ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Μοντέλου Μηχανικής Όρασης για τον Έλεγχο Τεστ Πολλαπλής Επιλογής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΕΥΑΓΓΕΛΟΥ ΓΚΟΝΕΖΟΥ

(ΑΕΜ: 1663)

Επιβλέπων : Σινάτικας Ιωάννης

Καθηγητης

Καστοριά Καστοριά 10/2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Μοντέλου Μηχανικής Όρασης για τον Έλεγχο Τεστ Πολλαπλής Επιλογής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΕΥΑΓΓΕΛΟΥ ΓΚΟΝΕΖΟΥ

(ΑΕΜ: 1663)

Επιβλέπων : Σινάτκας Ιωάννης

Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδίοτητα Μέλους	Ιδίοτητα Μέλους	Ιδίοτητα Μέλους

Καστοριά Καστοριά 10- 2024

Ευχαριστίες

Στην οικογένειά μου για την στήριξη καθόλη την διάρκεια της φοίτησής μου στη σχολή.

Περίληψη

Αυτή η διατριβή διερευνά την εφαρμογή και την εκπαίδευση του YOLOv8 (You Only Look Once) για εργασίες ανίχνευσης αντικειμένων. Καλύπτουμε την προεπεξεργασία των συνόλων δεδομένων χρησιμοποιώντας τεχνικές όπως η θόλωση και η μετατροπή σε κλίμακα του γκρι και η συνδυαστική τους επίδραση στην απόδοση του μοντέλου. Το έγγραφο περιγράφει λεπτομερώς τη διαδικασία εκπαίδευσης χρησιμοποιώντας συγκεκριμένες εντολές και τον σχολιασμό των δεδομένων χρησιμοποιώντας το LabelImg. Εμβαθύνουμε επίσης στη διαμόρφωση των αρχείων YAML που είναι απαραίτητα για την εκπαίδευση του YOLOv8 και εξηγούμε βασικές μετρήσεις αξιολόγησης, όπως η μήτρα σύγχυσης και η ανάλυση παρτίδας εκπαίδευσης. Παρουσιάζεται μια μελέτη περίπτωσης για την αυτοματοποιημένη βαθμολόγηση χρησιμοποιώντας το YOLOv8, με μια λεπτομερή εξήγηση του σχετικού κώδικα Python. Η εργασία αυτή υποστηρίζεται από αναφορές σε σχετική βιβλιογραφία και πρακτικούς οδηγούς.

Λέξεις Κλειδιά:

YOLOv8, Αναγνώριση αντικειμένων, Υπολογιστική όραση, Αναγνώριση σε πραγματικό χρόνο, Μηχανική μάθηση

Abstract

This thesis explores the implementation and training of YOLOv8 (You Only Look Once) for object detection tasks. We cover the preprocessing of datasets using techniques such as blurring and grayscale conversion, and their combined effect on model performance. The document details the training process using specific commands and the annotation of data using LabelImg. We also delve into the configuration of YAML files necessary for training YOLOv8 and explain key evaluation metrics such as the confusion matrix and training batch analysis. A case study on automated grading using YOLOv8 is presented, with a thorough explanation of the associated Python code. This work is supported by references to relevant literature and practical guides.

KeyWords:

YOLOv8, Object Detection, Computer Vision, Real-Time Detection, Machine learning

Πίνακας Περιεχομένων

Λίστα Εικόνων

Εισαγωγή

Τα αυτοματοποιημένα συστήματα βαθμολόγησης έχουν γίνει απαραίτητα, ιδιαίτερα σε περιβάλλοντα δοκιμών μεγάλης κλίμακας. Παραδοσιακά, τα συστήματα Optical Mark Recognition (OMR) ήταν η καλύτερη λύση. Ωστόσο, οι εξελίξεις στην υπολογιστική όραση επέτρεψαν πιο αποτελεσματικές και ακριβείς λύσεις χρησιμοποιώντας μοντέλα ανίχνευσης αντικειμένων όπως το YOLOv8. Αυτή η εργασία εστιάζει στη μόχλευση του YOLOv8 για τον εντοπισμό περιοχών απαντήσεων στα φύλλα απαντήσεων MCQ και την αυτοματοποίηση της διαδικασίας βαθμολόγησης.

1. YOLOv8 Επισκόπηση

Το YOLOv8 είναι ένα προηγμένο μοντέλο ανίχνευσης αντικειμένων σχεδιασμένο για επεξεργασία σε πραγματικό χρόνο. Σε αντίθεση με τους προκατόχους του, προσφέρει βελτιωμένη ακρίβεια, ταχύτητα και ευελιξία στο χειρισμό πολλαπλών αντικειμένων σε διάφορα περιβάλλοντα. Το YOLOv8 είναι ιδιαίτερα κατάλληλο για εργασίες όπως η ανίχνευση αντικειμένων, η παρακολούθηση και η τμηματοποίηση.

1.1 Βασικά χαρακτηριστικά του YOLOv8

Οι βασικές λειτουργίες του YOLOv8 είναι πέντε. Ανίχνευση αντικειμένων σε πραγματικό χρόνο, Single-Pass Detection, Βελτιωμένη Αρχιτεκτονική Backbone, Ευελιξία στα μεγέθη μοντέλων, Πολλαπλές εργασίες.

1.1.1 Ανίχνευση αντικειμένων σε πραγματικό χρόνο

Τα μοντέλα YOLO είναι γνωστά για την ταχύτητά τους. Το YOLOv8 μπορεί να επεξεργαστεί πολλαπλές εικόνες ανά δευτερόλεπτο, καθιστώντας το κατάλληλο για εφαρμογές όπως παρακολούθηση, αυτόνομα οχήματα και συστήματα βαθμολόγησης σε πραγματικό χρόνο.

1.1.2 Single-Pass Detection

Σε αντίθεση με τους παραδοσιακούς αλγόριθμους ανίχνευσης αντικειμένων που σαρώνουν μια εικόνα πολλές φορές, το YOLOv8 ανιχνεύει αντικείμενα σε ένα μόνο πέρασμα, γεγονός που αυξάνει την ταχύτητα επεξεργασίας διατηρώντας παράλληλα την ακρίβεια.

1.1.3 Βελτιωμένη Αρχιτεκτονική Backbone

Το YOLOv8 περιλαμβάνει αναβαθμίσεις στη ραχοκοκαλιά του (το τμήμα του μοντέλου που είναι υπεύθυνο για την εξαγωγή χαρακτηριστικών), επιτρέποντάς του να ανιχνεύει μικρότερα και πιο σύνθετα αντικείμενα με μεγαλύτερη ακρίβεια.

1.1.4 Ευελιξία στα μεγέθη μοντέλων

Το YOLOv8 συνοδεύεται από διάφορες εκδόσεις, από μικρότερα, ταχύτερα μοντέλα όπως το YOLOv8n (nano) έως μεγαλύτερες, πιο ακριβείς εκδόσεις όπως το YOLOv8x (εξαιρετικά μεγάλο). Αυτό επιτρέπει στους χρήστες να επιλέξουν μια έκδοση με βάση τις ανάγκες τους—ταχύτητα, ακρίβεια ή ισορροπία μεταξύ των δύο.

1.1.5 Πολλαπλές εργασίες

Εκτός από την ανίχνευση αντικειμένων, το YOLOv8 μπορεί επίσης να χειριστεί εργασίες όπως η παρακολούθηση αντικειμένων, η τμηματοποίηση εικόνων και η τμηματοποίηση παρουσίας. Αυτή η ευελιξία το καθιστά ευρέως εφαρμόσιμο σε μια ποικιλία βιομηχανιών, όπως η εκπαίδευση, η υγειονομική περίθαλψη, το λιανικό εμπόριο και τα αυτόνομα συστήματα.

1.2 Εφαρμογές και Σημασία

Η ευελιξία και η αποτελεσματικότητα του YOLOv8 το καθιστούν κατάλληλο για ένα ευρύ φάσμα εφαρμογών στον πραγματικό κόσμο. Ακολουθούν ορισμένοι βασικοί τομείς στους οποίους εφαρμόζεται το YOLOv8, τονίζοντας τον ρόλο του σε διάφορους κλάδους και τη σημασία του στις εργασίες ανίχνευσης αντικειμένων.

1.2.1 Επιτήρηση και Ασφάλεια

Το YOLOv8 χρησιμοποιείται σε συστήματα παρακολούθησης βίντεο για τον εντοπισμό και την παρακολούθηση αντικειμένων όπως άτομα, οχήματα ή ακόμα και ύποπτα αντικείμενα σε πραγματικό χρόνο. Η ταχύτητά του το καθιστά αποτελεσματικό για την παρακολούθηση μεγάλων περιοχών και την ενεργοποίηση ειδοποιήσεων με βάση τα εντοπισμένα αντικείμενα.

1.2.2 Αυτόνομα Οχήματα

Η ανίχνευση αντικειμένων είναι ζωτικής σημασίας για τα αυτόνομα αυτοκίνητα, καθώς το σύστημα πρέπει να αναγνωρίζει και να ανταποκρίνεται σε αντικείμενα όπως πεζοί, άλλα οχήματα, σήματα κυκλοφορίας και εμπόδια. Η ικανότητα του YOLOv8 να επεξεργάζεται δεδομένα σε πραγματικό χρόνο το καθιστά πολύτιμο συστατικό των συστημάτων αυτόνομης οδήγησης.

1.2.3 Διαχείριση λιανικής και αποθεμάτων

Το YOLOv8 χρησιμοποιείται επίσης σε περιβάλλοντα λιανικής για αυτοματοποιημένα ταμεία, διαχείριση ραφιών και παρακολούθηση αποθέματος. Μπορεί να ανιχνεύσει αντικείμενα στα ράφια, να παρακολουθεί τα επίπεδα αποθεμάτων και ακόμη και να διευκολύνει καταστήματα χωρίς ταμείο όπου οι πελάτες μπορούν να αρπάξουν αντικείμενα και να φύγουν χωρίς χειροκίνητη σάρωση.

1.2.4 Ιατρική Απεικόνιση

Στον κλάδο της υγειονομικής περίθαλψης, το YOLOv8 μπορεί να εφαρμοστεί στην ιατρική απεικόνιση για την ανίχνευση όγκων, ανωμαλιών ή άλλων σχετικών χαρακτηριστικών σε ακτινογραφίες, μαγνητικές τομογραφίες και αξονικές τομογραφίες. Η ικανότητά του να επεξεργάζεται γρήγορα μεγάλες ποσότητες δεδομένων εικόνας το καθιστά χρήσιμο για διαγνωστικά εργαλεία.

1.2.5 Ρομποτική και Drones

Στη ρομποτική, το YOLOv8 εφαρμόζεται ευρέως για τη βελτίωση της αναγνώρισης αντικειμένων για εργασίες όπως η ταξινόμηση, η επιλογή και η πλοήγηση. Χρησιμοποιείται επίσης σε drones για παρακολούθηση αντικειμένων, χαρτογράφηση και αυτόνομη πλοήγηση.

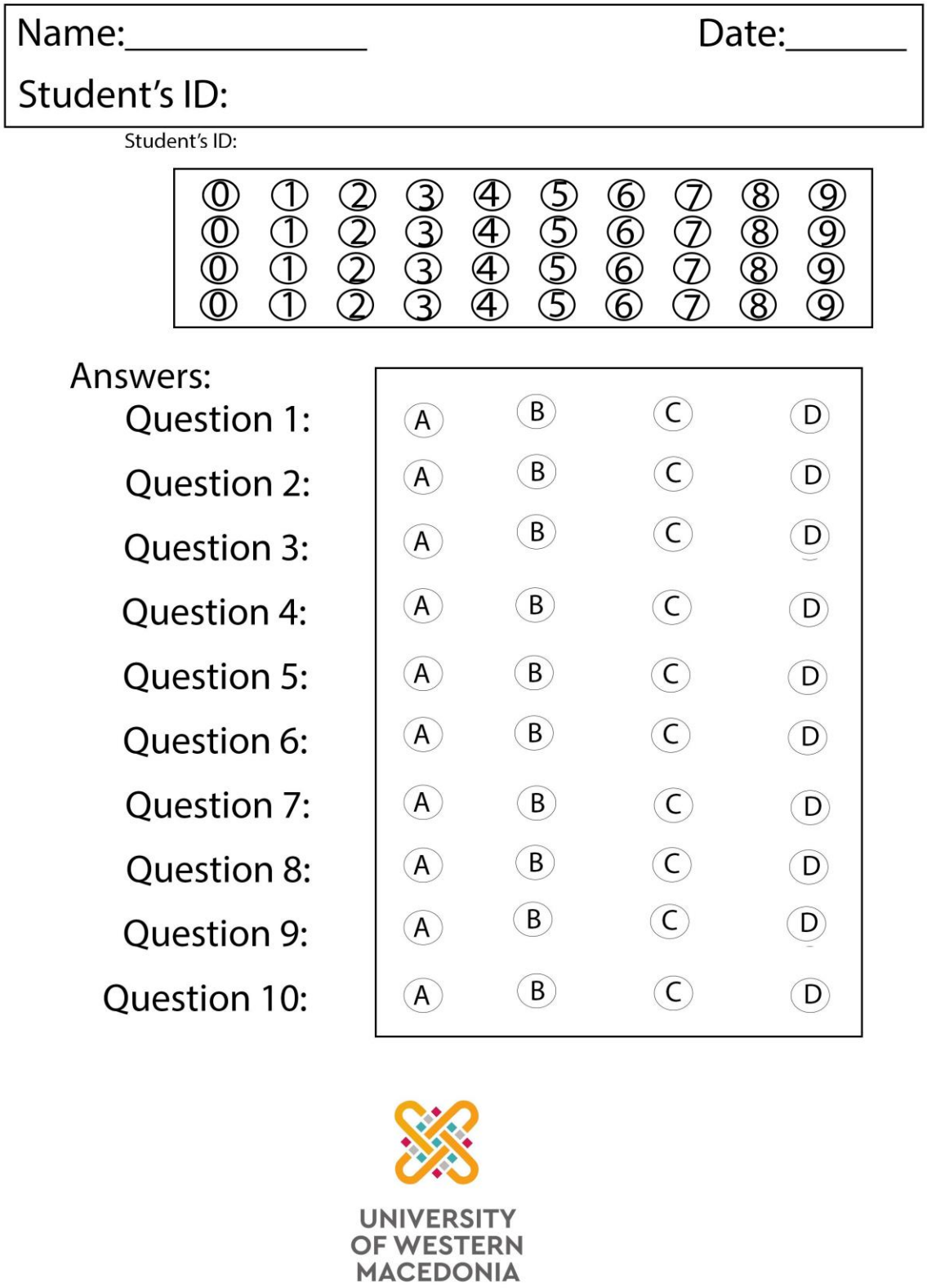
1.2.6 Γεωργία

Το YOLOv8 χρησιμοποιείται επίσης στη γεωργία για εργασίες όπως η παρακολούθηση των καλλιεργειών, η ανίχνευση παρασίτων και η αξιολόγηση της υγείας των φυτών. Χρησιμοποιώντας drones εξοπλισμένα με συστήματα που βασίζονται στο YOLOv8, οι αγρότες μπορούν να παρακολουθούν μεγάλα χωράφια σε πραγματικό χρόνο, επιτρέποντας την έγκαιρη ανίχνευση προβλημάτων όπως φυτικές ασθένειες ή προσβολές.

2. Προετοιμασία συνόλου δεδομένων

Για ακριβέστερα αποτελέσματα είναι πρέπων να μαζέψουμε ένα σύνολο δειγμάτων σε διαφορετικές συνθήκες ώστε το μοντέλο μας να μπορεί να κάνει καλύτερες προβλέψεις. Η χρήση του PyCharm για την ανάπτυξη προγραμμάτων σε Python παρέχει ένα αποτελεσματικό περιβάλλον για τη διαχείριση εξαρτήσεων και την οργάνωση κώδικα σε λειτουργικές μονάδες. Όταν εργαζόμαστε σε ένα έργο που απαιτεί εξωτερικές βιβλιοθήκες όπως NumPy, Ultralytics, PyTorch και OpenCV, πρέπει πρώτα να βεβαιωθούμε ότι αυτά τα πακέτα είναι εγκατεστημένα. Το ενσωματωμένο τερματικό και ο διαχειριστής πακέτων της PyCharm κάνουν αυτή τη διαδικασία απλή. Μπορούμε να εγκαταστήσουμε τις απαιτούμενες βιβλιοθήκες εκτελώντας εντολές όπως το pip install numpy ultralytics torch opencv-python απευθείας στο τερματικό. Αυτό διασφαλίζει ότι όλα τα απαραίτητα πακέτα είναι διαθέσιμα για το έργο σας, επιτρέποντάς σας να αξιοποιήσετε προηγμένες λειτουργίες όπως αριθμητικούς υπολογισμούς (NumPy), ανίχνευση αντικειμένων (Ultralytics και YOLOv8), βαθιά εκμάθηση (PyTorch) και επεξεργασία εικόνας (OpenCV).

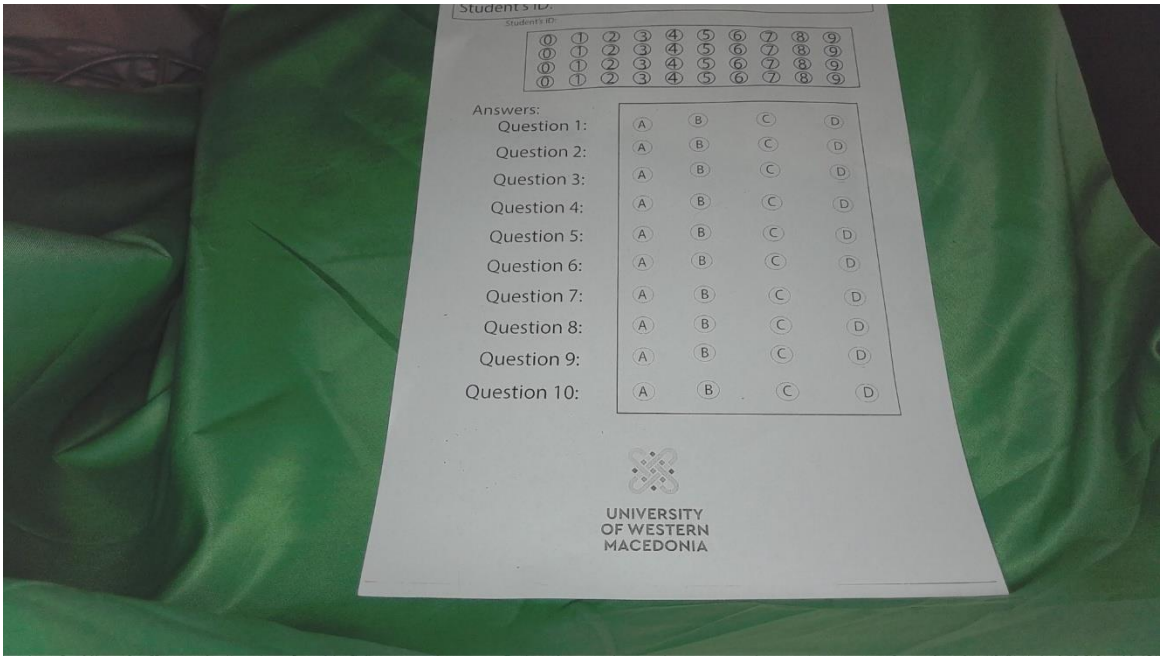
Ο διαχωρισμός του έργου σας σε πολλαπλά αρχεία, όπως το main.py και το functions.py, ακολουθεί τη λογική του αρθρωτού προγραμματισμού, ο οποίος βελτιώνει την οργάνωση του κώδικα και τη δυνατότητα συντήρησης. Σε αυτήν τη δομή, το main.py είναι υπεύθυνο για τον έλεγχο της ροής του κύριου προγράμματος, ενώ το functions.py περιέχει επαναχρησιμοποιήσιμες λειτουργίες και βοηθητικά προγράμματα. Για παράδειγμα, το functions.py μπορεί να περιλαμβάνει λειτουργίες προεπεξεργασίας εικόνας με χρήση OpenCV ή ρουτίνες φόρτωσης μοντέλων με PyTorch και Ultralytics. Στη συνέχεια, αυτές οι συναρτήσεις εισάγονται στο main.py (από την εισαγωγή συναρτήσεων preprocess_image) για να διατηρηθεί η βασική λογική καθαρή και εστιασμένη σε εργασίες υψηλού επιπέδου. Αυτή η προσέγγιση επιτρέπει ευκολότερο εντοπισμό σφαλμάτων, δοκιμές και μελλοντικές επεκτάσεις του κώδικα, διασφαλίζοντας ότι κάθε τμήμα του έργου είναι καλά δομημένο και εύκολο στη διαχείριση.



εικόνα 1. Πρότυπο εγγράφου.

2.1 Προεπεξεργασία εικόνας

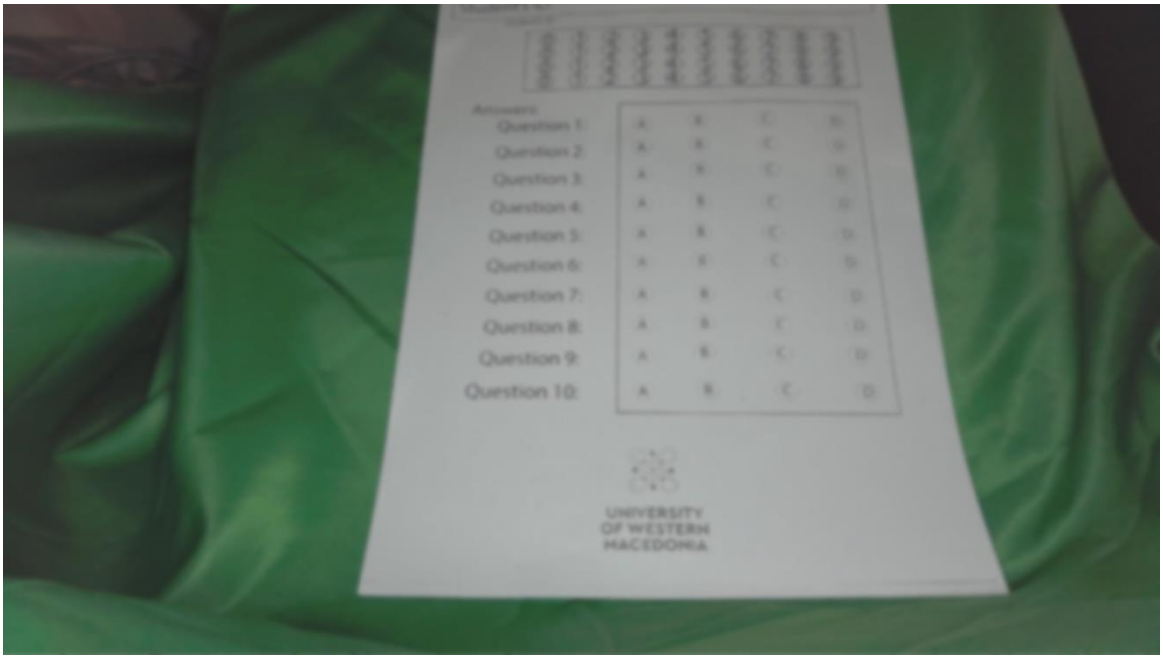
Σε εργασίες ανίχνευσης αντικειμένων όπως αυτές που αφορούν το YOLOv8, η προεπεξεργασία διαδραματίζει κρίσιμο ρόλο στη διασφάλιση ότι το μοντέλο μπορεί να ανιχνεύσει με ακρίβεια αντικείμενα υπό διαφορετικές συνθήκες. Η προεπεξεργασία αναφέρεται στα βήματα που λαμβάνονται για την προετοιμασία των ακατέργαστων δεδομένων εικόνας για ανάλυση, τη βελτίωση της ποιότητας της εισόδου και τη μείωση του θορύβου οι τεχνικές είναι οι εξής Blurring, Grayscale και ο συνδυασμός αυτών των δύο.



εικόνα 2. Αρχική φωτογραφία εκπαίδευσης του μοντέλου.

2.1.1 Blurring

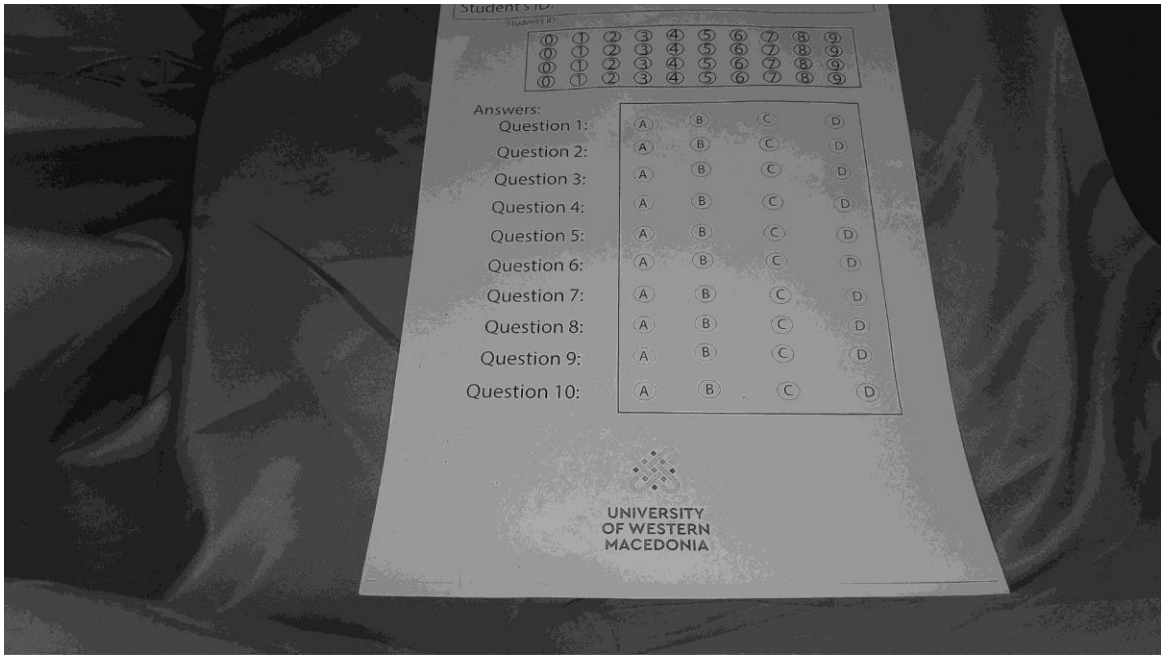
Αυτή η τεχνική μειώνει τον θόρυβο και τις λεπτομέρειες στην εικόνα εξομαλυνοντας την. Το Gaussian blur, συγκεκριμένα, χρησιμοποιήθηκε για τη μείωση του θορύβου υψηλής συχνότητας και τη βελτίωση των ορίων των αντικειμένων. Το θάμπωμα βοηθά στην αποτροπή της υπερβολικής προσαρμογής του μοντέλου σε μικρές διακυμάνσεις στα δεδομένα που προκαλούνται από τον θόρυβο.



εικόνα 3. φωτογραφία εκπαίδευσης του μοντέλου με εφαρμογή του blurring.

2.1.2 Αποκλιμάκωση του γκρι

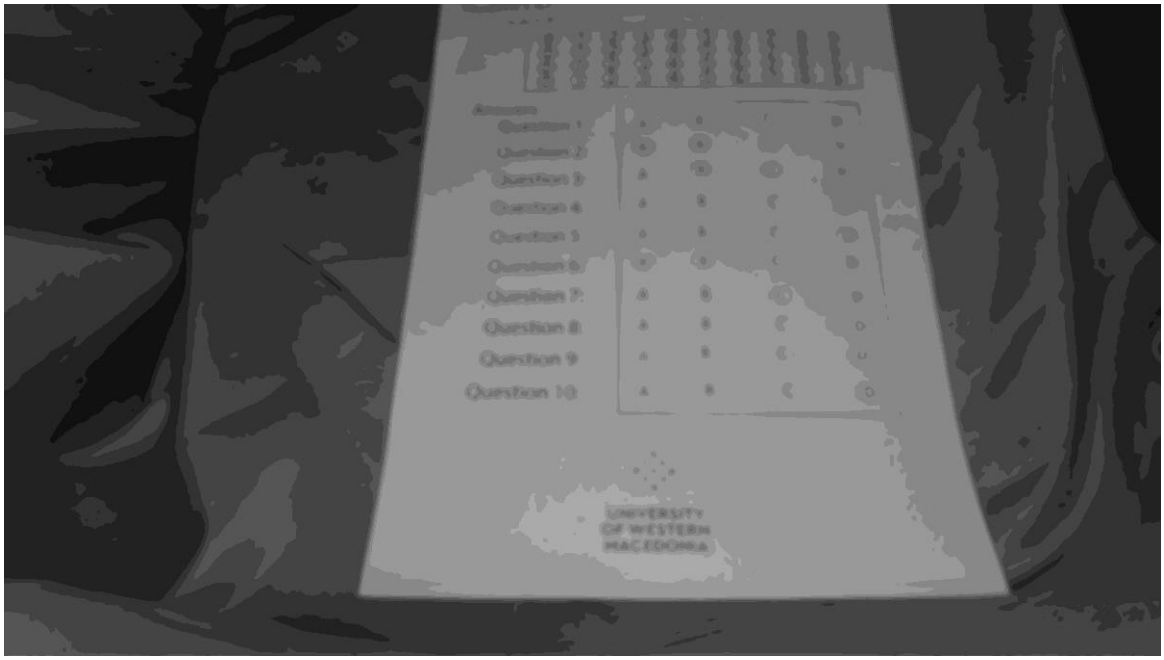
Η μετατροπή μιας εικόνας σε κλίμακα του γκρι μειώνει την πολυπλοκότητα εξαλείφοντας τις πληροφορίες χρώματος, οι οποίες μπορεί να μην είναι σχετικές με εργασίες ανίχνευσης αντικειμένων, όπως ο εντοπισμός περιοχών απάντησης. Μειώνοντας τη διάσταση χρώματος, το μοντέλο εστιάζει στο σχήμα και την αντίθεση, απλοποιώντας την εξαγωγή χαρακτηριστικών.Η μετατροπή μιας εικόνας σε κλίμακα του γκρι μειώνει την πολυπλοκότητα εξαλείφοντας τις πληροφορίες χρώματος, οι οποίες μπορεί να μην είναι σχετικές με εργασίες ανίχνευσης αντικειμένων, όπως ο εντοπισμός περιοχών απάντησης. Μειώνοντας τη διάσταση χρώματος, το μοντέλο εστιάζει στο σχήμα και την αντίθεση, απλοποιώντας την εξαγωγή χαρακτηριστικών.



εικόνα 4. φωτογραφία εκπαίδευσης του μοντέλου με εφαρμογή του Grayscale.

2.1.3 Συνδυασμός Blurring και Αποκλιμάκωση του γκρι

Η μετατροπή μιας εικόνας σε κλίμακα του γκρι μειώνει την πολυπλοκότητα εξαλείφοντας τις πληροφορίες χρώματος, οι οποίες μπορεί να μην είναι σχετικές με εργασίες ανίχνευσης αντικειμένων, όπως ο εντοπισμός περιοχών απάντησης. Μειώνοντας τη διάσταση χρώματος, το μοντέλο εστιάζει στο σχήμα και την αντίθεση, απλοποιώντας την εξαγωγή χαρακτηριστικών.



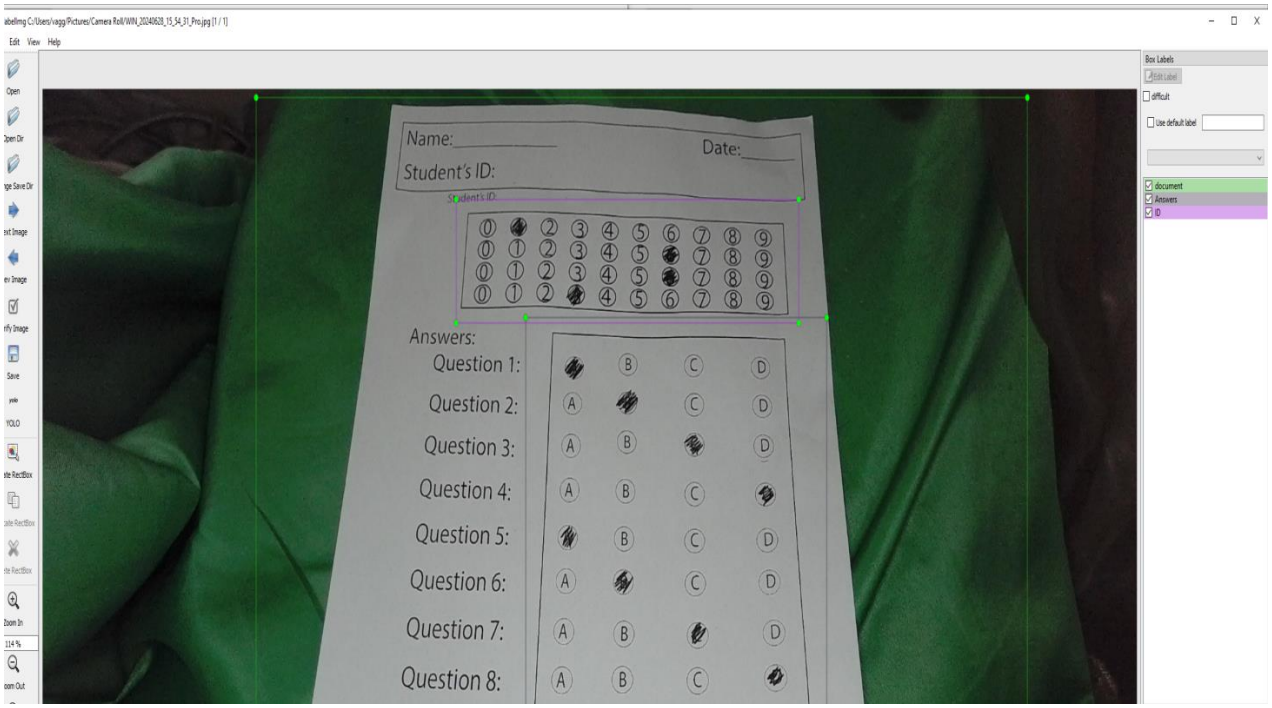
εικόνα 5. φωτογραφία εκπαίδευσης του μοντέλου με εφαρμογή του Συνδυασμός Blurring και Grayscale.

2.2 Annotation με LabelImg

Για μοντέλα εποπτευόμενης μάθησης όπως το YOLOv8, τα δεδομένα με ετικέτα είναι απαραίτητα. Το LabelImg είναι ένα εργαλείο γραφικού σχολιασμού ανοιχτού κώδικα που χρησιμοποιείται για τη μη αυτόματη επισήμανση εικόνων. Τα πλαίσια οριοθέτησης που σχεδιάζονται γύρω από αντικείμενα (σε αυτή την περίπτωση, περιοχές απαντήσεων) χρησιμεύουν ως η βασικό τεκμήριο για την εκπαίδευση του μοντέλου.

2.2.1 Οριοθέτηση κουτιών

Το LabelImg μας επιτρέπει να σχεδιάσουμε οριοθέτηση γύρω από αντικείμενα που θέλουμε να ανιχνεύσει το μοντέλο, εκχωρώντας μια ετικέτα κλάσης σε κάθε πλαίσιο.



εικόνα 6. Δημιουργία ετικετών για την εκπαίδευση του μοντέλου.

2.2.2 Εξαγωγή σχολιασμών

Μόλις επισημανθούν τα πλαίσια οριοθέτησης, το LabelImg εξάγει αυτούς τους σχολιασμούς σε μορφή PASCAL VOC ή YOLO. Στη μορφή YOLO, οι σχολιασμοί αποθηκεύονται ως αρχεία κειμένου, όπου κάθε γραμμή αντιστοιχεί σε ένα αντικείμενο με την ετικέτα κλάσης και τις κανονικοποιημένες συντεταγμένες του πλαισίου οριοθέτησής του.

2.3 YAML Files

Το YOLOv8 βασίζεται σε αρχεία διαμόρφωσης YAML (YAML Ain't Markup Language) για να καθορίσει το σύνολο δεδομένων και τις παραμέτρους του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Η δομή του αρχείου YAML περιέχει κρίσιμες λεπτομέρειες όπως τις διαδρομές προς τα σύνολα δεδομένων εκπαίδευσης και επικύρωσης, τον αριθμό των κλάσεων και τα ονόματά τους.

Ένα παράδειγμα αρχείου YAML που χρησιμοποιείται σε αυτό το έργο είναι το εξής:

```
train: ./images/train  # Διαδρομή προς το σύνολο δεδομένων εκπαίδευσης

val: ./images/val    # Διαδρομή προς το σύνολο δεδομένων επικύρωσης

nc: 4                    # Αριθμός κλάσεων (π.χ. A, B, C, D)

names: ['A', 'B', 'C', 'D']  # Τα ονόματα των κλάσεων
```

2.3.1 Train and val

Αυτά καθορίζουν τις διαδρομές προς τους καταλόγους όπου αποθηκεύονται οι εικόνες εκπαίδευσης και επικύρωσης και οι αντίστοιχοι σχολιασμοί τους.

2.3.2 Nc

Αναφέρεται στον αριθμό των κλάσεων που πρέπει να εντοπίσει το μοντέλο.

2.3.3 Names

Αυτή η ενότητα παρέχει τα ονόματα κλάσεων που αντιστοιχούν στις ετικέτες που χρησιμοποιούνται κατά τον σχολιασμό.

3. Εκπαίδευση μοντέλου

3.1 Εντολές Εκπαίδευσης

Το YOLOv8 χρησιμοποιεί συγκεκριμένες εντολές για την έναρξη της εκπαίδευσης. Αυτές οι εντολές καθορίζουν βασικές υπερπαραμέτρους και διαμορφώσεις μοντέλων. Ένα παράδειγμα εντολής που χρησιμοποιείται στην εκπαίδευση είναι:

```
yolo task=detect mode=train epochs=100
data=data_custom.yaml model=yolov8n.pt imgsz=640
```

3.1.1 task=detect

Καθορίζει ότι η εργασία είναι η ανίχνευση αντικειμένων (σε αντίθεση με την τμηματοποίηση ή την ταξινόμηση).

3.1.2 mode=train

Καθορίζει ότι το μοντέλο βρίσκεται σε λειτουργία εκπαίδευσης.

3.1.3 epochs=100

Το μοντέλο θα εκπαιδευτεί για 100 εποχές (επαναλήψεις σε ολόκληρο το σύνολο δεδομένων).

3.1.4 data=data_custom.yaml

Αυτό δείχνει το αρχείο YAML που ορίζει το σύνολο δεδομένων και τη δομή του.

3.1.5 model=yolov8n.pt

Καθορίζει την αρχιτεκτονική του μοντέλου που θα χρησιμοποιηθεί (το YOLOv8n είναι μια μικρή έκδοση του μοντέλου YOLOv8, βελτιστοποιημένη για ταχύτητα).

3.1.6 imgsz=640

Καθορίζει το μέγεθος της εικόνας εισόδου (640 x 640 pixel), το οποίο χρησιμοποιείται για την αλλαγή μεγέθους όλων των εικόνων στο σύνολο δεδομένων σε τυπικό μέγεθος.

Για περισσότερες πληροφορίες μπορείτε να επισκευθείτε την ιστοσελίδα της ultralytics[4].

3.2 Εκπαιδευτική Διαδικασία

Κατά τη διάρκεια της προπόνησης, το YOLOv8 επεξεργάζεται τις ετικέτες εικόνων και προσαρμόζει τα weights του μοντέλου μέσω της οπίσθιας διάδοσης. Ο στόχος είναι να ελαχιστοποιηθεί η διαφορά μεταξύ των προβλεπόμενων πλαισίων οριοθέτησης του μοντέλου και της αληθούς βάσης (τα πλαίσια οριοθέτησης με μη αυτόματο σχολιασμό). Δείτε πώς λειτουργεί η εκπαιδευτική διαδικασία.

3.2.1 Αρχικοποίηση

Το μοντέλο ξεκινά με τυχαία weights και κάνει προβλέψεις με βάση την εικόνα εισόδου.

3.2.2 Υπολογισμός απώλειας

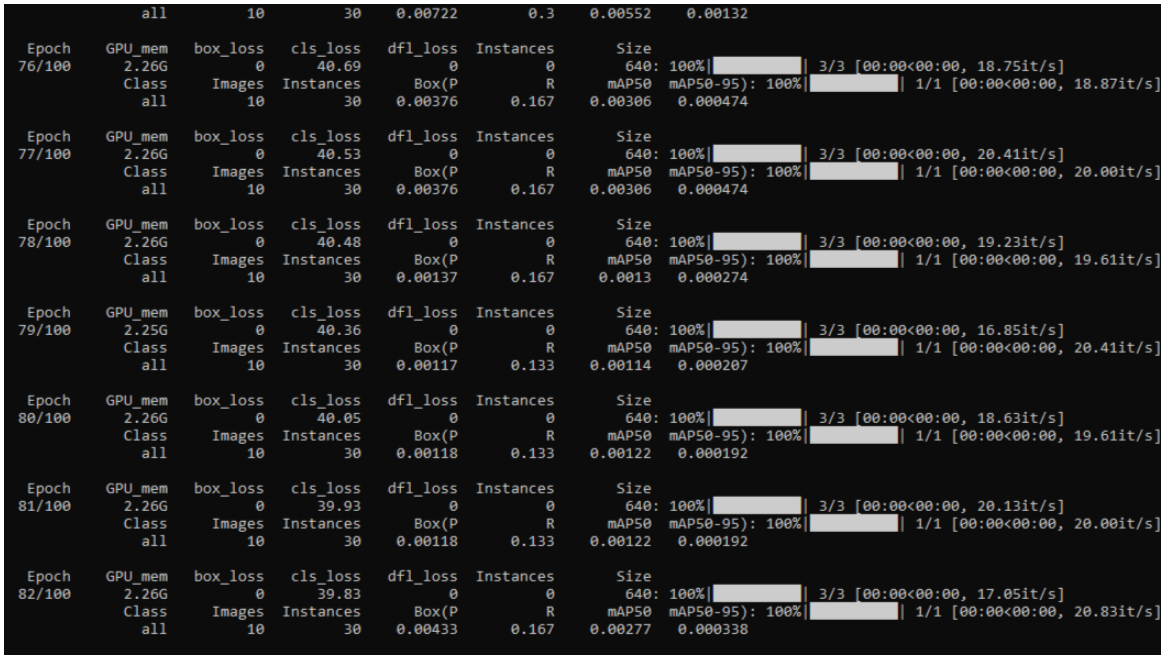
Η συνάρτηση απώλειας μετρά το σφάλμα μεταξύ των προβλεπόμενων πλαισίων οριοθέτησης και των ετικετών κλάσης σε σχέση με τους πραγματικούς σχολιασμούς. Το YOLOv8 χρησιμοποιεί έναν συνδυασμό απώλειας εντοπισμού (ακρίβεια οριοθέτησης), απώλειας ταξινόμησης (σωστή πρόβλεψη ετικέτας) και απώλειας εμπιστοσύνης (πόσο σίγουρο είναι το μοντέλο για τις προβλέψεις του).

3.2.3 Backpropagation

Με βάση την υπολογισμένη απώλεια, το μοντέλο προσαρμόζει τα weights του μέσω backpropagation, ενημερώνοντας τα weights για να μειώσει την απώλεια σε επόμενες επαναλήψεις.

3.2.4 Epochs

Η διαδικασία εκπαίδευσης επαναλαμβάνεται σε πολλές εποχές (100 σε αυτήν την περίπτωση), βελτιώνοντας σταδιακά την ακρίβεια του μοντέλου μειώνοντας τη συνολική απώλεια.



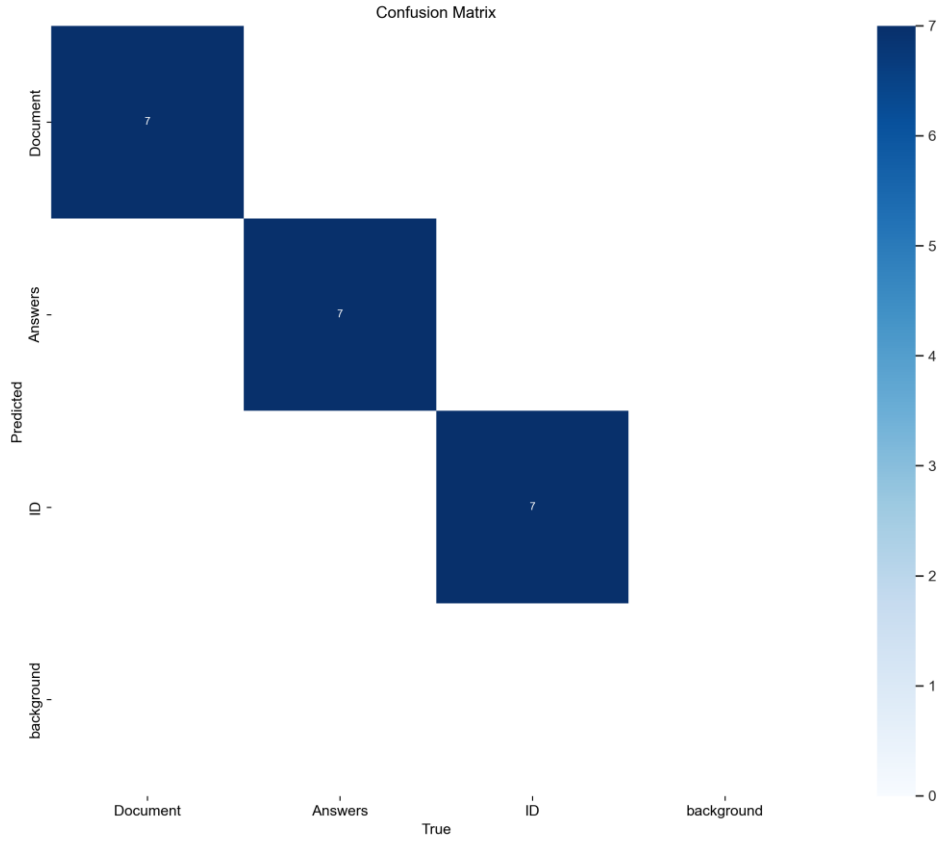
εικόνα 7. Εκπαίδευση του μοντέλου.

3.3 Μετρήσεις αξιολόγησης

Καθώς το μοντέλο εκπαιδεύεται, είναι απαραίτητο να αξιολογήσετε την απόδοσή του χρησιμοποιώντας διάφορες μετρήσεις.

3.3.1 Πίνακας σύγχυσης

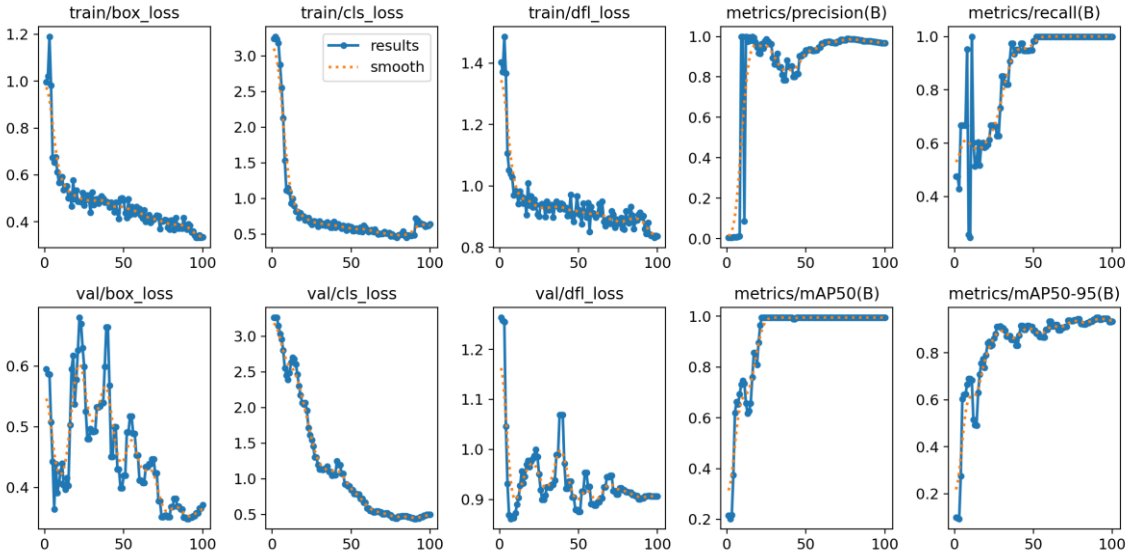
Αυτός ο πίνακας βοηθά στην κατανόηση του πόσο καλά το μοντέλο διακρίνει μεταξύ διαφορετικών κλάσεων. Δείχνει τον αριθμό των αληθινών θετικών, των ψευδώς θετικών, των αληθινών αρνητικών και των ψευδώς αρνητικών για κάθε τάξη.



εικόνα 8. Πίνακας συγχυσης του μοοντέλου.

3.3.2 Γραφήματα παρτίδας εκπαίδευσης

Αυτά τα γραφήματα απεικονίζουν πώς αλλάζει η απώλεια και η ακρίβεια του μοντέλου σε κάθε παρτίδα δεδομένων εκπαίδευσης. Βοηθούν στη διάγνωση προβλημάτων όπως η υπερπροσαρμογή ή η κακή προσαρμογή.



εικόνα 9. Γραφήματα παρτίδας εκπαίδευσης

3.3.3 Ακρίβεια επικύρωσης

Μετά από κάθε εποχή, το μοντέλο αξιολογείται στο σύνολο δεδομένων επικύρωσης για να μετρηθεί η απόδοσή του σε αόρατα δεδομένα. Αυτό διασφαλίζει ότι το μοντέλο γενικεύεται πολύ πέρα από το σύνολο εκπαίδευσης.

epoch,	train/box_loss,	train/cls_loss,	train/dfl_loss,	metrics/precis
1,	0,	87.511,	0,	0
2,	0,	87.205,	0,	0
3,	0,	86.755,	0,	0
4,	0,	86.171,	0,	0
5,	0,	85.144,	0,	0
6,	0,	83.054,	0,	0
7,	0,	81.757,	0,	0
8,	0,	80.119,	0,	0
9,	0,	78.097,	0,	0
10,	0,	75.861,	0,	0
11,	0,	73.343,	0,	0
12,	0,	70.475,	0,	0
13,	0,	69.937,	0,	0
14,	0,	68.082,	0,	0
15,	0,	66.452,	0,	0
16,	0,	64.788,	0,	0
17,	0,	65.33,	0,	0
18,	0,	64.208,	0,	0
19,	0,	63.145,	0,	0
20,	0,	62.165,	0,	0
21,	0,	61.289,	0,	0
22,	0,	60.992,	0,	0
23,	0,	60.167,	0,	0
24,	0,	59.129,	0,	0
25,	0,	58.164,	0,	0
26,	0,	57.596,	0,	0

εικόνα 10.

4. Βαθμολόγηση, Λογική και Εφαρμογή

Μόλις γίνει επεξεργασία του φύλλου απαντήσεων και εντοπιστούν οι περιοχές που περιέχουν απαντήσεις, η εικόνα χωρίζεται σε μικρότερα τμήματα που αντιστοιχούν σε μεμονωμένες ερωτήσεις και επιλογές απαντήσεων.

4.1 Διαίρεση εικόνων: np.vsplit και np.hsplit

Αυτό γίνεται χρησιμοποιώντας τις συναρτήσεις NumPy np.vsplit και np.hsplit.. Για περαιτέρω αναφορά, ανατρέξτε στο Guide to NumPy από τον Travis E. Oliphant, PhD[2].

4.1.1 np.vsplit()

Διαχωρίζει την εικόνα κάθετα σε ίσα μέρη. Αυτό είναι χρήσιμο για το διαχωρισμό σειρών ερωτήσεων.

4.1.2 np.hsplit()

Διαχωρίζει την εικόνα οριζόντια σε ίσα μέρη. Αυτό χρησιμοποιείται για τον διαχωρισμό των επιμέρους επιλογών απάντησης (Α, Β, Γ, Δ) σε κάθε ερώτηση.

Για παράδειγμα, εάν έχουμε ένα φύλλο απαντήσεων με 10 ερωτήσεις και 4 επιλογές απαντήσεων, η np.vsplit() μπορεί να χωρίσει την εικόνα σε 10 μέρη (το καθένα αντιπροσωπεύει μία ερώτηση) και στη συνέχεια η np.hsplit() μπορεί περαιτέρω να διαιρέσει κάθε ένα από αυτά τα μέρη σε 4 (το καθένα αντιπροσωπεύει μια επιλογή απάντησης).

4.2 Ανάλυση εικονοστοιχείων και ανίχνευση απαντήσεων

Μόλις οι εικόνες χωριστούν σε μεμονωμένες περιοχές απαντήσεων, το επόμενο βήμα είναι να αναλύσετε τις τιμές των εικονοστοιχείων για να εντοπίσετε ποια απάντηση έχει επισημανθεί. Αυτό γίνεται μετρώντας τον αριθμό των μη μηδενικών pixel (που υποδεικνύουν μαύρες ή γεμάτες περιοχές) σε κάθε περιοχή χρησιμοποιώντας τη συνάρτηση OpenCV cv2.countNonZero().

Η απάντηση με τον υψηλότερο αριθμό pixel σε μια δεδομένη ερώτηση θεωρείται η επιλεγμένη απάντηση. Αυτό συμβαίνει επειδή οι γεμάτοι κύκλοι απαντήσεων θα έχουν περισσότερα μη μηδενικά pixel σε σύγκριση με τις μη επισημασμένες επιλογές.

εικόνα 11. Ελεγχος αριθμού λευκών pixel

4.3 Διαδικασία βαθμολόγησης

Η λογική βαθμολόγησης λειτουργεί συγκρίνοντας τις απαντήσεις που εντοπίστηκαν (με βάση την ανάλυση εικονοστοιχείων) με τη σωστή αντιστοιχία των απαντήσεων. Κάθε σωστή αντιστοίχιση κερδίζει έναν βαθμό, ενώ οι λανθασμένες απαντήσεις δεν λαμβάνουν πόντους.

Για παράδειγμα, εάν οι σωστές απαντήσεις για ένα τεστ 10 ερωτήσεων είναι [A, B, C, D, A, B, C, D, A, D] και οι απαντήσεις που εντοπίστηκαν ταιριάζουν με 8 στις 10, η τελική βαθμολογία θα είναι 8/10. Στη συνέχεια, το σύστημα εμφανίζει τον τελικό βαθμό μαζί με μια οπτική επικάλυψη στο φύλλο απαντήσεων που υποδεικνύει τις σωστές και τις λανθασμένες απαντήσεις., ανατρέξτε στο βίντεο του Εργαστηρίου Murtaza[1]

Συμπεράσματα

Το μοντέλο YOLOv8 επέδειξε υψηλή ακρίβεια στην ανίχνευση απαντήσεων και τη βαθμολόγηση των δοκιμών. Τα αποτελέσματα συγκρίθηκαν με τη χειροκίνητη βαθμολόγηση, δείχνοντας ότι το αυτοματοποιημένο σύστημα είναι ταχύτερο και πιο συνεπές. Ωστόσο, ορισμένες προκλήσεις, όπως οι διαφορετικές συνθήκες φωτισμού και τα θολά σημάδια, δημιούργησαν δυσκολίες κατά την ανίχνευση.

Μελλοντικές βελτιώσεις

Οι μελλοντικές εργασίες θα περιλαμβάνουν την επέκταση του συνόλου δεδομένων, τη βελτίωση της ευρωστίας του μοντέλου και την ενσωμάτωση πρόσθετων λειτουργιών όπως η αναγνώριση χειρόγραφου κειμένου για σύντομες απαντήσεις.

Βιβλιογραφία

[1] Murtaza's Workshop - Robotics and AI. (n.d.). Optical Mark Recognition (OMR) MCQ Automated Grading - OpenCV Python. [YouTube video]. Retrieved from https://www.youtube.com/watch?v=0IqCOPlGBTs

[2] Oliphant, T. E. (2006). Guide to NumPy.

[3] Downey, A. (n.d.). Think Python: How to Think Like a Computer Scientist (2nd ed., Version 2.4.0).

[4] Ultralytics. (n.d.). YOLOv8 Documentation (Date:08/2024) . Retrieved from https://docs.ultralytics.com

Παράρτημα Κώδικα

Main.py

```
import cv2
import numpy as np
from ultralytics import YOLO
import functions

# Load the YOLOv8 model
model = YOLO('best.pt')

# Load the image
image_path = '1.jpg'
imgW = 640
imgH = 640
img = cv2.imread(image_path)
Answers = 4
Questions = 10
Correct = [0, 1, 2, 3, 0, 1, 2, 3, 0, 3]
WebCam = True
cameraNo = 0


capture = cv2.VideoCapture(cameraNo)
capture.set(10, 150)

while True:
        if WebCam: success,  img = capture.read()
        else:
        img = cv2.imread(image_path)
```

```
        # Perform detection
        results = model(img)
        bboxes = functions.get_bounding_boxes(results)

        # Initialize variables
        answers_img = None



        # Extract regions
        for bbox in bboxes:
        x1, y1, x2, y2, cls = bbox
        if cls == 1:  # Assuming class 1 is Answers
        answers_img = img[y1:y2, x1:x2]
        break  # Assuming we want the first occurrence

        # Display and save original detected answers
        if answers_img is not None:
        cv2.imshow('Original Answers', answers_img)
        functions.save_image(answers_img, 'output', 'answers_detected.jpg')

        # Preprocess the detected answers region
        answers_img = cv2.resize(answers_img,(imgW, imgH))
        AnsCopy = answers_img.copy()
        AnsLcopy = answers_img.copy()
        preprocessed_img = functions.preprocess_image(AnsCopy)



        try:
        #find Countours
        countours , hierarchy =
cv2.findContours(preprocessed_img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
        cv2.drawContours(AnsCopy,countours, -1, (0, 255, 0), 10)
```

```
rectCon = functions.rectCountout(countours)

largestC = functions.getCornerPoints(rectCon[0])



# Display and save preprocessed image
cv2.imshow('Preprocessed Answers', preprocessed_img)
functions.save_image(preprocessed_img, 'output', 'answers_preprocessed.jpg')


if largestC.size != 0 :
cv2.drawContours(AnsLcopy, largestC, -1 , (0, 255, 0), 20)


functions.reorder(largestC)

pt1 = np.float32(largestC)
pt2 = np.float32([[0,0], [0, imgH], [imgW, imgH], [imgW , 0]])
matrix = cv2.getPerspectiveTransform(pt1, pt2)
imgWarpC = cv2.warpPerspective(AnsLcopy, matrix, (imgW, imgH))


#Threshold
imgWarpG= cv2.cvtColor(imgWarpC, cv2.COLOR_BGR2GRAY)
ThreshImg = cv2.threshold(imgWarpG, 175, 355,  cv2.THRESH_BINARY_INV)[1]

functions.BoxSplit(ThreshImg)

boxes = functions.BoxSplit(ThreshImg)
#cv2.imshow("Test", boxes[0])
```

```
#Values of pixels
ValPixel = np.zeros((Questions, Answers))
countCol = 0
countRow = 0

for Image in boxes:
PixelTotal = cv2.countNonZero(Image)
ValPixel[countRow][countCol] = PixelTotal
countCol += 1
if countCol == Answers:
        countRow += 1
        countCol = 0

#print(ValPixel)
#Finding Markings
Index = []
for x in range(0, Questions) :
array = ValPixel[x]
ValIndex = np.where(array == np.amax(array))
Index.append(ValIndex[0][0])
#print(Index)

#Grade
grade = []
for x in range(0, Questions):
if Correct[x] == Index[x]:
        grade.append(1)
else:
        grade.append(0)
```

```
FinalScore = sum(grade)
print(FinalScore)



#Correct Answers
#Results = imgWarpC.copy()
#Results = functions.CorrectAnswers(Results, Index, grade, Correct, Questions, Answers)

# Display Final Score on Source Image
final_image = img.copy()
#final_image = cv2.resize(imgW, imgH)
font = cv2.FONT_HERSHEY_SIMPLEX
text = f"Score: {FinalScore}"
text_size = cv2.getTextSize(text, font, 1, 2)[0]
text_x = final_image.shape[1] - text_size[0] - 10
text_y = final_image.shape[0] - 10
cv2.putText(final_image, text, (text_x, text_y), font, 1, (0, 255, 0), 2)
cv2.imshow("Final Image", final_image)

ArrayImg = ([answers_img, preprocessed_img, AnsCopy, AnsLcopy],
            [imgWarpC, ThreshImg, ThreshImg, ThreshImg])

except:
ArrayImg = ([answers_img, answers_img, answers_img, answers_img],
            [answers_img, answers_img, answers_img, answers_img])
StackedImg = functions.stackImages(ArrayImg, 0.5)




cv2.imshow("stacked", StackedImg)
if cv2.waitKey(1) & 0xFF == ord('s'):
```

```
        cv2.imwrite("finalResult.jpg", final_image )
        cv2.waitKey(300)

        else:
        print("No 'Answers' detected in the image.")
```

Functions.py

```
import cv2
import numpy as np
import os




def stackImages(imgArray,scale,lables=[]):
        rows = len(imgArray)
        cols = len(imgArray[0])
        rowsAvailable = isinstance(imgArray[0], list)
        width = imgArray[0][0].shape[1]
        height = imgArray[0][0].shape[0]
        if rowsAvailable:
        for x in range ( 0, rows):
        for y in range(0, cols):
        imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)
        if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y],
cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
        hor[x] = np.hstack(imgArray[x])
        hor_con[x] = np.concatenate(imgArray[x])
        ver = np.vstack(hor)
        ver_con = np.concatenate(hor)
        else:
        for x in range(0, rows):
        imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
        if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x],
cv2.COLOR_GRAY2BGR)
```

```
        hor= np.hstack(imgArray)
        hor_con= np.concatenate(imgArray)
        ver = hor
        if len(lables) != 0:
        eachImgWidth= int(ver.shape[1] / cols)
        eachImgHeight = int(ver.shape[0] / rows)
        #print(eachImgHeight)
        for d in range(0, rows):
        for c in range (0,cols):

        cv2.rectangle(ver,(c*eachImgWidth,eachImgHeight*d),(c*eachImgWidth+len(lables[d][c
])*13+27,30+eachImgHeight*d),(255,255,255),cv2.FILLED)

        cv2.putText(ver,lables[d][c],(eachImgWidth*c+10,eachImgHeight*d+20),cv2.FONT_HERS
HEY_COMPLEX,0.7,(255,0,255),2)
        return ver




def get_bounding_boxes(results):
        bboxes = []
        for box in results[0].boxes:
        xyxy = box.xyxy.tolist()[0]  # Get bounding box coordinates
        x1, y1, x2, y2 = map(int, xyxy[:4])
        cls = int(box.cls.tolist()[0])  # Get class label
        bboxes.append((x1, y1, x2, y2, cls))
        return bboxes



def save_image(img, folder, name):
        if not os.path.exists(folder):
```

```
        os.makedirs(folder)
        cv2.imwrite(os.path.join(folder, name), img)


def preprocess_image(img):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        blurred = cv2.GaussianBlur(gray, (5, 5), 0)
        edged = cv2.Canny(blurred, 10, 50)
        return edged


def detect_rectangles(edged_img):
        contours, _ = cv2.findContours(edged_img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        rectangles = []

        for cnt in contours:
        approx = cv2.approxPolyDP(cnt, 0.02 * cv2.arcLength(cnt, True), True)
        if len(approx) == 4:
        x, y, w, h = cv2.boundingRect(cnt)
        rectangles.append((x, y, w, h))

        return rectangles


def draw_rectangles(img, rectangles):
        for (x, y, w, h) in rectangles:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
        return img


def rectCountout(countours):
```

```
        rectCon = []
        for i in countours:
        area = cv2.contourArea(i)
        #print(area)
        if area > 100:
        perimeter = cv2.arcLength(i, True)
        approx = cv2.approxPolyDP(i, 0.02*perimeter, True)
        #print("corner points", len(approx))
        if len(approx) == 4:
        rectCon.append(i)
        rectCon = sorted(rectCon, key = cv2.contourArea, reverse = True)
        return rectCon

def getCornerPoints(cont):
        peri = cv2.arcLength(cont, True)
        approx = cv2.approxPolyDP(cont, 0.02*peri, True)
        return approx


def reorder(points):
        points= points.reshape((4, 2))
        pointsNew = np.zeros((4, 1, 2), np.int32)
        add = points.sum(1)
        #print(add)
        pointsNew[0] = points[np.argmin(add)]
        pointsNew[3] = points[np.argmax(add)]
        diff = np.diff(points, axis=1)
        pointsNew[1] = points[np.argmin(diff)]
        pointsNew[2] = points[np.argmax(diff)]
        #print(diff)
```

```
def BoxSplit(img):
        BoxRows = np.vsplit(img, 10)
        Box = []
        for r in BoxRows:
        BoxCols = np.hsplit(r, 4)
        for box in BoxCols:
        Box.append(box)

        return Box

#def CorrectAnswers(img, Index, grade, Correct, Questions, Answers):
        #SectionWeight = img.shape[1] / Questions
        #SectionHeight = img.shape[0] / Answers
        #for x in range(Questions):
        #Answer = Index[x]
        #CorrectX = int((Answer * SectionWeight) + SectionWeight // 2)
        #CorrectY = int((x * SectionHeight) + SectionHeight // 2)
        #cv2.circle(img, (CorrectX, CorrectY), 50, (0, 255, 0), cv2.FILLED)
        #return img
```