό 1:1 μεταξύ των ToI που εξαχθήκαν από τα δύο υποδίκτυα. Στόχος μας είναι να «συμπίεσουμε» τους χάρτες της χρονι-



Σχήμα 1.6: TPN structure after adding 2 new layers, where  $k = 5n$ .

κής διάστασης, προκειμένου να προτείνουμε ToIs σύμφωνα μόνο με τις χωρικές πληροφορίες. Έτσι, βρήκαμε με 2 τεχνικές για να πραγματοποιήσουμε κάτι τέτοιο:

1. Να χρησιμοποιήσουμε 3D Convolutional Layers με μέγεθος πυρήνα = (διάρκεια δείγματος, 1, 1), stride = 1 και χωρίς padding για scoring και regression. Αυτός ο πυρήνας «κοιτάει» μόνο στη χρονική διάσταση των χαρτών ενεργοποίησης και δεν θεωρεί καμία χωρική εξάρτηση.
2. Να αποκτήσουμε τις μέσες τιμές από τη χρονική διάσταση και στη συνέχεια να χρησιμοποιήσουμε ένα 2D Convolutional Layer για scoring και regression.

Οι διαδικασίες training και validation παραμένουν ίδιες. Η μόνη μεγάλη διαφορά είναι ότι τώρα έχουμε κόστη από 2 διαφορεά σύστημα που προτείνουν TOIs. Έτσι, κατά την διάρκεια του validation, εμείς αρχικά, ενώνουμε τα προτεινόμενα ToIs και, στη συνέχεια, ακολουθούμε την ίδια διαδικασία, η οποία είναι να υπολογίσουμε το recall. Για το training loss, έχουμε 2 διαφορετικές Cross-entropy loss συναρτήσεις και 2 διαφορετικά smooth-L1 losses. Έτσι, η απώλεια εκπαίδευσης, τώρα, ορίζεται ως:

$$L = \sum_i L_{cls}(p_i, p_i^*) + \sum_i L_{cls}(p_{fixed,i}, p_{fixed,i}^*) + \sum_i p_i^* L_{reg}(t_i, t_i^*) + \sum_i p_{fixed,i}^* L_{reg}(t_{fixed,i}, t_{fixed,i}^*) \quad (1.1)$$

όπου:

- $L_{cls}$  είναι η Cross Entropy loss που χρησιμοποιούμε για να εκπαιδεύσουμε τα anchors, με  $p_i$  είναι η προβλεπόμενη κλάση,  $p_i^*$  είναι η πραγματική κλάση και  $p_i, p_i^* \in \{0, 1\}$
- $L_{reg}$  είναι η συνάρτηση κόστους smooth-L1, η οποία πολλαπλασιάζεται με  $p_i^*$  προκειμένου να ενεργοποιείται όταν υπάρχει θετικό anchor ( $p_i^* = 1$ ) και να απενεργοποιείται για τα background anchors ( $p_i^* = 0$ ).
- $p_i$  είναι τα anchors από τα scoring και regression layers με μεταβλητή χρονική διάρκεια και  $p_i^*$  είναι η αντίστοιχη πραγματική τους κλάση.
- $p_{fixed,i}$  είναι τα anchors από τα scoring και regression layers με σταθερή χρονική διάρκεια ίση με 16 καρέ και  $p_{fixed,i}^*$  είναι η αντίστοιχη πραγματική του κλάση.

Εκπαιδεύουμε το δίκτυο TPN χρησιμοποιώντας και τις δύο τεχνικές και η απόδοση του recall εμφανίζεται στον πίνακα 1.2.

Όπως μπορούμε να δούμε από τα προηγούμενα αποτελέσματα, τα νέα layers αύξησαν σημαντικά την απόδοση του recall. Πέρα από αυτό, ο πίνακας 1.2 δείχνει ότι η λήψη των μέσων τιμών από τη χρονική διάσταση μας δίνει τα καλύτερα αποτελέσματα.

Dataset	Fix-time anchors	Type	Recall(0.5)
JHMDB	No	-	0.7052
	Yes	Kernel	0.6978
		Mean	0.7463
UCF	No	-	0.4829
	Yes	Kernel	0.4716
		Mean	0.4885

Table 1.2: Recall results after adding fixed time duration anchors

#### 1.4.6 Προσθήκη Regressor

Το αποτέλεσμα του TPN είναι τα *alpha*-υψηλότερα βαθμολογικά anchors που μετακινήθηκαν σύμφωνα με την regression πρόβλεψη τους. Μετά από αυτό, πρέπει να μετατρέψουμε τα προτεινόμενα anchors σε ToIs. Για να γίνει αυτό, προσθέτουμε ένα σύστημα παλινδρόμησης που παίρνει ως εισόδο τους χάρτες χαρακτηριστικών των τρισδιάστατων κουτιών και επιστρέφει μια ακολουθία από δυσδιάστατα κουτιά, ένα για κάθε καρέ. Το μόνο πρόβλημα είναι ότι η παλινδρόμηση χρειάζεται ως είσοδο χάρτες χαρακτηριστικών σταθερού μεγέθους. Αυτό το πρόβλημα είναι ήδη λυμένο από τα R-CNNs που χρησιμοποιούν ROI pooling και ROI align προκειμένου να πάρουμε του σταθερό μέγεθος χάρτες ενεργοποίησης από ROIs μεταβαλλόμενα μεγέθους. Στην περίπτωση μας, επεκτείνουμε την λειτουργία RoI Align, που παρουσιάζεται από το MaskR-CNN, και εμείς το ονομάζουμε **3D ROI align**.

**3D Roi Align** Το 3D ROI align είναι μια τροποποίηση του ROI Align που παρουσιάστηκε από το MaskR-CNN. Η κύρια διαφορά μεταξύ αυτών των δύο είναι ότι το MaskR-CNN, στο RoiAlign, χρησιμοποιεί διγραμμική παρεμβολή για την εξαγωγή των χαρακτηριστικών των ROIs και το δικός μας 3D ROI Align χρησιμοποιεί τριγραμμική παρεμβολή για τον ίδιο λόγο. Και πάλι, η 3η διάσταση είναι χρόνος. Συνεπώς, έχουμε ως είσοδο ένα χάρτη χαρακτηριστικών που εξάγεται από το ResNet34 με διαστάσεις (64, 16, 28, 28) και έναν τένσορα που περιέχει τα προτεινόμενα ToIs. Για κάθε ToI του οποίου ο χάρτης ενεργοποίησης έχει μέγεθος ίσο με (64, 16, 7, 7), έχουμε ως έξοδο ένα χάρτη χαρακτηριστικών με μέγεθος (64, 16, 7, 7).

#### Regression procedure

Στην αρχή, για κάθε προτεινόμενο ToI, έχουμε τους αντίστοιχους χάρτες ενεργοποίησης χρησιμοποιώντας 3D ROI align. Αυτά τα χαρακτηριστικά δίνονται ως είσοδο σε έναν Regressor. Αυτός επιστρέφει  $16 \cdot 4$  προβλεπόμενες μεταβολές  $(\delta_x, \delta_y, \delta_w, \delta_h)$ , 4 για κάθε καρέ, όπου  $\delta_x, \delta_y$  καθορίζουν τις συντεταγμένες του κέντρου των προτάσεων και  $\delta_w, \delta_h$  το πλάτος και το ύψος του, όπως ορίζεται από τους Girshick et al. 2013. Κρατάμε μόνο τις προβλεπόμενες μεταβολές, για τα καρέ που  $\geq t_1$  και  $< t_2$  και για υπόλοιπα θέτουμε ένα μηδενικό 2D κουτί. Μετά

από αυτό, τροποποιούμε κάθε anchor, γραμμένο ως έναν κύβο δηλαδή γραμμένο ως  $(x_1, y_1, t_1, x_2, y_2, t_2)$  σε μια ακολουθία πλαισίων 2D, όπως:  
 $(0, 0, 0, 0, \dots, x_{T_1}, y_{T_1}, x'_{T_1}, y'_{T_1}, \dots, x_i, y_i, x'_i, \dots, x_{T_2}, y_{T_2}, x'_{T_2}, y'_{T_2}, 0, 0, 0, 0, \dots)$ ,  
όπου:

- $T_1 \leq i \leq T_2$ , για  $T_1 < t_1 + 1, T_2 < t_2$  και  $T_1, T_2 \in \mathbb{Z}$
- $x_i = x_1, y_i = y_1, x'_i = x_2, y'_i = y_2$ .

**Training** Για να εκπαιδεύσουμε τον Regressor μας, ακολουθούμε τα ίδια βήματα που ακολουθήσαμε προηγουμένως για την προηγούμενη διαδικασία εκπαίδευσης του TPN. Αυτό σημαίνει ότι επιλέγουμε τυχαία 16 ToIs από αυτές που προτείνονται από το scoring layer του TPN. Από αυτά, 4 είναι τα anchors προσκήνιου, το οποίο σημαίνει ότι αποτελούν το 25% του συνολικού αριθμού των anchors όπως συνέβη προηγουμένως. Εξάγουμε τα αντίστοιχα χαρακτηριστικά τους χρησιμοποιώντας 3D ROI Align και υπολογίζουμε τους στόχους τους, όπως κάναμε για το regression layer. Τροφοδοτούμε το δίκτυο μας με αυτά τα χαρακτηριστικά και συγκρίνουμε τους προβλεπόμενους στόχους με τους αναμενόμενους. Ξανά πάλι, χρησιμοποιούμε smooth-L1 loss function για τη συνάρτηση κόστους, υπολογίζοντας την μόνο για ToIs που είναι στο προσκήνιο. Έτσι, προσθέτουμε μια άλλη παράμετρο στο φόρμουλα απώλειας εκπαίδευσης που πλέον ορίζεται ως:

$$L = \sum_i L_{cls}(p_i, p_i^*) + \sum_i L_{cls}(p_{fixed,i}, p_{fixed,i}^*) + \sum_i p_i^* L_{reg}(t_i, t_i^*) + \sum_i p_{fixed,i}^* L_{reg}(t_{fixed,i}, t_{fixed,i}^*) + \sum_i q_i^* L_{reg}(c_i, c_i^*) + \quad (1.2)$$

όπου εκτός από τις παραμέτρους που καθορίστηκαν προηγουμένως, ορίζουμε  $c_i$  ως στόχους παλινδρόμησης για τα επιλεγμένα τυβες  $q_i$ . Αυτά τα tubes είναι που επιλέγονται τυχαία από τα προτεινόμενα ToIs και  $q_i^*$  είναι τα αντίστοιχοι πραγματικά tubes, οι οποίοι είναι τα πλησιέστερα σε κάθε  $q_i$  tube. Και πάλι χρησιμοποιούμε το  $q_i^*$  ως παράγοντα, επειδή θεωρούμε ένα tube ως φόντο όταν δεν επικαλύπτεται με οποιοδήποτε πραγματικό tube περισσότερο από 0,5.

**Validation** Χρησιμοποιούμε, ξανά, τη μετρική recall για να αξιολογήσουμε την απόδοση του παλινδρομητή. Υπολογίζουμε 3 επιδόσεις recall:

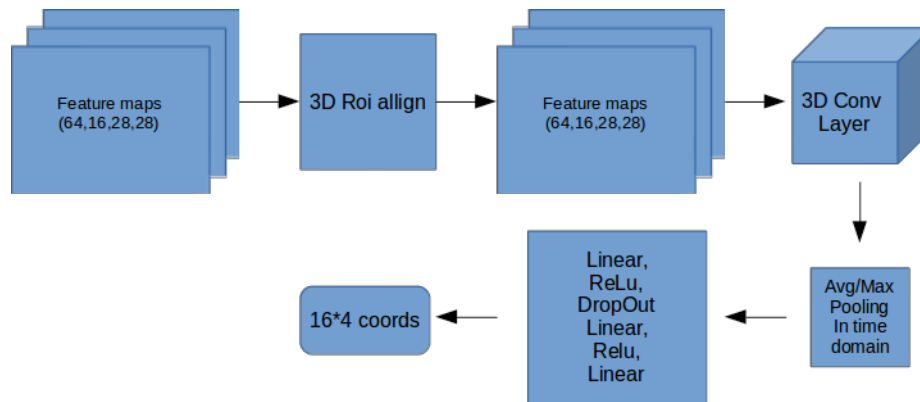
**Cuboid Recall**, που είναι η recall μετρική για τα προτεινόμενα τρισδιάστατα κουτιά. Ενδιαφερόμαστε για αυτήν την επίδοση γιατί, θέλουμε να μάθουμε πόσο καλές είναι οι προτάσεις μας πριν τις τροποποιήσουμε σε σειρές κουτιών.

**Single frame Recall**, η οποία είναι η επίδοση recall για τις προτεινόμενες ακολουθίες κουτιών σε σχέση με τις πραγματικές.

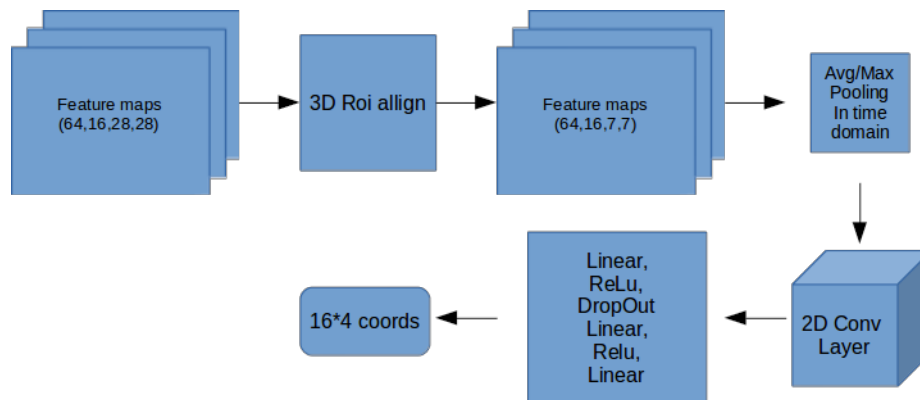
**Follow-up Single Frame Recall**, που είναι η απόδοση της ανάκλησης μόνο για τα τρισδιάστατα Κουτιά που ήταν πάνω από το όριο επικάλυψης μεταξύ των προτεινόμενων κυβών και των πραγματικών κύβων. Χρησιμοποιούμε αυτή τη μετρική για να γνωρίζουμε πόσα από τους προτεινόμενα τρισδιάστατα κουτιά κατέληξαν να είναι καλές προτάσεις.

### Αρχιτεκτονικές για τον Regressor

Σχεδιάσαμε 2 προσεγγίσεις για την υλοποίηση του Regressor. Αυτές απεικονίζονται στις Εικόνες 1.7 και 1.8, με την πρώτη προσέγγιση να αποτελείται από ένα 3D Convolutional Layer σε αντίθεση με την δεύτερη προσέγγιση που έχει ένα 2D Convolutional Layer.



Σχήμα 1.7: Στρυςτυρε οφ Ρεγρεσσορ



Σχήμα 1.8: Στρυςτυρε οφ Ρεγρεσσορ

Η διαδικασία που ακολουθούν και οι δύο Regressors περιγράφεται παρακάτω:

1. Αρχικά εξάγουμε τα αντίστοιχα feature maps για κάθε ToI χρησιμοποιώντας 3D Roi Align, και ακολούθως τα κανονικοποιούμε. Μετά, στην πρώτη προσέγγιση τροφοδοτούμε ένα 3D Convolutional Layer με kernel ίσο με 1, stride ίσο με 1 και χωρίς padding. Στην συνέχεια, εφαρμόζουμε μια pooling διαδικασία στην διάσταση του χρόνου (είτε avg είτε max pooling). Απ' την άλλη, στην δεύτερη προσέγγιση, πρώτα εφαρμόζουμε avg/max pooling στην διάσταση του χρόνου και στην συνέχεια τροφοδοτούμε ένα 2D Convolutional Layer με kernel = 1, stride = 1 και χωρίς padding.
2. Και στις δύο περιπτώσεις λαμβάνουμε ως έξοδο ένα feature map με διαστάσεις (64,7,7) το οποίο περνάμε διαδοχικά από 1 γραμμικό layer, 1 Relu Layer, 1 Dropout Layer, άλλο ένα γραμμικό Layer, άλλο ένα ReLu layer και ένα τελικό γραμμικό layer. Το τελευταίο layer μας δίνει ως έξοδο 64 στόχους,  $4 \cdot 16$  μετακινήσεις.

Dataset	Pooling	Cuboid	Singl. Fr.	Follow-up S.F.
JHMDB	avg	0.8545	0.7649	0.7183
	max	0.8396	0.7761	0.5783
UCF	avg	0.5319	0.4694	0.5754
	max	0.5190	0.5021	0.5972

Table 1.3: Recall results after convertying cuboids into sequences of frames

Dataset	Pooling	F. Map	Recall	Recall SR	Recall SRF
JHMDB	mean	64	0.6828	0.5112	0.7610
		128	0.8694	0.7799	0.6756
		256	0.8396	0.7687	0.7029
	max	64	0.8582	0.7985	0.5914
		128	0.8358	0.7724	0.8118
		256	0.8657	0.8022	0.7996
UCF	mean	64	0.5055	0.4286	0.5889
		128	0.5335	0.4894	0.5893
		256	0.5304	0.4990	0.6012
	max	64	0.5186	0.4990	0.5708
		128	0.5260	0.4693	0.5513
		256	0.5176	0.4878	0.6399

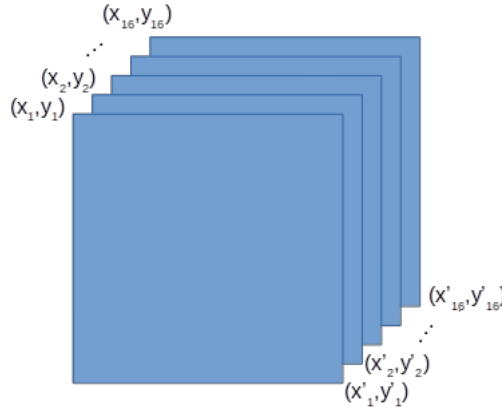
Table 1.4: Recall performance using 3 different feature maps as Regressor's input and 2 pooling methods

Οι πίνακες 1.3 και 1.4 περιέχουν τα αποτελέσματα για την πρώτη και δεύτερη προσέγγιση. Μάλιστα, για την δεύτερη προσέγγιση ελεγξαμε 3 διαφορετικούς χάρτες χαρακτηριστικών, ενώ και στις 2 περιπτώσεις πειραματιστήκαμε χρησιμοποιώντας και τις 2 προαναφερθέντες pooling μεθόδους για τα σύνολα δεδομένων JHMDB και UCF-101. Παρατηρούμε ότι, με βάση τα παραπάνω αποτελέσματα,

λαμβάνουμε σχετικά χαμηλή recall απόδοση για το dataset UCF ενώ για το JH-MDB τα αποτελέσματα είναι κάπως καλύτερα. Πιο συγκεκριμένα, με βάση την πρώτη προσέγγιση λαμβάνουμε τελικά recall απόδοση ίση με 76-77% για το JH-MDB και 46-50% για το UCF. Με βάση την δεύτερη προσέγγιση, στην καλύτερη περίπτωση λαμβάνουμε 80% απόδοση recall για το JH-MDB ενώ για το UCF παραμένουμε στα ίδια περίπου αποτελέσματα. Παράλληλα παρατηρούμε ότι χάνουμε περίπου 30-40% από τις καλές cuboid προτάσεις και στις 2 περιπτώσεις, το οποίο αποτελεί μεγάλο πρόβλημα και των δύο προσεγγίσεων. Όλ' αυτά μας κάνουν να ξανασκεφτούμε τον τρόπο που σχεδιάσαμε το TPN και μας οδήγησε στο να σχεδιάσουμε ένα νέο μοντέλο.

## 1.5 Τα τριασδιάστατα ανςηορς ως 4k διανύσματα

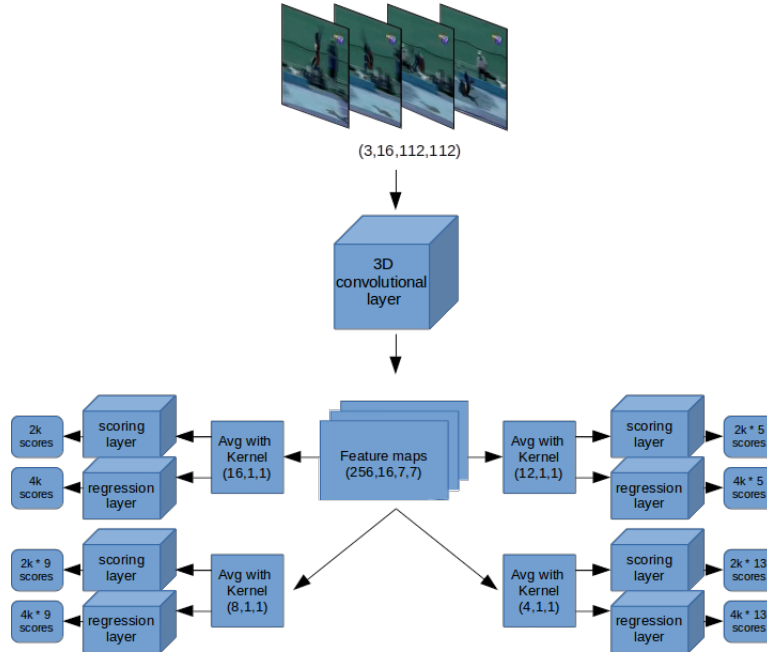
Σε αυτή την προσέγγιση, ορίζουμε τους 3D anchors ως διανύσματα με 4k συντεταγμένες ( $k = 16$  καρέ = διάρκεια δείγματος). Έτσι ένα τυπικό anchor γράφεται ως  $(x_1, y_1, x'_1, y'_1, x_2, y_2, \dots)$  όπου  $x_1, y_1, x'_1, y'_1$  είναι οι συντεταγμένες για 1ο καρέ,  $x_2, y_2, x'_2, y'_2$  για το 2ο καρέ κλπ, όπως παρουσιάστηκε από τους Γιρδηαρ κ.ά. 2017. Η εικόνα 1.9 απεικονίζει ένα τέτοιου τύπου anchor.



Σχήμα 1.9: Αν εξαπλε οφ τηρ ανςηορ  $(x_1, y_1, x'_1, y'_1, x_2, y_2, \dots)$

Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι δεν χρειάζεται να μεταφράσουμε τα 3D anchors σε 2D κουτιά, γεγονός που προκάλεσε πολλά προβλήματα στην προηγούμενη προσέγγιση. Ωστόσο, αυτή η προσέγγιση έχει ένα μεγάλο μειονέκτημα, το οποίο είναι το γεγονός ότι αυτός ο τύπος anchor έχει σταθερή χρονική διάρκεια. Για να αντιμετωπίσουμε αυτό το πρόβλημα, έχουμε ορίσει anchors με διαφορετικές χρονικές διάρκειες, οι οποίες είναι 16, 12, 8 και 4 καρέ. Anchors με διάρκεια  $<$  διάρκεια του δείγματος (16 καρέ) μπορούν να γραφτούν ως διάνυσμα 4k με μηδενικές συντεταγμένες στα καρέ μεγαλύτερα από τη χρονική διάρκεια. Για παράδειγμα, ένα anchor με 2 καρέ διάρκεια, ξεκινώντας από το 2ο καρέ και

τερματίζοντας στον 3ο μπορεί να γραφεί ως  $(0, 0, 0, 0, x_1, y_1, x'_1, y'_1, x_2, y_2, x'_2, y'_2, 0, 0, 0, 0)$  εάν η διάρκεια δείγματος είναι 4 καρέ.



Σχήμα 1.10: Τηε στρυστυρε οφ THIN αςζορδινγ το νεω αππροαση

Αυτή η νέα προσέγγιση μας οδήγησε στο να αλλάξουμε την δομή του TPN. Η νέα δομή του απεικονίζεται στην εικόνα 1.10. Όπως μπορούμε να δούμε, προσθέσαμε scoring και regression layers για κάθε διάρκεια. Έτσι, το TPN ακολουθεί τα επόμενα βήματα για να παράγει ToIs.

1. Στην αρχή, τροφοδοτούμε τον χάρτη χαρακτηριστικών, που εξάγεται από το 3D ResNet34, ως είσοδο σε ένα 3D Convolutional Layer με μέγεθος πυρήνα = 1, stride = 1 και χωρίς padding.
2. Από το Convolutional Layer, έχουμε ως έξοδο ένα χάρτη ενεργοποίησης με διαστάσεις (256, 16, 7, 7). Για τη μείωση της χρονικής διάστασης, χρησιμοποιούμε 4 pooling layers, ένα για κάθε δείγμα διάρκειας με μεγέθη πυρήνα (16, 1, 1), (12, 1, 1), (8, 1, 1) και (4, 1, 1) και stride = 1, για τη διάρκεια του δείγματος 16, 12, 8 και 4 καρέ αντίστοιχα. Έτσι, έχουμε χάρτες ενεργοποίησης με διαστάσεις (256, 1, 7, 7), (256, 5, 7, 7), (256, 9, 7, 7) και (256, 13, 7, 7), στις οποίες η δεύτερη διάσταση είναι ο αριθμός των πιθανών χρονικών διακυμάνσεων. Για παράδειγμα, σε χάρτη χαρακτηριστικών με διαστάσεις (256, 5, 7, 7), το οποίο σχετίζεται με anchors με διάρκεια 12 καρέ, μπορούμε να έχουμε 5 πιθανές περιπτώσεις, από το καρέ 0 μέχρι το καρέ 11, από το καρέ 1 μέχρι το καρέ 12 κλπ.



3. Ξανά, όπως και στην προηγούμενη προσέγγιση, για κάθε pixel του χάρτη ενεργοποίησης αντιστοιχούμε  $\mathbf{n} = \mathbf{k} = \mathbf{15}$  anchors (5 κλίμακες από 1, 2, 4, 8, 16, και 3 aspect ratios 1:1, 1:2, 2:1). Φυσικά, έχουμε 4 διαφορετικούς χάρτες ενεργοποίησης, με 1, 5, 9 και 13 διαφορετικές περιπτώσεις και  $7 \times 7$  διαστάσεις σε κάθε φίλτρο. Έτσι, συνολικά έχουμε  $28 \cdot 15 \cdot 49 = 20580$  διαφορετικά anchors. Αντίστοιχα, έχουμε 20580 διαφορετικούς στόχους regression.

### 1.5.1 Training

Η διαδικασία τραινινγκ παραμένει σχεδόν η ίδια όπως και στην προηγούμενη προσέγγιση. Έτσι, και πάλι, εμείς τυχαία επιλέγουμε ένα τμήμα βίντεο και τα αντίστοιχα πραγματικά tubes. Όμως, θεωρούμε τα anchors ως προσκήνιο όταν έχουν επικάλυψη μεγαλύτερη από 0,8 με οποιαδήποτε πραγματικό tube, ενώ θεωρούμε anchors παρασκηνίου αυτά των οποίων η επικάλυψη είναι μεγαλύτερη που 0,1 και μικρότερη από 0,3. Δεν ασχολούμαστε με τα υπόλοιπα anchors.

Dataset	Pooling	Recall(0.5)	Recall(0.4)	Recall(0.3)
JHMDDB	mean	0.6866	0.7687	0.8582
	max	0.8134	0.8694	0.9216
UCF	avg	0.5435	0.6326	0.7075
	max	0.6418	0.7255	0.7898

Table 1.5: Recall results using 2nd approach for anchors

Όπως δείχνει ο πίνακας 1.5, είναι προφανές ότι έχουμε καλύτερες επιδόσεις recall σε σύγκριση με την προηγούμενη προσέγγιση. Επιπλέον, μπορούμε να δούμε ότι το 3D max pooling αποδίδει καλύτερα από το 3D avg pooling. Η διαφορά μεταξύ των δύο είναι περίπου 10%, η οποία είναι αρκετά μεγάλη για να μας κάνει να επιλέξουμε το max pooling ως λειτουργία ομαδοποίησης πριν από τη λήψη των scores και regression targets των anchors.

### 1.5.2 Προσθήκη Regressor

Ακόμα και αν, το TPN μας εξάγει κουτιά σε επίπεδο καρέ, πρέπει να βελτιώσουμε αυτές τις προβλέψεις προκειμένου να αλληλοεπικαλύπτονται με τα πραγματικά κουτιά όσο το δυνατόν καλύτερα. Έτσι, σε πλήρη αντιστοιχία με την προηγούμενη προσέγγιση, προσθέσαμε έναν Regressor για να προσπαθήσουμε να πάρουμε καλύτερα αποτελέσματα recall.

**3D Roi align** Σε αυτή την προσέγγιση, γνωρίζουμε ήδη τις χωρικές συντεταγμένες. Έτσι, μπορούμε να χρησιμοποιήσουμε τη μέθοδο που προτείνεται από τους Girdhar et al. 2017. Αποτελεί επέκταση του RoiAlign χωρίζοντας το tube σε  $T$  δυσδιάστατα κουτιά. Στη συνέχεια, χρησιμοποιείται το κλασικό RoiAlign για

να εξαχθεί μια περιοχή από κάθε μίας από τα χρονικά slices στον χάρτη ενεργοποίησης. Μετά από αυτό, τα feature maps που προέκυψαν μέσω του RoiAlign συνδέονται στην διάσταση του χρόνου, ώστε να προκύψει χάρτης χαρακτηριστικών με διαστάσεις  $T \times R \times R$ , όπου  $R$  είναι η ανάλυση εξόδου του RoiAlign, το οποίο είναι 7 στην περίπτωση μας.

Εκπαιδεύουμε τον Regressor μας χρησιμοποιώντας την ίδια loss function όπως ο τύπος της προηγούμενης προσέγγισης που είναι:

$$L = \sum_i L_{cls}(p_i, p_i^*) + \sum_i L_{cls}(p_{fixed,i}, p_{fixed,i}^*) + \sum_i p_i^* L_{reg}(t_i, t_i^*) + \sum_i p_{fixed,i}^* L_{reg}(t_{fixed,i}, t_{fixed,i}^*) + \sum_i q_i^* L_{reg}(c_i, c_i^*) +$$

Και σ' αυτήν την προσέγγιση χρησιμοποιούμε 2 διαφορετικές αρχιτεκτονικές για την υλοποίηση του Regressor. Ως πρώτη προσέγγιση, χρησιμοποιούμε ένα 3D Convolutional Layer ακολουθούμενο από 2 γραμμικά Layer. Αντίστοιχα, στην δεύτερη προσέγγιση χρησιμοποιούμε ένα 2D Convolution Layer ακολουθούμενο, και αυτό, από 2 γραμμικά Layer. Η πρώτη προσέγγιση είναι ακριβώς η ίδια με πριν. Ωστόσο, η δεύτερη προσέγγιση αντιμετωπίζει τα feature maps σαν να μην υπάρχουν χρονικές εξαρτήσεις μεταξύ τους. Δηλαδή:

1. Στην αρχή, χρησιμοποιούμε το 3D Roi Align για να εξάγουμε τα feature maps Και μετά τα κανονικοποιούμε. Έστω λοιπόν ότι προκύπτουν  $k$  χάρτες ενεργοποίησης με διαστάσεις  $(k, 256, 16, 7, 7)$
2. Χωρίζουμε τα υποψήφια ToIs σε  $T$  2D κουτιά, οπότε οι διαστάσεις του τένσορα που περιέχει τις συντεταγμένες των ToIs γίνονται από  $(k, 4 \cdot \text{samplerduration})$  σε  $(k, \text{samplerduration}, 4)$ . Διαφοροποιούμε τον τένσορα ώστε να πάρει τις διαστάσεις  $(k \cdot \text{samplerduration}, 4)$ , όπου οι πρώτες  $k$  συντεταγμένες αναφέρονται στο πρώτο καρέ, οι επόμενες  $k$  στο δεύτερο κλπ.
3. Αντίστοιχα διαφοροποιούμε και τους χάρτες ενεργοποίησης όπου από διαστάσεις  $(k, 64, \text{samplerduration}, 7, 7)$  καταλήγουμε σε χάρτες με διαστάσεις  $(k \cdot \text{samplerduration}, 64, 7, 7)$ . Πλέον λοιπόν επεξεργαζόμαστε του χάρτες χαρακτηριστικών σαν να είναι δυσδιάστατοι. Έτσι τροφοδοτούμε το 2D Convolutional Layer και ακολουθούμενο από τα άλλα δύο γραμμικά Layer.
4. Τα εξαγόμενα targets, φυσικά θα είναι μόνο 4 όσο δηλαδή για 1 καρέ.

Με βάση τους παραπάνω πίνακες 1.6 και 1.7, η καλύτερη επίδοση προκύπτει για τους χάρτες χαρακτηριστικών μεγέθους  $(64, 16, 7, 7)$ . Αυτό το αποτέλεσμα ήταν αναμενόμενο γιατί αυτοί οι χάρτες ενεργοποίησης βρίσκονται «πιο κοντά» στα πραγματικά χαρακτηριστικά. Συγκρίνοντας τις δύο προτεινόμενες μεθόδους παρατηρούμε ότι η δεύτερη, αυτή δηλαδή που χρησιμοποιεί 2D Convolutional Layer έχει τα καλύτερα αποτελέσματα. Αν και βελτιώσαμε την απόδοση του TPN ακόμα δεν έχουμε καταφέρει να έχουμε σχεδόν σε όλες τις περιπτώσεις καλές προτάσεις tube.

Dataset	Feat. Map	Recall(0.5)	Recall(0.4)	Recall(0.3)
JHMDB	64	0.7985	0.903	0.9552
	128	0.7836	0.8881	0.944
UCF	64	0.5794	0.7206	0.8134
	128	0.5622	0.7204	0.799

Table 1.6: Recall performance when using a 3D Convolutional Layer in Regressor’s architecture

Dataset	Feat. Map	Recall(0.5)	Recall(0.4)	Recall(0.3)
JHMDB	64	0.8358	0.9216	0.9739
	128	0.8172	0.9142	0.9627
	256	0.7724	0.8731	0.9328
UCF	64	0.6368	0.7346	0.7737
	128	0.6363	0.7133	0.7822
	256	0.6363	0.7295	0.7822

Table 1.7: Recall performance when using a 2D Convolutional Layer instead of 3D in Regressor’s model

### 1.5.3 Μείωση της διάρκειας του δείγματος

Προκειμένου να πετύχουμε καλύτερα αποτελέσματα αποφασίσαμε να μειώσουμε την διάρκεια του δείγματος από τα 16 καρέ σε 8 και 4 αντίστοιχα. Κι αυτό γιατί με αυτόν τον τρόπο θα μειωθούν παράλληλα με τον αριθμός τους οι διαστάσεις των anchors, άρα και οι διαστάσεις των στόχων παλινδρόμησης και γενικά ο αριθμός των παραμέτρων που πρέπει να εκπαιδευτούν από το σύστημα.

Εκπαιδευόμαστε το TPN με και χωρίς Regressor προκειμένου να βρούμε την κατάλληλη αρχιτεκτονική. Τα αποτελέσματα χωρίς Regressor παρουσιάζονται στον πίνακα 1.8 ενώ τα αποτελέσματα της αρχιτεκτονικής με Regressor στον πίνακα 1.9.

Dataset	Sample dur	Recall(0.5)	Recall(0.4)	Recall(0.3)
JHMDB	16	0.8134	0.8694	0.9216
	8	0.9515	0.9888	1.0000
	4	0.8843	0.9627	0.9888
UCF	16	0.6418	0.7255	0.7898
	8	0.7942	0.8877	0.9324
	4	0.7879	0.8924	0.9462

Table 1.8: Recall results when reducing sample duration to 4 and 8 frames per video segment

Το πρώτο συμπέρασμα που προκύπτει από τους πίνακες 1.8 και 1.9 είναι ότι όντως μειώνοντας την διάρκεια του δείγματος πετυχαίνουμε καλύτερα αποτελέσματα. Παράλληλα, με βάση τον πίνακα 1.9 και δεδομένων των προηγούμενων αποτελεσμάτων (Πίνακες 1.6 και 1.7) γίνεται ξεκάθαρο ότι η προσέγγιση που περιλαμβάνει

Dataset	Sample dur	Type	Recall(0.5)	Recall(0.4)	Recall(0.3)
UCF	8	2D	0.8078	0.8870	0.9419
		3D	0.8193	0.8930	0.9487
	4	2D	0.7785	0.8914	0.9457
		3D	0.7449	0.8605	0.9362
JHDMBD	8	2D	0.9366	0.9851	0.9925
		3D	0.8918	0.9776	0.9963
	4	2D	0.9552	0.9963	1.0000
		3D	0.9142	0.9701	0.9888

Table 1.9: Recall results when a regressor and sample duration equal with 4 or 8 frames per video segment

ένα 2D Convolutional Layer υπερέχει αυτής με το 3D Convolutional Layer. Σ' ότι αφορά τα σύνολα δεδομένων παρατηρούμε ότι η προσέγγιση με διάρκεια δείγματος 8 καρέ υπερέχει σχεδόν σε όλες τις περιπτώσεις. Συνεπώς, για τα επόμενα κεφάλαια θα προτιμάται να χρησιμοποιείται αυτή.