



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Δημιουργία συστήματος που αξιοποιεί νευρωνικά δίκτυα
για την ανίχνευση εικόνων που παράγονται από τεχνητή
νοημοσύνη και κατηγοριοποίηση πραγματικών εικόνων**

Διπλωματική Εργασία

Δημοσθένης Μπουρουτζήκας

Επιβλέπων: Γεράσιμος Ποταμιάνος

Σεπτέμβριος 2025



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

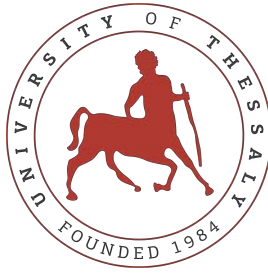
**Δημιουργία συστήματος που αξιοποιεί νευρωνικά δίκτυα
για την ανίχνευση εικόνων που παράγονται από τεχνητή
νοημοσύνη και κατηγοριοποίηση πραγματικών εικόνων**

Διπλωματική Εργασία

Δημοσθένης Μπουρουτζήκας

Επιβλέπων: Γεράσιμος Ποταμιάνος

Σεπτέμβριος 2025



UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Development of a system using neural networks for the
detection of AI-generated images and the classification of
real images**

Diploma Thesis

Dimosthenis Bouroutzikas

Supervisor: Gerasimos Potamianos

September 2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Γεράσιμος Ποταμιάνος**

Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και
Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Δημήτριος Κατσαρός**

Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και
Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Μιχαήλ Βασιλακόπουλος**

Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπο-
λογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, Αναπλ. Καθ. Γεράσιμο Ποταμιάνο, για την καθοδήγησή του και την υποστήριξή του σε όλη την διάρκεια της εκπόνησης αυτής της Διπλωματικής Εργασίας. Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου που με υποστήριξαν σε όλη την διάρκεια των σπουδών μου.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Δημοσθένης Μπουρουτζήκας

Διπλωματική Εργασία

Δημιουργία συστήματος που αξιοποιεί νευρωνικά δίκτυα για την ανίχνευση εικόνων που παράγονται από τεχνητή νοημοσύνη και κατηγοριοποίηση πραγματικών εικόνων

Δημοσθένης Μπουρουτζήκας

Περίληψη

Σε αυτή την διπλωματική εργασία παρουσιάζεται η δημιουργία ενός συστήματος ταξινόμησης εικόνων, με στόχο την διάκριση μεταξύ αληθινών εικόνων και εικόνων από γενετική τεχνητή νοημοσύνη και την εύρεση της θεματολογίας της εικόνας. Για την επίτευξη του στόχου, αναπτύσσεται και εκπαιδεύεται ένα νευρωνικό δίκτυο δύο κεφαλών εξόδου, στηριζόμενο σε κοινό υπόβαθρο αρχιτεκτονικής ενός ειδικού μοντέλου ταξινόμησης, του ResNet. Για να επιβεβαιωθεί η αποτελεσματικότητα της προσέγγισης αυτής, διεξάγονται τρία συμπληρωματικά πειράματα. Στο πρώτο γίνεται μελέτη της ικανότητας ταξινόμησης του μοντέλου σε εικόνες με διαφορετικές παραμορφώσεις θορύβου, ενώ στο δεύτερο συγκρίνεται με συστήματα που αξιοποιούν παραδοσιακές τεχνικές μηχανικής μάθησης. Τέλος, στο τρίτο μελετάται η απόδοση πολλών υβριδικών μοντέλων, που χρησιμοποιούν τον χώρο των χαρακτηριστικών του νευρωνικού δικτύου σε συνδυασμό με τεχνικές μηχανικής μάθησης. Από τα πειράματα επιβεβαιώνεται η υψηλή απόδοση του μοντέλου, που συγκρίνεται με συστήματα αιχμής.

Λέξεις-κλειδιά:

ταξινόμηση εικόνων, συνελκτικά νευρωνικά δίκτυα, βαθιά μάθηση, όραση υπολογιστών, αναγνώριση προτύπων, μηχανική μάθηση, επεξεργασία εικόνας

Diploma Thesis

Development of a system using neural networks for the detection of AI-generated images and the classification of real images

Dimosthenis Bouroutzikas

Abstract

In this thesis, the development of a dual task image classification system is presented. On one hand, the system detects the source of the images, whether they are real or AI generated, and on the other, it classifies them based on the object they depict. To this end, a neural network with two distinct output heads with a shared backbone of a ResNet model is initialized and trained. In order to verify the effectiveness of the proposed approach, three more experiments are conducted. The first tests the robustness to noise and image corruptions, whereas the second pits the network against models based on various machine learning techniques and compares their ability to correctly classify images. In the final experiment, the classification performance of hybrid systems is assessed. Those hybrid systems, utilize the feature space of the original neural network and then apply machine learning techniques on it. The experiments demonstrate that the proposed method achieves state-of-the-art results.

Keywords:

image classification, convolutional neural networks, deep learning, computer vision, pattern recognition, machine learning, image processing

Πίνακας περιεχομένων

Ευχαριστίες	ix
Περίληψη	xii
Abstract	xiii
Πίνακας περιεχομένων	xv
Κατάλογος σχημάτων	xix
Κατάλογος πινάκων	xxix
Συντομογραφίες	xxxi
1 Εισαγωγή	1
1.1 Αντικείμενο της διπλωματικής	2
1.2 Συνεισφορά	2
1.3 Οργάνωση του τόμου	3
2 Βιβλιογραφική ανασκόπηση	5
2.1 Είδη παραγωγικής τεχνητής νοημοσύνης	5
2.1.1 Παραγωγικά Αντιπαραθετικά Δίκτυα	5
2.1.2 Μοντέλα διάχυσης	7
2.1.3 Μοντέλα μετασχηματιστές	8
2.2 Μέθοδοι ταξινόμησης εικόνων	9
2.2.1 Ταξινόμηση με χρήση μηχανικής μάθησης	11
2.2.2 Ταξινόμηση με CNN και μετασχηματιστή ViT	13
2.2.3 Ταξινόμηση βάσει ειδικών χαρακτηριστικών εικόνων	17

3	Θεωρητικό υπόβαθρο	21
3.1	Συνελικτικά νευρωνικά δίκτυα	21
3.2	Μοντέλο ResNet50	22
3.2.1	Υπολειμματική μάθηση	23
3.2.2	Δομή ResNet50	24
3.3	Μηχανική μάθηση για ταξινόμηση εικόνων	26
3.3.1	Μηχανές διανυσματικής υποστήριξης	26
3.3.2	Random Forest ταξινομητές	29
3.3.3	XGBoost ταξινομητές	30
4	Προετοιμασία δεδομένων	31
4.1	Βάση δεδομένων CIFAKE	31
4.2	Διαχειριστής δεδομένων CIFAKEDatasetHandler	35
4.3	Μετασχηματισμοί, σύνολα δεδομένων και dataloaders	37
5	Μοντέλο ResNet50Classifier	39
5.1	Αντικείμενο ResNet50Classifier	39
5.1.1	Συνάρτηση <code>__init__</code>	39
5.1.2	Συνάρτηση <code>__forward__</code>	41
5.2	Εκπαίδευση μοντέλου	42
5.2.1	Αλγόριθμος εκπαίδευσης	42
5.2.2	Μετρικές κατά την εκπαίδευση και σημασία “ξεπαγώματος”	47
5.3	Έλεγχος μοντέλου	51
5.3.1	Αλγόριθμος ελέγχου	51
5.3.2	Μετρικές απωλειών και ακρίβειας στον έλεγχο	51
5.4	Δοκιμή μοντέλου	54
5.4.1	Αλγόριθμος δοκιμής	54
5.4.2	Συνάρτηση υπολογισμού μετρικών <code>__compute_metrics__</code>	54
5.4.3	Μετρικές ανά κλάση και συνάρτηση <code>__plot_creation__</code>	57
5.4.4	Υπολογισμός κοινών μετρικών	59
6	Πειραματικά αποτελέσματα	61
6.1	Αποτελέσματα κεφαλών	61
6.2	Αποτελέσματα συνδυασμών κλάσεων	63

7	Αντοχή μοντέλου σε εικόνες με θόρυβο	67
7.1	Είδη θορύβου υπό εξέταση	67
7.1.1	Αντικείμενο GaussianNoise	70
7.1.2	Αντικείμενο JPEGCompression	71
7.2	Δημιουργία εικόνων με θόρυβο	72
7.2.1	Μελέτη συναρτήσεων	72
7.2.2	Διαχειριστής δεδομένων CIFAKENoisyDatasetHandler	75
7.2.3	Δημιουργία datasets και dataloaders	76
7.3	Εκπαίδευση και έλεγχος μοντέλων	76
7.3.1	Εκπαίδευση και έλεγχος του full_noise_model	77
7.3.2	Εκπαίδευση και έλεγχος του both_noise_model	77
7.3.3	Εκπαίδευση και έλεγχος του jpeg_model	77
7.3.4	Εκπαίδευση και έλεγχος του gaussian_model	81
7.3.5	Συνολική συγκριτική αξιολόγηση εκπαίδευσης και ελέγχου	81
7.4	Ανάλυση δοκιμής των μοντέλων	86
7.4.1	Δοκιμή clean_model και γραφήματα	86
7.4.2	Ανάλυση δοκιμής μοντέλου full_noise_model υπό διαφορετικές συν- θήκες θορύβου	90
7.4.3	Ανάλυση δοκιμής μοντέλου both_noise_model	91
7.4.4	Ανάλυσης δοκιμής μοντέλου jpeg_model	95
7.4.5	Ανάλυση δοκιμής μοντέλου gaussian_noise_model	102
7.5	Σύγκριση απόδοσης μοντέλων ανά συνθήκη θορύβου	103
8	Έρευνα απόδοσης ResNet50Classifier σε σχέση με παραδοσιακές τεχνικές μηχανικής μάθησης	109
8.1	Ανάπτυξη μοντέλων ML και εκπαίδευσή τους απευθείας στις εικόνες	109
8.1.1	Εξαγωγή δεδομένων από εικόνες και κανονικοποίηση διανύσματος δεδομένων	110
8.1.2	Χρήση linear SVM για το πρώτο στάδιο του πειράματος	112
8.1.3	Χρήση Random Forest για το πρώτο στάδιο του πειράματος	115
8.1.4	Χρήση XGBoost για το πρώτο στάδιο του πειράματος	115
8.2	Χρήση χαρακτηριστικών απευθείας από το CNN για εκπαίδευση ML μοντέλων	119
8.2.1	Εξαγωγή χαρακτηριστικών	119

8.2.2	RBF kernel SVM	120
8.2.3	Linear SVM	121
8.2.4	Random Forest Classifier	124
8.2.5	XGBoost Classifier	124
9	Συμπεράσματα	129
9.1	Σύνοψη και συμπεράσματα	129
9.2	Μελλοντικές επεκτάσεις	129
	Βιβλιογραφία	131
	ΠΑΡΑΡΤΗΜΑΤΑ	137
A	Αποτελέσματα έρευνας αντοχής του μοντέλου σε θόρυβο	139
A.1	Πίνακες αποτελεσμάτων	140
A.2	Γραφήματα έρευνας αντοχής στον θόρυβο	145
A.2.1	Γραφήματα μοντέλου “clean_model”	145
A.2.2	Γραφήματα μοντέλου “full_noise_model”	149
A.2.3	Γραφήματα μοντέλου “both_noise_model”	154
A.2.4	Γραφήματα μοντέλου “jpeg_model”	159
A.2.5	Γραφήματα μοντέλου “gaussian_model”	164
B	Προγραμματιστική υλοποίηση	169

Κατάλογος σχημάτων

2.1	Αναπαράσταση της δομής ενός Παραγωγικού Αντιπαραθετικού Δικτύου GAN (σχήμα προσαρμοσμένο από [1]).	7
2.2	Η δομή ενός μετασχηματιστή και ανάλυση της εσωτερικής λειτουργίας του κωδικοποιητή και του αποκωδικοποιητή του (σχήμα από [2]).	10
3.1	Η δομή ενός στρώματος υπολειμματικής μάθησης residual block (σχήμα προσαρμοσμένο από [3]).	24
3.2	Η δομή του ResNet50. Το a) περιγράφει την επεξεργασία της εικόνας κατά την είσοδο. Το b) περιγράφει την συνολική δομή του ResNet50 και τα σύμβολα $\times 2$, $\times 3$, $\times 5$ αντιπροσωπεύουν τον αριθμό των στρωμάτων. Το c) δείχνει την αρχιτεκτονική των convolutional blocks όπου οι διαστάσεις εισόδου διαφέρουν και το d) δείχνει την αρχιτεκτονική των identity blocks όπου οι διαστάσεις εισόδου είναι σταθερές (σχήμα από [4]).	25
4.1	Στο σχήμα αυτό παρουσιάζεται ένα παράδειγμα από κάθε δυνατό συνδυασμό κλάσεων για τη ΒΔ CIFAKE[5, 6]. Οι εικόνες είναι μεγέθους 32×32 και επομένως χαμηλής ποιότητας.	32
5.1	Η δομή του ResNet50Classifier	41
5.2	Το γράφημα των απωλειών εκπαίδευσης και ελέγχου	49
5.3	Τα accuracy κάθε κεφαλής ανά εποχή κατά την εκπαίδευση	50
5.4	Η ακρίβεια ανά κεφαλή και η κοινή ακρίβεια ελέγχου	53
6.1	Οι μετρικές precision, recall, f1 score της πρώτης κεφαλής αναλυτικά για κάθε κλάση και οι μέσοι όροι μαζί με μέση ακρίβεια.	62
6.2	Confusion matrix πρώτης κεφαλής	62

6.3	Οι μετρικές precision, recall, f1 score της δεύτερης κεφαλής αναλυτικά για κάθε κλάση και οι μέσοι όροι μαζί με μέση ακρίβεια.	63
6.4	Confusion Matrix για την δεύτερη κεφαλή	64
6.5	Οι κοινές μετρικές και οι μετρικές για κάθε συνδυασμό ετικετών των δύο κεφαλών	65
7.1	Παρουσιάζονται οι αρχικές εικόνες μαζί με τις αντίστοιχες παραμορφωμένες εικόνες (corrupted). Στις πρώτες 2 σειρές προστίθεται μόνο κανονικός θόρυβος, στις επόμενες 2 γίνεται μόνο JPEG Compression και στην τελευταία σειρά προστίθενται και τα 2 είδη θορύβου.	73
7.2	Οι απώλειες του full_noise_model	78
7.3	Οι ακρίβειες εκπαίδευσης του full_noise_model	78
7.4	Οι ακρίβειες ελέγχου του full_noise_model	79
7.5	Οι απώλειες του both_noise_model	79
7.6	Οι ακρίβειες εκπαίδευσης του both_noise_model	80
7.7	Οι ακρίβειες ελέγχου του both_noise_model	80
7.8	Οι απώλειες του jpeg_model	81
7.9	Οι ακρίβειες εκπαίδευσης του jpeg_model	82
7.10	Οι ακρίβειες ελέγχου του jpeg_model	82
7.11	Οι απώλειες του gaussian_model	83
7.12	Οι ακρίβειες εκπαίδευσης του gaussian_model	83
7.13	Οι ακρίβειες ελέγχου του gaussian_model	84
7.14	Οι απώλειες όλων των μοντέλων	84
7.15	Οι ακρίβειες εκπαίδευσης όλων των μοντέλων	85
7.16	Οι ακρίβειες ελέγχου όλων των μοντέλων	85
7.17	Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο “full”	88
7.18	Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο “both”	88
7.19	Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο “jpeg”	89
7.20	Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με Gaussian noise	89

7.21 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων του μοντέλου για κάθε περίπτωση	90
7.22 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”	92
7.23 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”	92
7.24 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”	93
7.25 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”	93
7.26 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”	94
7.27 Οι κοινές μετρικές του μοντέλου στα διαφορετικά σενάρια δοκιμής	94
7.28 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”	95
7.29 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”	96
7.30 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”	96
7.31 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”	97
7.32 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”	97
7.33 Οι κοινές μετρικές για το μοντέλο σε κάθε σετ δοκιμής	98
7.34 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”	99
7.35 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”	99
7.36 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”	100
7.37 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”	100
7.38 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”	101
7.39 Οι κοινές μετρικές του μοντέλου ανά σενάριο δοκιμής	101
7.40 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”	103
7.41 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”	104
7.42 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”	104
7.43 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”	105
7.44 Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”	105
7.45 Οι κοινές μετρικές ανά σετ δοκιμής	106
7.46 Η απόδοση μοντέλων για το dataset “clean”	106
7.47 Η απόδοση μοντέλων για το dataset “full”	107
7.48 Η απόδοση μοντέλων για το dataset “both”	107
7.49 Η απόδοση μοντέλων για το dataset “jpeg”	108
7.50 Η απόδοση μοντέλων για το dataset “gaussian”	108

8.1	Οι μετρικές για το binary linear SVM εκπαιδευμένο απευθείας στον χώρο των εικόνων.	113
8.2	Το confusion matrix για το binary linear SVM.	113
8.3	Οι μετρικές για το multiclass linear SVM εκπαιδευμένο απευθείας στον χώρο των εικόνων.	114
8.4	Το confusion matrix για το multiclass linear SVM.	114
8.5	Οι μετρικές για το binary Random Forest εκπαιδευμένο απευθείας στον χώρο των εικόνων.	115
8.6	Το confusion matrix για το binary Random Forest.	116
8.7	Οι μετρικές για το multiclass Random Forest εκπαιδευμένο απευθείας στον χώρο των εικόνων.	116
8.8	Το confusion matrix για το multiclass Random Forest.	117
8.9	Οι μετρικές για το binary XGBoost εκπαιδευμένο απευθείας στον χώρο των εικόνων.	117
8.10	Το confusion matrix για το binary XGBoost.	118
8.11	Οι μετρικές για το multiclass XGBoost εκπαιδευμένο απευθείας στον χώρο των εικόνων.	118
8.12	Το confusion matrix για το multiclass XGBoost.	119
8.13	Οι μετρικές για το binary RBF SVM εκπαιδευμένο σε ResNet50Classifier features.	120
8.14	Το confusion matrix για το binary RBF SVM.	121
8.15	Οι μετρικές για το multiclass RBF SVM εκπαιδευμένο σε ResNet50Classifier features.	121
8.16	Το confusion matrix για το RBF SVM.	122
8.17	Οι μετρικές για το binary Linear SVM εκπαιδευμένο σε ResNet50Classifier features.	122
8.18	Το confusion matrix για το binary Linear SVM.	123
8.19	Οι μετρικές για το multiclass Linear SVM εκπαιδευμένο σε ResNet50Classifier features.	123
8.20	Το confusion matrix για το Linear SVM.	124
8.21	Οι μετρικές για το binary Random Forest εκπαιδευμένο σε ResNet50Classifier features.	125

8.22	To confusion matrix για το binary Random Forest.	125
8.23	Οι μετρικές για το multiclass Random Forest εκπαιδευμένο σε ResNet50Classifier features.	126
8.24	To confusion matrix για το Random Forest.	126
8.25	Οι μετρικές για το binary XGBoost εκπαιδευμένο σε ResNet50Classifier features.	127
8.26	To confusion matrix για το binary XGBoost.	127
8.27	Οι μετρικές για το multiclass XGBoost εκπαιδευμένο σε ResNet50Classifier features.	128
8.28	To confusion matrix για το XGBoost.	128
A.1	Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου	145
A.2	Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου	145
A.3	Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου, χωρίς να συνυπάρχουν σε εικόνες και τα δύο είδη θορύβου	146
A.4	Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου, χωρίς να συνυπάρχουν σε εικόνες και τα δύο είδη θορύβου	146
A.5	Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.	147
A.6	Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.	147
A.7	Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.	148
A.8	Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.	148
A.9	Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε καθαρά δεδομένα.	149
A.10	Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε καθαρά δεδομένα.	149

A.11 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα.	150
A.12 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα	150
A.13 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες που έχουν έως ένα είδος θορύβου.	151
A.14 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες που έχουν έως ένα είδος θορύβου	151
A.15 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.	152
A.16 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.	152
A.17 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με κανονικό θόρυβο.	153
A.18 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με κανονικό θόρυβο.	153
A.19 Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα χωρίς θόρυβο.	154
A.20 Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα χωρίς θόρυβο.	154
A.21 Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου.	155
A.22 Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου.	155
A.23 Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα.	156

A.24 Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου	156
A.25 Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.	157
A.26 Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.	157
A.27 Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.	158
A.28 Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.	158
A.29 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες χωρίς θόρυβο. . .	159
A.30 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.	159
A.31 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.	160
A.32 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.	160
A.33 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.	161
A.34 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.	161
A.35 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε στα αντίστοιχα δεδομένα. . .	162
A.36 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε στα αντίστοιχα δεδομένα.	162

A.37 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.	163
A.38 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο	163
A.39 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.	164
A.40 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.	164
A.41 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.	165
A.42 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.	165
A.43 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.	166
A.44 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.	166
A.45 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.	167
A.46 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.	167
A.47 Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε αντίστοιχες εικόνες.	168
A.48 Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε αντίστοιχες εικόνες.	168
B.1 Οι κλάσεις	169
B.2 Η υλοποίηση των συναρτήσεων __init__, __len__ και __getitem__ του διαχειριστή δεδομένων CIFAKEDatasetHandler	170
B.3 Ο καθορισμός των βαρών	171
B.4 Οι λίστες μετασχηματισμών	171

B.5 Τα σύνολα δεδομένων και οι dataloaders. Τα train_dataset_ML και test_dataset_ML είναι σύνολα δεδομένων απαραίτητα για το τρίτο πείραμα, αλλά όχι για το συγκεκριμένο.	172
B.6 Ο ορισμός του αντικειμένου ResNet50Classifier και οι συναρτήσεις __init__ και __forward__.	173
B.7 Η αρχική προσαρμογή των υπερπαραμέτρων εκπαίδευσης	174
B.8 Η συνάρτηση __compute_metrics__	174
B.9 Η συνάρτηση __plot_creation__	175
B.10 Το αντικείμενο GaussianNoise	175
B.11 Το αντικείμενο JPEGCompression	176
B.12 Η συνάρτηση gen_encoding_map	177
B.13 Οι βασικοί μετασχηματισμοί ανεξαρτήτως encoding	178
B.14 Η συνάρτηση decode_transform	179
B.15 Η συνάρτηση create_transforms_cache	179
B.16 Ο διαχειριστής δεδομένων CIFAKENoisyDatasetHandler	180
B.17 Δημιουργία συνόλων με όλα τα είδη θορύβου και συνύπαρξη των δύο ειδών θορύβου ταυτόχρονα σε φωτογραφίες.	181
B.18 Δημιουργία συνόλων με όλα τα είδη θορύβου χωρίς συνύπαρξη των δύο ειδών θορύβου ταυτόχρονα σε φωτογραφίες.	181
B.19 Δημιουργία συνόλων με JPEG Compression θόρυβο.	181
B.20 Δημιουργία συνόλων με Gaussian noise.	182
B.21 Οι dataloaders	182
B.22 Ο μετασχηματισμός για ML δεδομένα	182
B.23 Η συνάρτηση flatten_loader	183
B.24 Η απόκτηση των διανυσμάτων χαρακτηριστικών	183
B.25 Η συνάρτηση feature_extraction_ML	184

Κατάλογος πινάκων

4.1	Η κατανομή των εικόνων στην βάση CIFAKE [5], [6]. Στην στήλη κλάση για την ονομασία κάθε κλάσης πραγματοποιήθηκε το καρτεσιανό γινόμενο των δύο διαφορετικών κατηγοριών κλάσεων, όπου η 1η κατηγορία είναι real ή fake και η 2η είναι οι 10 κλάσεις του CIFAR-10 [6].	33
4.2	Η κατανομή των εικόνων στην νέα βάση CIFAKE [5], [6].	34
A.1	Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “clean_model”.	140
A.2	Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “full_noise_model”.	141
A.3	Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “both_noise_model”.	142
A.4	Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “jpeg_model”.	143
A.5	Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “gaussian_model”.	144

Συντομογραφίες

ΒΔ	Βάση Δεδομένων
κ.λπ.	και λοιπά
π.χ.	παραδείγματος χάρη
TN	Τεχνητή Νοημοσύνη
ANN	Artificial Neural Network
CDPM	Conditional Diffusion Probabilistic Models
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DDPM	Denoised Diffusion Probabilistic Models
DL	Deep Learning
DM	Diffusion Model
EMA	Exponential Moving Average
GAN	Generative Adversarial Networks
Gen-AI	Generative Artificial Intelligence
HOG	Histogram of Gradients
ML	Machine Learning
PCA	Principal Component Analysis
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RGB	Red Green Blue
SVM	Support Vector Machine
VGG	Visual Geometry Group
ViT	Vision Transformer

Κεφάλαιο 1

Εισαγωγή

Ένας από τους σημαντικότερους τομείς της όρασης υπολογιστών είναι η ταξινόμηση εικόνων. Η ταξινόμηση εικόνων χρησιμοποιείται για την αναγνώριση αντικειμένων που εμφανίζονται σε μια εικόνα. Περιλαμβάνει την ανάθεση ετικετών στις αρχικές εικόνες με βάση την πιθανότητα το αντικείμενο απεικόνισης να υπάγεται σε μια κλάση. Κάθε εικόνα, έχει μια πραγματική ετικέτα που αντικατοπτρίζει την αληθινή κλάση του αντικειμένου [7] και την ετικέτα που ανατίθεται από το μοντέλο.

Ιστορικά, η ταξινόμηση εικόνων πραγματοποιούνταν με την χρήση χειροποίητων χαρακτηριστικών ή μοντέλων μηχανικής (ML) και βαθιάς μάθησης (DL) για τον εντοπισμό και ταξινόμηση του περιεχομένου της εικόνας. Με την ανάπτυξη της τεχνητής νοημοσύνης, όμως, δημιουργούνται νέες εξελιγμένες τεχνολογίες ικανές να παράξουν ψηφιακά μέσα, όπως εικόνες. Τα μέσα αυτά, γνωστά ως μέσα Γενετικής Τεχνητής Νοημοσύνης (Generative Artificial Intelligence, Gen-AI), έχουν την ικανότητα να φτιάχνουν αληθοφανείς εικόνες.

Παρά τα πολύ μεγάλα πλεονεκτήματα και δυνατότητες που προσφέρουν αυτά τα νέα εργαλεία, η λανθασμένη χρήση τους μπορεί να προκαλέσει μεγάλους κινδύνους. Αρχικά, οι εικόνες που δημιουργούν μοιάζουν εκπληκτικά με αληθινές και οι άνθρωποι σύμφωνα με έρευνες μπορούν να ξεχωρίσουν επιτυχώς μόνο το 61.3% αυτών [8], δημιουργώντας έτσι πρόσφορο έδαφος για την διάδοση ψευδών ειδήσεων. Επιπλέον, μπορούν να χρησιμοποιηθούν πολύ συχνά για προπαγανδιστικούς σκοπούς και για την υπονόμευση της κυβερνοασφάλειας. Προκειμένου να δημιουργηθούν αυτές οι εικόνες, τα μοντέλα κατά την εκπαίδευσή τους έχουν πρόσβαση σε αληθινά έργα δημιουργών. Οι καλλιτέχνες αυτοί αρκετά συχνά δεν ενημερώνονται εγκαίρως και οι εικόνες ή τα έργα τους χρησιμοποιούνται χωρίς την συγκατάθεσή τους, καταπατώντας έτσι τα πνευματικά δικαιώματά τους. Μάλιστα σε αυτές τις

περιπτώσεις, ορισμένες εικόνες που παράγονται τείνουν να προσομοιάζουν σε μεγάλο βαθμό με έργα τέχνης εγείροντας αμφιβολίες σχετικά με την αυθεντικότητά τους [8, 9, 10, 11]. Ο μεγαλύτερος, ίσως, κίνδυνος που παρουσιάζεται, είναι ο καθορισμός της αυθεντικότητάς τους σε φωτογραφίες deepfake. Οι deepfake εικόνες παράγονται με την χρήση τεχνικών μηχανικής μάθησης και οι εικόνες που παράγουν απεικονίζουν συνήθως αληθινούς ανθρώπους, ολόκληρους ή μέρη του σώματός τους, σε καταστάσεις που δεν βρέθηκαν στην πραγματικότητα. Ο μεγάλος ρεαλισμός των εικόνων αυτών μπορεί να λειτουργήσει καταστροφικά για τους απεικονιζόμενους ανθρώπους δημιουργώντας σοβαρά προβλήματα για την ασφάλειά τους [12].

1.1 Αντικείμενο της διπλωματικής

Γίνεται κατανοητό πως είναι αναγκαίο οι εικόνες που δημιουργούνται από μέσα τεχνητής νοημοσύνης να ανιχνεύονται εύκολα και γρήγορα. Για να επιτευχθεί αυτό, δημιουργούνται συνεχώς νέα συστήματα ανίχνευσης και ταξινόμησης. Η χρήση συνελκτικών νευρωνικών δικτύων (CNNs) μπορεί να αποδειχθεί καταλυτική για την επίτευξη του στόχου αυτού λόγω της μεγάλης ικανότητάς τους σε ταξινόμηση. Σκοπός της διπλωματικής εργασίας είναι η δημιουργία ενός τέτοιου συστήματος που χρησιμοποιεί ένα CNN μοντέλο, το ResNet-50 [3], ως βάση και ταξινομεί τις εικόνες ανάλογα με την πηγή προέλευσής τους σε αληθινές και κατασκευασμένες από Gen-AI, ενώ ταυτόχρονα τις τοποθετεί σε διάφορες κλάσεις ανάλογα με το αντικείμενο απεικόνισης. Το μοντέλο εκπαιδεύεται με γνώμονα την μεγιστοποίηση της κοινής ακρίβειας, δηλαδή την ταυτόχρονη σωστή τοποθέτηση και των δύο ετικετών σε μια εικόνα.

1.2 Συνεισφορά

Σε αυτή την διπλωματική ερευνάται η δημιουργία ενός συστήματος ταξινόμησης εικόνων. Για να πραγματοποιηθεί αυτό:

1. Αναπτύσσεται το συνελκτικό νευρωνικό δίκτυο ResNet50Classifier, που χρησιμοποιεί το μοντέλο ResNet-50 [3] ως βάση για την ανάπτυξη ενός συνελκτικού δικτύου δύο κεφαλών, ώστε να αναγνωρίζει εικόνες από μοντέλα διάχυσης (Diffusion Models-DM) και να τις ξεχωρίζει από αληθινές εικόνες και συγχρόνως να τις ταξινομεί σε δέκα

- κλάσεις ανάλογα με το αντικείμενο απεικόνισης.
2. Δοκιμάζεται η ακρίβεια του μοντέλου σε διάφορες συνθήκες θορύβου που προσομοιώνουν σε πραγματικά σενάρια χρήσης και ελέγχεται η αποτελεσματικότητά του. Συγκεκριμένα, δημιουργούνται πέντε διαφορετικές συνθήκες θορύβου και εκπαιδεύεται ένα μοντέλο σε κάθε μία. Όλα τα μοντέλα δοκιμάζονται και στις πέντε συνθήκες θορύβου και πραγματοποιείται συγκριτική αξιολόγηση ανά μοντέλο και ανά συνθήκη θορύβου.
 3. Πραγματοποιείται συγκριτική αξιολόγηση της απόδοσης του μοντέλου σε σχέση με τρία παραδοσιακά μοντέλα μηχανικής μάθησης.
 4. Μελετάται η δυνατότητα ενσωμάτωσης του CNN σε υβριδικό μοντέλο ανίχνευσης, που αναλαμβάνει ρόλο εξαγωγής χαρακτηριστικών, τα οποία έπειτα τροφοδοτούνται σε ταξινομητή ML. Συγκεκριμένα εκπαιδεύονται τέσσερα διαφορετικά υβριδικά μοντέλα, που το κάθε ένα έχει διαφορετικό ταξινομητή, και πραγματοποιείται σύγκριση των υβριδικών μοντέλων με το αρχικό ResNet50Classifier.

1.3 Οργάνωση του τόμου

Η υπόλοιπη Διπλωματική έχει την εξής δομή:

- Στο κεφάλαιο 2 παρουσιάζεται ο τρόπος λειτουργίας των παραγωγικών μέσων τεχνητής νοημοσύνης και μελετώνται προηγούμενες έρευνες στον τομέα της ταξινόμησης εικόνων με έμφαση στις εικόνες Gen-AI.
- Στο κεφάλαιο 3 παρουσιάζεται το θεωρητικό υπόβαθρο για τα συνελκτικά νευρωνικά μοντέλα, το μοντέλο ResNet50 [3] και την έννοια της υπολειμματικής μάθησης [3], καθώς και για τους ταξινομητές Support Vector Machine, Random Forest Classifier και XGBoost.
- Στο κεφάλαιο 4 παρουσιάζεται η βάση δεδομένων (ΒΔ) που χρησιμοποιείται, καθώς και όλα τα βήματα προεπεξεργασίας που δέχονται τα δεδομένα.
- Στο κεφάλαιο 5 παρουσιάζεται το μοντέλο ResNet50Classifier και γίνεται μελέτη της δομής και της λειτουργίας του και των ρουτινών εκπαίδευσης, ελέγχου και δοκιμής.

- Στο κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα δοκιμής του μοντέλου στην βάση δεδομένων.
- Στο κεφάλαιο 7 μελετάται η αντοχή του μοντέλου υπό διαφορετικές συνθήκες θορύβου, περιγράφεται εκτενώς η δημιουργία νέων δεδομένων και η διαδικασία διεξαγωγής των πειραμάτων και παρουσιάζονται τα αποτελέσματα.
- Στο κεφάλαιο 8 συγκρίνεται η απόδοση του μοντέλου σε σχέση με παραδοσιακές τεχνικές μηχανικής μάθησης και ερευνάται η δημιουργία υβριδικού συστήματος ταξινόμησης, το οποίο χρησιμοποιεί: α) το ResNet50Classifier ως μέσο εξαγωγής χαρακτηριστικών και β) μοντέλα μηχανικής μάθησης ως ταξινομητές.
- Στο κεφάλαιο 9 παρουσιάζονται: α) η σύνοψη της έρευνας, β) τα συμπεράσματα των πειραμάτων και γ) οι κατευθύνσεις για μελλοντική έρευνα.

Κεφάλαιο 2

Βιβλιογραφική ανασκόπηση

Σε αυτό το κεφάλαιο αρχικά παρουσιάζονται τα κύρια είδη αρχιτεκτονικών των μέσων παραγωγικής τεχνητής νοημοσύνης. Έπειτα, αναφέρονται οι μέθοδοι που χρησιμοποιούνται για την ταξινόμηση εικόνων και ειδικότερα για την ανίχνευση εικόνων που παράγονται από μέσα τεχνητής νοημοσύνης (TN).

2.1 Είδη παραγωγικής τεχνητής νοημοσύνης

Με την ραγδαία ανάπτυξη στον τομέα της βαθιάς μάθησης αναπτύχθηκαν νέα μοντέλα βαθιάς μάθησης που είναι ικανά να παραγάγουν διάφορα μέσα, όπως εικόνες, βίντεο, ήχους κλπ. Τα κύρια είδη αυτών των μέσων είναι τρία:

1. Τα Παραγωγικά Αντιπαραθετικά Δίκτυα, γνωστά και ως GAN,
2. τα Μοντέλα Διάχυσης (DM),
3. τα μοντέλα μετασχηματιστή (Transformer).

Τα τρία είδη διαφέρουν μεταξύ τους ως προς την αρχιτεκτονική και τον τρόπο λειτουργίας [1].

2.1.1 Παραγωγικά Αντιπαραθετικά Δίκτυα

Τα Παραγωγικά Αντιπαραθετικά Δίκτυα αποτελούνται από ένα σύμπλεγμα δύο διαφορετικών νευρωνικών δικτύων, το παραγωγικό δίκτυο και το διαχωριστικό δίκτυο και λειτουργούν ως ανταγωνιστές με τελικό στόχο την παραγωγή ρεαλιστικών εικόνων. Αρχικά

δίνεται ως είσοδος θόρυβος για εκμάθηση στο σύστημα και ο παραγωγός προσπαθεί να παράγει παραδείγματα από ένα λανθάνοντα χώρο, ενώ ο διαχωριστής χρησιμοποιώντας την κατανομή των παραγόμενων δεδομένων προσπαθεί να διαχωρίσει τις πραγματικές από τις κατασκευασμένες εικόνες, απορρίπτοντας τις συνθετικές. Με την σειρά του, ο παραγωγός καλείται να δημιουργήσει όλο και πιο ρεαλιστικές φωτογραφίες προκειμένου να “ξεγελάσει” τον διαχωριστή, δημιουργώντας έτσι έναν κύκλο στον οποίο μετά από πολλές επαναλήψεις ο παραγωγός πετυχαίνει να κατασκευάσει εικόνες που έχουν ακρίβεια αποδεκτή από τον διαχωριστή [1]. Στην ουσία πρόκειται περί ενός “παιχνιδιού” ελαχιστοποίησης-μεγιστοποίησης, στο οποίο ο διαχωριστής D εκπαιδεύεται με στόχο την σωστή ταξινόμηση των δεδομένων που του δίνονται και η γεννήτρια G προσπαθεί να δημιουργήσει όλο και πιο ρεαλιστικές εικόνες “ξεγελώντας” τον διαχωριστή, και επομένως, ελαχιστοποιώντας τις πιθανότητες ο D να κάνει σωστή ταξινόμηση. Μαθηματικά αυτή η διαδικασία βελτιστοποίησης εκφράζεται ως:

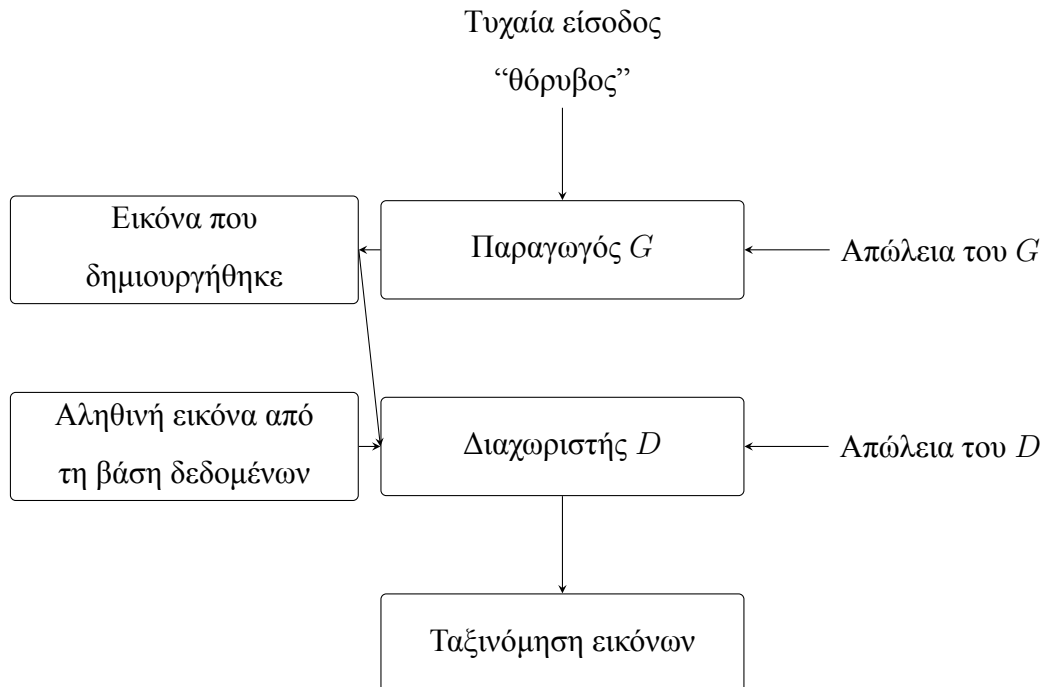
$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Τέλος, δεν είναι δυνατόν ο διαχωριστής και η γεννήτρια να εκπαιδεύονται με τον ίδιο ρυθμό, διότι οι πράξεις που απαιτούνται είναι “ακριβές” και επιβαρύνεται υπερβολικά το σύστημα. Για να λυθεί το πρόβλημα αυτό, σε ορισμένα βήματα γίνεται εναλλαγή του μοντέλου που βελτιστοποιείται και συγκεκριμένα αν έχουμε $n+1$ βήματα τότε ο D θα βελτιστοποιηθεί στα n από αυτά, ενώ ο G στο 1 από αυτά. Αυτή η εναλλαγή δεν προκαλεί προβλήματα στην επίδοση του μοντέλου, διότι και ο D παραμένει πάντα κοντά στην βέλτιστη κατάσταση και ο G βελτιώνεται σταδιακά [13].

Βέβαια, αυτό δεν σημαίνει πως οι εικόνες δεν έχουν ανεπιθύμητες παραμορφώσεις, εξαιτίας των οποίων διακρίνονται από τις αληθινές. Μάλιστα αυτές οι παραμορφώσεις μπορούν να λειτουργήσουν ως είδος “αποτυπώματος”, καθώς είναι ξεχωριστές για κάθε μοντέλο GAN [9]. Υπάρχει, επίσης, η ανάγκη για επίβλεψη κατά την διάρκεια της εκπαίδευσης του μοντέλου, διότι είναι ορατός ο κίνδυνος το παραγωγικό δίκτυο, λόγω της πιο απαιτητικής φύσης της λειτουργίας του, να επικεντρωθεί σε ορισμένα μόνο χαρακτηριστικά του συνόλου εκπαίδευσης, με αποτέλεσμα να παράγονται εικόνες χωρίς την επιθυμητή θεματολογική ποικιλία. Το φαινόμενο αυτό ονομάζεται mode collapse [14]. Μοντέλα GAN που χρησιμοποιούνται συχνά είναι τα:

1. ProGAN [15],

2. StyleGAN [16],
3. BigGAN [17].



Σχήμα 2.1: Αναπαράσταση της δομής ενός Παραγωγικού Αντιπαραθετικού Δικτύου GAN (σχήμα προσαρμοσμένο από [1]).

2.1.2 Μοντέλα διάχυσης

Τα μοντέλα διάχυσης από την άλλη λειτουργούν με αρκετά διαφορετικό τρόπο. Αρχικά, είναι πιθανοκρατικά παραγωγικά μοντέλα και μαθαίνουν να δημιουργούν περιεχόμενο μέσα από εκπαίδευση δύο σταδίων. Στο πρώτο στάδιο, αντιστοιχούν την κατανομή των δεδομένων σε μια απλούστερη πρότερη κατανομή, συνήθως Γκαουσιανή και στο δεύτερο ακολουθούν την αντίστροφη διαδικασία, προσπαθούν δηλαδή μέσα από την χρήση κατάλληλα εκπαιδευμένων νευρωνικών δικτύων να αφαιρέσουν τον θόρυβο και να καταλήξουν στην επιθυμητή έξοδο. Ανάλογα με την μέθοδο που χρησιμοποιούν στα στάδια αυτά διακρίνονται σε Denoised Diffusion Probabilistic Models (DDPM), Score SDE formulation και Conditional Diffusion Probabilistic Models [18].

Η διαδικασία προσθήκης Γκαουσιανού θορύβου σε στάδια σε μοντέλα DDPM αποτελεί στην ουσία μια μαρκοβιανή αλυσίδα κατά την οποία το μοντέλο μεταβαίνει από μια άγνωστη

κατανομή $x_0 \sim p_{\text{data}}(x_0)$ σε T βήματα σε μια απλούστερη κατανομή $q(x_T) \approx \mathcal{N}(0, I)$, αξιοποιώντας μια σειρά από σταθερές θορύβου β για την δημιουργία μεταβατικών μαρκοβιανών πυρήνων. Μαθηματικά η σταδιακή αυτή προσθήκη εκφράζεται ως:

$$q(x_t | x_{t-1}) = \mathcal{N}(\sqrt{\alpha_t} x_{t-1}, \beta_t I)$$

με $\alpha_t = 1 - \beta_t$ και $\alpha_t = \prod_{s=1}^t \alpha_s$.

Το δεύτερο στάδιο αποτελεί και εκείνο μαρκοβιανή αλυσίδα όπου το μοντέλο ξεκινάει από την κατανομή

$$p(x_t) \approx \mathcal{N}(0, I), \quad p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t)$$

και μεταβαίνει σε προηγούμενα στάδια με βήμα $p_\theta(x_{t-1} | x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ [19].

Τα μοντέλα SDE αποτελούν φυσική εξέλιξη των DDPM καθώς επεκτείνουν το πλαίσιο λειτουργίας διακριτού χρόνου των DDPM και λειτουργούν σε πλαίσιο συνεχούς χρόνου. Για να το επιτύχουν αυτό, βασίζονται σε ένα προωθητικό στάδιο και σε ένα στάδιο γέννησης, απλά ενσωματώνοντας στοχαστικές διαφορικές εξισώσεις για την μετάβαση στις διαφορετικές κατανομές. Τέλος τα CDPM, σε αντίθεση με τα DDPM, ξεκινούν από μια κατανομή $p_\theta(x | c)$, όπου c είναι μια γνωστή συνθήκη, π.χ. ετικέτα για μια κλάση ή κείμενο περιγραφής [18].

Σε αντίθεση με τα GAN, κατά τη διαδικασία εκπαίδευσης των DM ο στόχος παραμένει σταθερός, καθώς δεν υπάρχουν δύο ανταγωνιστικά δίκτυα. Από την άλλη πλευρά, η διαδικασία της εκπαίδευσης πολύ συχνά είναι πιο “ακριβή” από την αντίστοιχη των GAN, διότι η μετατροπή από μια απλή κατανομή δεδομένων σε μια σύνθετη είναι χρονοβόρα και πιο ασταθής, κυρίως στο στάδιο της δειγματοληψίας [18]. Τα μοντέλα διάχυσης μπορούν να δημιουργήσουν ήχο και εικόνες [5], ακόμη και κείμενα και βίντεο. Μάλιστα, οι εικόνες-έργα τέχνης είναι τόσο ρεαλιστικές που ξεγελούν ανθρώπους. Γνωστό μοντέλο DM είναι το Latent Diffusion [20], που χρησιμοποιείται συχνά τόσο για εμπορικούς όσο και για ερευνητικούς σκοπούς με την δημιουργία ενός συνόλου από συνθετικές εικόνες για την εκπαίδευση μοντέλων τεχνητής νοημοσύνης.

2.1.3 Μοντέλα μετασχηματιστές

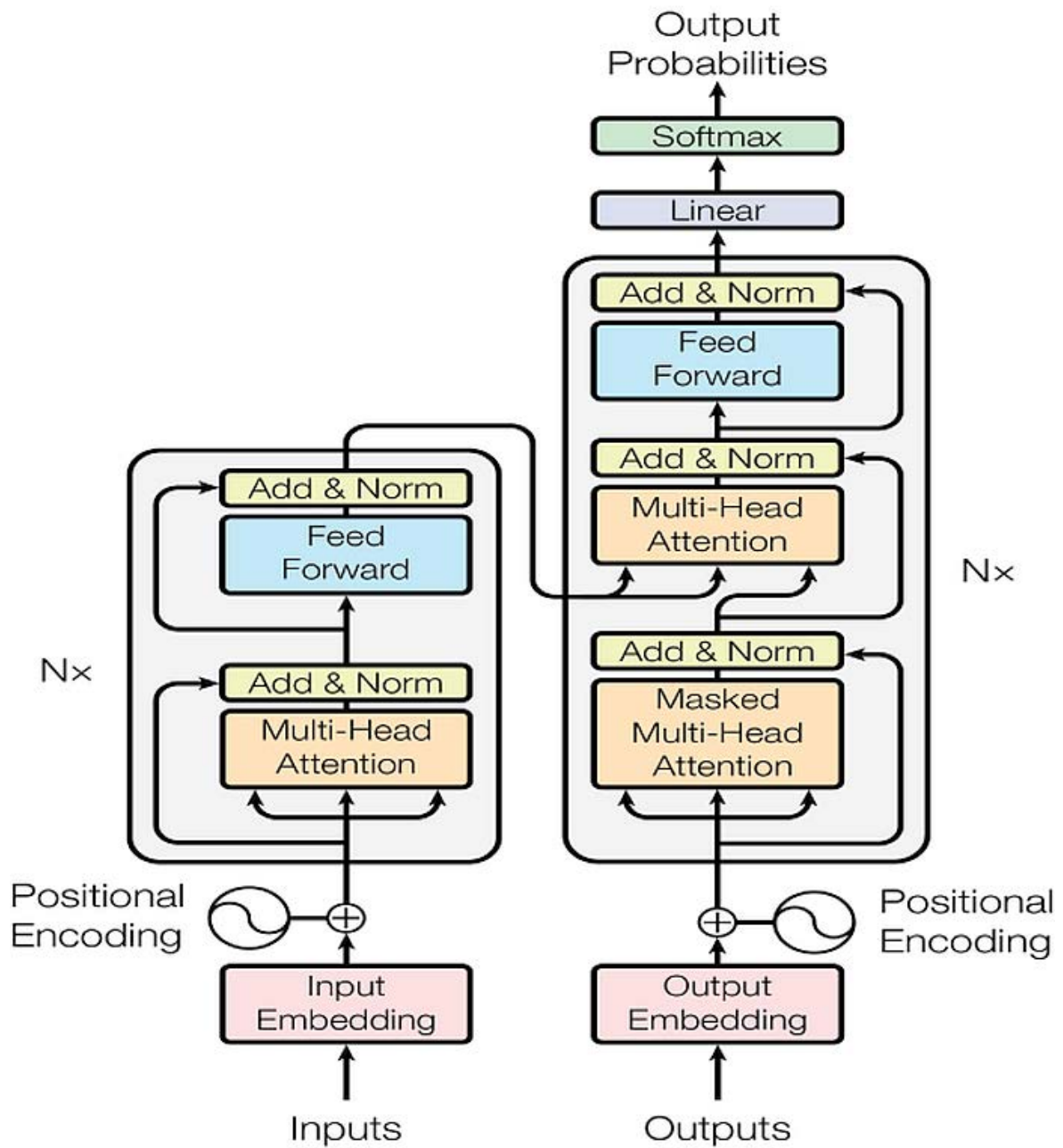
Οι μετασχηματιστές αποτελούνται από δύο διακριτά νευρωνικά δίκτυα πρόσθιας τροφοδότησης που αξιοποιούν ένα μηχανισμό προσοχής. Ο μηχανισμός προσοχής, γνωστός ως

self-attention mechanism, επιτρέπει στο μοντέλο να υποθέσει πόση επιρροή έχει κάθε στοιχείο της εισόδου στα υπόλοιπα στοιχεία της εισόδου. Για να το πετύχει αυτό, αντιπροσωπεύει κάθε στοιχείο της εισόδου με ένα διάνυσμα $X = (x_1, x_2, \dots, x_n)$ και έπειτα για κάθε ένα από αυτά δημιουργεί τρία νέα διανύσματα Q, K, V που ονομάζονται αίτημα (Query), κλειδί (Key), τιμή (Value) αντιστοίχως, πολλαπλασιάζοντάς τα με τους αντίστοιχους πίνακες βαρών W_Q, W_K, W_V . Για να υπολογίσει την αξία του κάθε στοιχείου, αρχικά υπολογίζει το εσωτερικό γινόμενο των διανυσμάτων αίτησης και κλειδιού, μετά κανονικοποιεί τα αποτελέσματα προκειμένου να λάβει την κατανομή των βαρών για κάθε στοιχείο και τέλος εφαρμόζει αυτά τα βάρη στο διάνυσμα τιμών, για να δημιουργήσει το τελικό διάνυσμα εξόδου $Z = (z_1, z_2, \dots, z_n)$. Ο μηχανισμός αυτός είναι ιδιαίτερα χρήσιμος, καθώς από την μία επιτρέπει στο μοντέλο να επεξεργαστεί μεγάλο όγκο δεδομένων χωρίς μεγάλο χρονικό κόστος, διότι ο αλγόριθμος υπολογισμού της εξόδου είναι παραλληλίστιμος, και από την άλλη το μοντέλο μπορεί να καταλάβει μοτίβα που εμφανίζονται σε μεγάλο αριθμό χαρακτηριστικών και επομένως να κατανοήσει σύνθετες δομές [2, 21].

Τα νευρωνικά δίκτυα, από την άλλη, αποτελούν μέρος δύο μεγαλύτερων μονάδων, του κωδικοποιητή και του αποκωδικοποιητή. Στόχος του κωδικοποιητή είναι να κωδικοποιήσει την είσοδο και τις σχέσεις των στοιχείων της εισόδου σε διακριτά σύμβολα, τα οποία με την σειρά τους τα χρησιμοποιεί ο αποκωδικοποιητής για να προσθέσει πληροφορίες από συμφραζόμενα και τέλος να παρουσιάσει την έξοδο ως μια κατανομή δεδομένων [21]. Η δομή ενός μετασχηματιστή παρουσιάζεται στο σχήμα 2.2.

2.2 Μέθοδοι ταξινόμησης εικόνων

Πριν από την ανάπτυξη της βαθιάς μάθησης και των νευρωνικών δικτύων τα κύρια μέσα ταξινόμησης εικόνων ήταν τα μοντέλα μηχανικής μάθησης και τα χαρακτηριστικά των εικονοστοιχείων. Η ανάπτυξη της βαθιάς μάθησης επιτρέπει να αξιοποιηθούν στον τομέα αυτό νέες μέθοδοι πολύ πιο αποτελεσματικές. Από τις πλέον δημοφιλείς είναι η χρήση νευρωνικών δικτύων με αρχιτεκτονικές που μπορεί να είναι σχετικά απλές, όπως αυτή ενός κλασσικού CNN, έως και αρκετά σύνθετες, όπως ενός GAN ή ενός μετασχηματιστή και η φασματική ανάλυση των εικόνων. Τέλος, πολλές φορές χρησιμοποιούνται υβριδικά συστήματα ταξινόμησης που αξιοποιούν τουλάχιστον δύο μεθόδους ανίχνευσης.



Σχήμα 2.2: Η δομή ενός μετασχηματιστή και ανάλυση της εσωτερικής λειτουργίας του κωδικοποιητή και του αποκωδικοποιητή του (σχήμα από [2]).

2.2.1 Ταξινόμηση με χρήση μηχανικής μάθησης

Τα μοντέλα μηχανικής μάθησης ήταν από τα πρώτα που αξιοποιήθηκαν στον τομέα αυτό, εξαιτίας του σχετικά μικρού τους υπολογιστικού κόστους και της σχετικά καλής ακρίβειας. Μάλιστα, ακόμη και σήμερα χρησιμοποιούνται αυτές οι τεχνικές σε συστήματα ταξινόμησης και αναλαμβάνουν είτε βοηθητικό ρόλο είτε εξ ολοκλήρου την ταξινόμηση, παρά την ανάπτυξη νέων πιο αποτελεσματικών μέσων. Μερικά μοντέλα μηχανικής μάθησης (Machine Learning - ML) είναι οι μηχανές διανυσματικής στήριξης (Support Vector Machines - SVMs), τα Multi-layer Perceptron (MLP), διάφοροι αλγόριθμοι ταξινόμησης όπως K-NN [22], τα δέντρα απόφασης (decision trees), καθώς και πιο σύνθετες τεχνικές που στηρίζονται στα δέντρα απόφασης.

Κυρίαρχη θέση στην ταξινόμηση κατέχουν τα SVMs, όπως φανερώνεται και από την βιβλιογραφία, χάρη στην ικανότητά τους να βρίσκουν σχετικά εύκολα το υπερεπίπεδο διαχωρισμού, ακόμη και σε περιπτώσεις που οι κλάσεις δεν είναι γραμμικά διαχωρίσιμες εξαρχής. Για τον στόχο της ταξινόμησης εικόνων τα μέσα αυτά συνοδεύονται από κάποια μέθοδο μείωσης διαστασιμότητας χώρου χαρακτηριστικών, π.χ. PCA ή κάποιο τρόπο εξαγωγής χαρακτηριστικών, π.χ. χρήση ιστογράμματος κλίσεων HOG.

Στην έρευνα [23] χρησιμοποιείται SVM και PCA για μείωση διαστάσεων χαρακτηριστικών. Με αυτό τον τρόπο ταξινομούνται μονοχρωματικές εικόνες και συγκρίνεται η απόδοση του συστήματος με άλλα τρία νευρωνικά δίκτυα. Ενώ θεωρητικά το SVM πρέπει να έχει χαμηλότερη απόδοση εξαιτίας της απλούστερης δομής του και της μειωμένης ικανότητάς του να αντιλαμβάνεται περίπλοκα μοτίβα σε σχέση με τα δίκτυα βαθιάς μάθησης, καταφέρνει να επιτύχει πολύ υψηλότερα ποσοστά ακρίβειας, δείχνοντας την εξαιρετική ικανότητά του να συγκλίνει και να βρίσκει όρια απόφασης. Το ίδιο συμπέρασμα σχετικά με την ανώτερη ικανότητα του SVM σε ταξινόμηση προκύπτει και σε σύγκριση με άλλα μέσα μηχανικής μάθησης. Σε σύγκριση με τις μεθόδους MLP, Naive Bayes, K-NN και δέντρο απόφασης, το SVM πετυχαίνει τα καλύτερα αποτελέσματα ταξινόμησης σε πολλές κλάσεις. Ωστόσο, αξίζει να σημειωθεί πως ακόμη και αυτό δυσκολεύεται πολλές φορές να ξεχωρίσει αντικείμενα από κάποιες κλάσεις και τείνει να τοποθετεί λάθος ετικέτες, ειδικά σε παρόμοιες κλάσεις, ακόμη και όταν γίνεται προσεκτική προσαρμογή των υπερπαραμέτρων εκπαίδευσης και προεπεξεργασία των δεδομένων [24]. Σε αντίθεση με τα προηγούμενα αποτελέσματα που δείχνουν πως το SVM είναι πολύ καλό εργαλείο ταξινόμησης, η έρευνα [25] αποδεικνύει πως ακόμη και ένα απλό συνελκτικό νευρωνικό δίκτυο μπορεί να επιτύχει αρκετά καλύτερα αποτελέ-

σματα επιβεβαιώνοντας και στην πράξη την αδυναμία των SVMs να αντιληφθούν περίπλοκα κρυμμένα μοτίβα, όπως στην περίπτωση αναγνώρισης εικόνων από μέσα Gen-AI.

Το μειονέκτημα της μικρής πολυπλοκότητας των μοντέλων ML μπορεί να επιλυθεί, εάν χρησιμοποιηθεί συμπληρωματικά ένα άλλο είδος ταξινομητή στην αρχή για την εξαγωγή χαρακτηριστικών, τα οποία στην συνέχεια τροφοδοτούνται ως είσοδος στον ML ταξινομητή που επιλέγει τις ετικέτες για τις εικόνες. Για το πρώτο στάδιο προτιμώνται νευρωνικά δίκτυα τα οποία έχουν μεγαλύτερη υπολογιστική δύναμη, οπότε μπορούν να εντοπίσουν και να αποτυπώσουν πιο εύκολα σχέσεις μεταξύ δεδομένων. Αυτά δεν τοποθετούν ετικέτες και συνήθως σταματούν την επεξεργασία σε ένα ενδιάμεσο στάδιο, στο οποίο έχουν βρεθεί αρκετά σημαντικά χαρακτηριστικά που περνούν ως είσοδος στον τελικό ταξινομητή. Με αυτόν τον τρόπο επιταχύνεται η εκπαίδευση του μοντέλου και η διαδικασία ταξινόμησης, εξοικονομούνται πολύτιμοι υπολογιστικοί πόροι, καθώς το “ακριβό” δίκτυο βαθιάς μάθησης δεν χρησιμοποιείται μέχρι το τέλος, και επιτυγχάνεται μεγάλη ικανότητα ταξινόμησης, σαφώς ανώτερη από αυτή ενός συστήματος που χρησιμοποιεί απλώς μέσα ML και συγκρίσιμη με εκείνη συστήματος DL.

Πρωτοποριακή στον τομέα αυτό είναι η μέθοδος που προτάθηκε από τους Le et al. [26] για την αναγνώριση εικόνων με ρωμαϊκά νούμερα. Σύμφωνα με αυτή, στην αρχή οι εικόνες δέχονται προεπεξεργασία και εξάγονται τα χαρακτηριστικά τους. Ο χώρος των χαρακτηριστικών μετά υποδιαιρείται σε τρεις υποχώρους και κάθε ένας λειτουργεί ως είσοδος για ένα τεχνητό νευρωνικό δίκτυο (Artificial Neural Network - ANN) όπου γίνεται περαιτέρω επεξεργασία και ταξινόμηση. Τέλος, όλα τα δεδομένα περνούν σε ένα SVM όπου συνενώνονται και γίνεται η τελική ταξινόμηση. Η πρωτοτυπία στην έρευνα αυτή βρίσκεται στην βοηθητική χρήση ANN πριν ακόμα κυριαρχήσουν τα CNN.

Με την εξέλιξη της τεχνολογίας και την ανάπτυξη των συνελκτικών δικτύων πολλές έρευνες στράφηκαν στην χρήση αυτών παράλληλα με διάφορα μέσα μηχανικής μάθησης. Για παράδειγμα, στην έρευνα του Xi[27] και στην έρευνα των Jiao et al. [28] προτείνονται υβριδικά συστήματα που χρησιμοποιούν CNN για αρχική εξαγωγή χαρακτηριστικών και Random Forest και XGBoost αντίστοιχα, για την ταξινόμηση. Η διαφορά αυτών των δύο έγκειται στον στόχο που επέλεξαν. Το πρώτο μοντέλο [27] έχει ως στόχο την επίτευξη μέγιστης απόδοσης, δηλαδή με περιορισμένη χρήση πόρων να ελαχιστοποιηθεί το λάθος ταξινόμησης και γι' αυτό χρησιμοποιήθηκαν τυχαία βάρη στο CNN, παρακάμπτοντας την εκπαίδευση μέσω οπισθοδιάδοσης και μειώνοντας έτσι το μεγαλύτερο μέρος του κόστους λειτουργίας.

Η μέθοδος αυτή απέδωσε σε σύγκριση με παραδοσιακά μοντέλα Random Forest, πετυχαίνοντας μικρότερο ποσοστό λάθους ταξινόμησης [27]. Το δεύτερο μοντέλο [28], αντιθέτως, προσπάθησε να πετύχει βέλτιστη απόδοση χωρίς να παραλείψει την διαδικασία εκπαίδευσης και του δικτύου βαθιάς μάθησης και του XGBoost. Οι παράμετροι και των δύο μοντέλων προσαρμόζονται στις ανάγκες της έρευνας ταυτόχρονα, γεγονός που επιτρέπει τα δύο μέρη να εξειδικευτούν στους ρόλους τους και να συνεργαστούν καλύτερα, με αποτέλεσμα το σύστημα να πετυχαίνει πολύ υψηλά ποσοστά ακρίβειας σε απαιτητική βάση δεδομένων [28]. Πάντως, και από τις τρεις αυτές έρευνες [26, 27, 28] τονίζεται ότι παρά την ανάπτυξη νευρωνικών δικτύων, παραδοσιακές μέθοδοι ταξινόμησης ML δεν έχουν χάσει την θέση τους στην ταξινόμηση εικόνων, απλώς συνηθίζεται πλέον να επιτελούν ένα συμπληρωματικό ρόλο, σε συνεργασία με τα υπόλοιπα μέρη του συστήματος.

2.2.2 Ταξινόμηση με CNN και μετασχηματιστή ViT

Τα συνελκτικά νευρωνικά δίκτυα κυριάρχησαν σταδιακά στον τομέα της ταξινόμησης εικόνων λόγω της ικανότητάς τους να εντοπίζουν χαρακτηριστικά, όπως γωνίες, χρώμα, υφή και σημασιολογικό περιεχόμενο απευθείας από απλά εικονοστοιχεία, σε αντίθεση με τις παραδοσιακές μεθόδους που εξετάστηκαν, οι οποίες απαιτούν κάποια μέθοδο εξαγωγής χαρακτηριστικών. Η χρήση τους σηματοδοτεί την μεταστροφή από χειροκίνητα σχεδιασμένα χαρακτηριστικά στην αυτοματοποιημένη εξαγωγή τους.

Τα μοντέλα αυτά μπορούν να εντοπίσουν εύκολα μοτίβα σε εικόνες τόσο σε επίπεδο εικονοστοιχείου όσο και σε επίπεδο περιοχής της εικόνας. Η ικανότητα αυτή οφείλεται στην χρήση συνελκτικών φίλτρων, στην χρήση υποδειγματοληψίας και στην εφαρμογή κανονικοποίησης. Τα φίλτρα είναι παραμετροποιήσιμα και διαφέρουν ως προς το μέγεθος του πυρήνα και το βήμα. Δίνουν ελευθερία στον σχεδιαστή του μοντέλου να τα προσαρμόσει ανάλογα με τις ανάγκες του συστήματος ταξινόμησης. Επίσης, η υποδειγματοληψία και η κανονικοποίηση επιτρέπουν στο μοντέλο να εστιάζει σε γενικότερα χαρακτηριστικά και σταθεροποιούν την διαδικασία εκπαίδευσης αυξάνοντας την αντοχή σε θόρυβο. Γίνεται κατανοητό πως ακόμη και σχετικά απλές αρχιτεκτονικές είναι ιδανικές για ταξινόμηση, ειδικά για τον σκοπό της ανίχνευσης εικόνων από Gen-AI που χρειάζεται δυαδική ταξινόμηση. Για παράδειγμα οι έρευνες των Bird et al. [5] και Hossain et al. [10] εξερευνούν τον ρόλο των CNN στην ταξινόμηση τέτοιων εικόνων και κάνουν συγκρίσεις ανάμεσα σε παρόμοιες αρχιτεκτονικές. Η πρώτη έρευνα [5] με μόλις τρία συνελκτικά στρώματα πετυχαίνει ακρίβεια της

τάξης του 93% για αναγνώριση πολύ μικρών εικόνων που παράγονται από μοντέλα DM. Οι εικόνες αυτές ανήκουν στη βάση δεδομένων CIFAKE [5, 6]. Η δεύτερη [10] έρευνα βασίζεται στα ευρήματα της πρώτης, χρησιμοποιεί την ίδια βάση δεδομένων CIFAKE [5, 6] και αναπτύσσει διάφορα μοντέλα CNN και ένα μοντέλο ViT. Τα συνελκτικά δίκτυα που χρησιμοποιεί έχουν όλα περισσότερα συνελκτικά στρώματα από τα αντίστοιχα της έρευνας [5] και πετυχαίνουν ανώτερα ποσοστά ακρίβειας φτάνοντας το 96.31%. Η βαθύτερη αρχιτεκτονική που χρησιμοποιείται στην [10] σε συνδυασμό με την κατάλληλη προσαρμογή περισσότερων παραμέτρων οδηγούν σε καλύτερη εξαγωγή χαρακτηριστικών και δίνουν την δυνατότητα στο μοντέλο να εντοπίζει πιο δύσκολα μοτίβα επιτρέποντάς του έτσι να ταξινομεί καλύτερα τις φωτογραφίες.

Γίνεται φανερό πως συνήθως βαθύτερες αρχιτεκτονικές οδηγούν σε καλύτερα ποσοστά ταξινόμησης, αφού προστίθεται πολυπλοκότητα στην δομή του μοντέλου, η οποία του επιτρέπει να αντιλαμβάνεται περισσότερες λεπτομέρειες. Σε αυτή την παρατήρηση βασίζεται και το μοντέλο VGG (Visual Geometry Group), το οποίο είναι ένα CNN με βαθιά αρχιτεκτονική πολλών συνελκτικών στρωμάτων [29]. Το VGG [29] μαζί με το ResNet [3], που αναλύεται σε βάθος στο τρίτο κεφάλαιο 3, και το DenseNet [30] που βασίζεται με την σειρά του στο ResNet, αλλά αντί για συνδέσεις υπολειμμάτων συνδέει κάθε στρώμα με όλα τα προηγούμενά του, χρησιμοποιούνται σε έρευνες ως μέσα αναφοράς και σύγκρισης για την υψηλή ταξινομητική τους ικανότητα. Χαρακτηριστικό παράδειγμα αποτελεί η έρευνα των Taeb et al. [12], στην οποία συγκρίνονται ένα CNN, το DenseNet-264 [30] και το VGG-19 [29] στην ανίχνευση εικόνων deepfake. Στην συγκεκριμένη έρευνα [12] αποδεικνύεται πως όλα τα μοντέλα είναι ικανά να εντοπίσουν εικόνες που προήλθαν από εργαλεία GAN. Λόγω της βαθύτερης και περίπλοκης αρχιτεκτονικής τους τα DenseNet [30] και VGG [29] πετυχαίνουν σαφώς καλύτερες επιδόσεις από το CNN. Μάλιστα η χρήση VGG-19 επιφέρει τα καλύτερα αποτελέσματα, αλλά η εκπαίδευση και η χρήση του απαιτούν πολλούς πόρους [12].

Το μεγαλύτερο βάθος ενός συνελκτικού δικτύου βέβαια δεν εγγυάται καλύτερη απόδοση και εάν η αρχιτεκτονική είναι πολύ περίπλοκη και βαθιά για ορισμένα δεδομένα, τότε υπάρχει κίνδυνος το δίκτυο να προσαρμοστεί υπερβολικά στα δεδομένα εκπαίδευσης και να χάσει την δυνατότητα γενίκευσης ή να μην μπορεί να αποτυπώσει απλά μοτίβα μεταξύ εικόνων. Σε αυτές τις περιπτώσεις μια καλή λύση είναι να χρησιμοποιηθεί ένα πιο απλό CNN με λιγότερα στρώματα, το οποίο είναι υπολογιστικά ελαφρύτερο και γρηγορότερο. Οι έρευνες των

Lokner et al. [14] και Chinta et al. [31] υποστηρίζουν αυτή την παρατήρηση. Στην πρώτη [14] αναπτύσσεται ένα ελαφρύ μοντέλο CNN που πετυχαίνει επιδόσεις καλύτερες από το ResNet [3] και VGGNet [29] και συγκρίσιμες με το DenseNet [30]. Με αυτό τον τρόπο αποδεικνύει πως δεν είναι πάντοτε απαραίτητο να χρησιμοποιούνται βαθύτερες αρχιτεκτονικές για καλά αποτελέσματα, αρκεί να αναπτυχθεί και να προσαρμοστεί κατάλληλα ένα ελαφρύ μοντέλο που πετυχαίνει εξίσου καλή δυνατότητα ταξινόμησης [14]. Το ίδιο συμπέρασμα αποτυπώνεται και στην [31] όπου το CNN είναι καλύτερο των VGG-19 [29] και ResNet-50 [3] με αρκετά πιο απλή αρχιτεκτονική.

Μια άλλη προσέγγιση για την βελτίωση της απόδοσης τέτοιων βαθιών αρχιτεκτονικών είναι η ενσωμάτωση τεχνικών παραμορφώσεων των εικόνων εκπαίδευσης, όπως τυχαία περιστροφή της εικόνας, αποκοπή μέρους αυτής, αλλαγές στην φωτεινότητα και το χρώμα. Όταν προστίθεται τέτοια ποικιλία δειγμάτων, το μοντέλο δυσκολεύεται περισσότερο να ταξινομήσει τις εικόνες κατά την εκπαίδευση, επειδή τροποποιούνται τα χαρακτηριστικά χαμηλού επιπέδου (π.χ. ακμές και υφές) της εικόνας. Το μοντέλο αναγκάζεται να μάθει πιο δύσκολα μοτίβα και έτσι εκπαιδεύεται πιο αργά και μπορεί να εμφανίσει λίγο χειρότερη επίδοση στην ακρίβεια εκπαίδευσης. Στον αντίποδα όμως, επειδή αναγκάζεται να μάθει από πιο απαιτητικά δεδομένα, αποκτά καλύτερη ικανότητα γενίκευσης και γίνεται πιο ανθεκτικό σε συνθήκες θορύβου. Επομένως, είναι πιο αποτελεσματικό σε πρακτικές εφαρμογές, που οι εικόνες αντιμετωπίζουν διάφορα είδη παραμορφώσεων. Ταυτόχρονα, η ενσωμάτωση ενός φίλτρου μείωσης ή αποκωδικοποίησης θορύβου μπορεί να ενισχύσει την ακρίβειά του σε πρακτικές εφαρμογές. Τα ευρήματα των Lv et al. [32] υποστηρίζουν αυτή την υπόθεση. Σύμφωνα με την έρευνά τους η προσθήκη ενός αυτόματου κωδικοποιητή μείωσης θορύβου, η χρήση τεχνικών παραμορφώσεων των δεδομένων και η ενσωμάτωση ενός αραιού αυτόματου κωδικοποιητή για την εκμάθηση πιο δύσκολων μοτίβων σε μοντέλο VGG [29] συμβάλλει στην δημιουργία ενός συστήματος ανίχνευσης εικόνων από Gen-AI που είναι ανθεκτικό σε παραμορφώσεις θορύβου και δεν προσαρμόζεται υπερβολικά σε δεδομένα εκπαίδευσης [32].

Επίσης, μια μέθοδος που επιλύει το πρόβλημα του υψηλού κόστους εκπαίδευσης ενός μοντέλου και επιτυγχάνει μεγάλα ποσοστά ακρίβειας είναι η χρήση προεκπαιδευμένων μοντέλων CNN σε άλλες ΒΔ. Η μέθοδος αυτή βασίζεται στην έννοια της μεταφοράς μάθησης (transfer learning) από ένα ειδικό μοντέλο σε ένα άλλο. Σύμφωνα με αυτή είναι δυνατή η μεταφορά γνώσης που αποκτάται από την εκπαίδευση για ένα στόχο σε ένα μοντέλο που έχει να ολοκληρώσει διαφορετικό αλλά παρόμοιο έργο [33].

Στα συνελικτικά δίκτυα σε κάθε συνελικτικό στρώμα γίνεται εκμάθηση διαφορετικών χαρακτηριστικών τα οποία είναι δυνατόν να εξαχθούν ξεχωριστά. Με την ενσωμάτωση αυτών των στρωμάτων ως βάση για ένα νέο μοντέλο και την κατάλληλη αντιγραφή των βαρών των αρχικών στρωμάτων, το νέο μοντέλο χρησιμοποιεί τα προηγούμενα εκπαιδευμένα στρώματα ως μέθοδο εξαγωγής χαρακτηριστικών. Τα χαρακτηριστικά αυτά συνήθως έχουν καλύτερες ιδιότητες, διότι έχουν ήδη δεχθεί κατάλληλη επεξεργασία και με την είσοδό τους στα νέα συμπληρωματικά στρώματα μπορούν να προσαρμοστούν περαιτέρω ανάλογα με τις ανάγκες του νέου στόχου ταξινόμησης. Με αυτόν τον τρόπο γίνεται εξοικονόμηση υπολογιστικών πόρων, διότι τα περισσότερα στρώματα δεν απαιτούν εκ νέου εκπαίδευση και η οπισθοδιάδοση εφαρμόζεται σε περιορισμένο μέρος του δικτύου. Επίσης, σαν βάση σε τέτοια μοντέλα χρησιμοποιούνται συνήθως ειδικά μοντέλα με βαθιές αρχιτεκτονικές, όπως το ResNet [3], που περιέχουν αρκετά ειδικά στρώματα, τα οποία βελτιώνουν την ποιότητα των εξαγόμενων χαρακτηριστικών. Ενδεικτικά παραδείγματα ερευνών που μελετούν συστήματα που βασίζονται σε transfer learning είναι οι έρευνες των Desai et al. [33] και Dewan et al. [34]. Στην έρευνα [33] χρησιμοποιούνται δύο προεκπαιδευμένα μοντέλα VGG-16 και VGG-19 [29] στην ΒΔ ImageNet [35], από τα οποία αφαιρούν το τελικό στρώμα εξόδου που είναι υπεύθυνο για τις τελικές προβλέψεις. Έτσι, χρησιμοποιούν αυτά τα μοντέλα ως εξαγωγείς χαρακτηριστικών τα οποία τροφοδοτούνται έπειτα σε ένα άλλο μοντέλο, που κάνει την τελική πρόβλεψη. Στην περίπτωση του VGG-16 πετυχαίνουν τελική ακρίβεια 90% [33]. Με αυτό τον τρόπο επιβεβαιώνουν την αξία που προσφέρει η χρήση προεκπαιδευμένων μοντέλων για την ταξινόμηση εικόνων με περιορισμένους πόρους. Αντίστοιχα αποτελέσματα προσφέρει και η έρευνα [34], αν και εστιάζει περισσότερο στην σύγκριση διαφορετικών προεκπαιδευμένων μοντέλων ως βάση για τον βέλτιστο ταξινομητή. Σε αυτή την μελέτη όλοι οι ανιχνευτές ανεξαρτήτως της αρχιτεκτονικής που επέλεξαν ως βάση, πετυχαίνουν πολύ υψηλά ποσοστά ακρίβειας με το σύστημα που χρησιμοποιεί το DenseNet-121 [30] να είναι το καλύτερο και να αποκτά μέση ακρίβεια 99.67% [34].

Τέλος, αυτή η μέθοδος με μικρές τροποποιήσεις μπορεί να αποδειχθεί ωφέλιμη στην δημιουργία ενός ανιχνευτή εικόνων Gen-AI που θα λειτουργεί αποτελεσματικά ανεξάρτητα του μέσου παραγωγής των εικόνων. Με την εκπαίδευση πολλαπλών μοντέλων σε διαφορετικές ΒΔ, όπου η κάθε ΒΔ έχει εικόνες που προέρχονται από διαφορετικές γεννήτριες τεχνητής νοημοσύνης, δημιουργούνται πολλά ειδικά δίκτυα που αναγνωρίζουν τις εικόνες με τα ξεχωριστά χαρακτηριστικά της γεννήτριας με την οποία εκπαιδεύτηκαν. Εάν αυτά τα δίκτυα

ενωθούν σε ένα δίκτυο και χρησιμοποιηθούν ξεχωριστά το κάθε ένα ως βάση για το γενικότερο μοντέλο, είναι θεωρητικά δυνατή η δημιουργία ενός καθολικού ανιχνευτή ψεύτικων εικόνων [36]. Η ένωση μπορεί να επιτευχθεί μέσα από την προσθήκη ενός δικτύου με “πύλες”, συνήθως MLP. Κατά την εξαγωγή πρόβλεψης το δίκτυο αυτό αναθέτει δυναμικά βάρη σε κάθε ειδικό μοντέλο ανάλογα με τα χαρακτηριστικά της εικόνας που εξετάζει. Ανάλογα με τα βάρη αυτά στην ουσία αποφασίζει ποιο ειδικό μοντέλο θα επιλέξει για εξαγωγή χαρακτηριστικών και διαμόρφωση τελικής πρόβλεψης. Αυτή η μέθοδος εισάγεται στην έρευνα των Saskorro et al. [36] και τα ευρήματά της αποδεικνύουν πως ένα τέτοιο μοντέλο είναι δυνατόν να ανιχνεύσει εικόνες από κάποιες αρχιτεκτονικές GAN αλλά δυσκολεύεται λίγο σε μοντέλα DM. Αν χρησιμοποιηθεί ενιαίο σύνολο εκπαίδευσης αντί για μικρά υποσύνολα, η απόδοση βελτιώνεται σε άγνωστες γεννήτριες. Αυτό συμβαίνει διότι τα μικρά υποσύνολα πολύ συχνά δεν περιλαμβάνουν όλα τα μοτίβα που υπάρχουν στις εικόνες.

Οι μετασχηματιστές όρασης (Visual Transformer - ViT) μπορούν και αυτοί να χρησιμοποιηθούν ως ταξινομητές εικόνων, όπως τα CNN, με την βασική διαφορά πως χρησιμοποιούν μηχανισμούς προσοχής για να μοντελοποιήσουν σχέσεις στις εικόνες σε μεγάλο βάθος. Χωρίζουν τις πληροφορίες μιας εικόνας σε κομμάτια, τις κωδικοποιούν και μπορούν να μοντελοποιήσουν προσθέτοντας τον χώρο των χαρακτηριστικών ως ενσωματώσεις, επιτρέποντας έτσι να εντοπίσουν σχέσεις που πιθανώς τα CNN να μην ανακάλυπταν. Επιβαρύνονται όμως με μεγαλύτερο κόστος εκπαίδευσης [37]. Η ικανότητά τους αυτή πραγματικά φαίνεται σε μεγάλα σύνολα δεδομένων που επιτρέπουν τα ViT να χρησιμοποιήσουν στο μέγιστο τις ενσωματώσεις τους. Στην έρευνα [37] όπου συνέκριναν την απόδοση ενός κλασσικού CNN με πολλά μοντέλα ViT σε μικρές ΒΔ αποδεικνύεται ότι τα ViT μπορούν να ξεπεράσουν τα CNN, ειδικά όταν ο χρόνος εκπαίδευσης και οι υπολογιστικοί πόροι δεν συνιστούν περιορισμούς.

2.2.3 Ταξινόμηση βάσει ειδικών χαρακτηριστικών εικόνων

Πέρα από τις προηγούμενες μεθόδους που περιγράφηκαν, οι οποίες λειτουργούν κυρίως με χαρακτηριστικά ολόκληρης της εικόνας, υπάρχουν μέθοδοι που εστιάζουν σε φασματικά χαρακτηριστικά και χαρακτηριστικά μεταξύ εικονοστοιχείων. Η χρήση μετρικών, όπως PRNU (Photo Response Non Uniformity) και ELA (Error Level Analysis), είναι ιδιαίτερα δημοφιλής στον τομέα της αναγνώρισης εικόνων από Gen-AI. Το PRNU είναι είδος θορύβου που χρησιμοποιείται για την εύρεση του είδους κάμερας προέλευσης και είναι η μικρή απόκλιση στην ευαισθησία των εικονοστοιχείων του αισθητήρα της κάμερας, η οποία οφείλεται

σε κατασκευαστικές ατέλειες και είναι αναπόφευκτη. Θεωρητικά, όλες οι εικόνες από γεννήτριες δεν θα έπρεπε να εμφανίζουν τέτοιες ατέλειες, αλλά ο υπολογισμός της μετρικής αυτής δίνει πάντα μη μηδενικό αποτέλεσμα [38]. Το ELA είναι τεχνική για τον εντοπισμό εικόνων, συνήθως jpeg, οι οποίες έχουν δεχθεί ψηφιακή επεξεργασία και έτσι υπάρχουν μέρη αυτών με διαφορετικά επίπεδα συμπίεσης [39]. Όταν εφαρμόζεται σε εικόνες από μέσα Gen-AI παρουσιάζει πως όλα τα εικονοστοιχεία έχουν μεταβληθεί [38]. Η έρευνα των Martin-Rodriguez et al. [38] αναπτύσσει δύο CNN, που το ένα βασίζεται σε μετρήσεις PRNU στις εικόνες για εξαγωγή χαρακτηριστικών και το άλλο σε ELA. Αποδεικνύουν πως η χρήση αυτών των τεχνικών και ειδικότερα του ELA είναι αποτελεσματικότερες από την χρήση απλών CNN για την ταξινόμηση εικόνων.

Μια άλλη προσέγγιση που είναι ιδιαίτερα δημοφιλής είναι η ανάλυση φασματικών χαρακτηριστικών και χαρακτηριστικών υφής. Σε εικόνες από γεννήτριες υπάρχουν μεγάλες διαφορές ανάμεσα στις περιοχές με πλούσια και φτωχή υφή, σε αντίθεση με τις αληθινές εικόνες. Με βάση αυτή την παρατήρηση οι Zhong et al. [40] προτείνουν την μέθοδο Smash & Reconstruct προκειμένου να δημιουργήσουν ένα καθολικό αποτύπωμα για εικόνες από Gen-AI, το οποίο δεν εξαρτάται από τις ιδιαιτερότητες της αρχιτεκτονικής του μέσου παραγωγής. Σύμφωνα με αυτή την μέθοδο χωρίζουν την εικόνα σε επιθέματα πλούσια και φτωχά σε υφή και δημιουργούν έπειτα μία εικόνα από την κάθε κατηγορία, αφαιρώντας έτσι σημασιολογικές πληροφορίες. Στην συνέχεια, χρησιμοποιούν υψιπερατά φίλτρα, για να τονίσουν την αντίθεση ανάμεσα στα εικονοστοιχεία των δύο κατηγοριών και να εξάγουν το αποτύπωμα, το οποίο κατόπιν εισάγεται σε δυαδικό ταξινομητή [40]. Αυτή η μέθοδος ενέπνευσε τους Yan et al. [11] που προτείνουν την μέθοδο AIDE (AI-generated Image DEtector with Hybrid Features) και τους Chen et al. [41] που αναπτύσσουν μέθοδο που αξιοποιεί φασματικά χαρακτηριστικά. Η AIDE χρησιμοποιεί ενσωματώσεις σημασιολογικού περιεχομένου, τις οποίες εξάγει από τις εικόνες, και εφαρμόζει μετασχηματισμό διακριτού συνημιτόνου (Discrete Cosine Transform - DCT) για να αναλύσει και να κατατάξει τα επιθέματα με πλούσια και φτωχή υφή. Τέλος, εισάγει τα χαρακτηριστικά σε ένα MLP για την τελική ταξινόμηση [11]. Οι Chen et al. [41] στην μεθοδό τους προσπαθούν να αφαιρέσουν όσα σημασιολογικά χαρακτηριστικά απομένουν από την εφαρμογή Smash & Reconstruct [40] και αναμειγνύουν τα επιθέματα κατά την ανακατασκευή των εικόνων, με σκοπό να αφαιρεθούν τοπικά σημασιολογικά χαρακτηριστικά, όπως ακμές. Κατόπιν, σε αντίθεση με τις άλλες δύο μεθόδους, εφαρμόζουν δισδιάστατο μετασχηματισμό Φουριέ σε όλα τα επιθέματα και αζιμουθιακή ολο-

κλήρωση σε μια συχνότητα . Ταυτόχρονα, χρησιμοποιούν κωδικοποιήσεις θέσεων κατά την ολοκλήρωση για να διατηρήσουν αναλλοίωτη την πληροφορία της τοποθεσίας του επιθέματος στην εικόνα. Τέλος, χρησιμοποιούν μετασχηματιστή όρασης με στόχο την εξαγωγή χαρακτηριστικών, τα οποία εισάγουν σε ένα πλήρες συνδεδεμένο στρώμα που λαμβάνει την τελική απόφαση [41].

Κεφάλαιο 3

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζεται όλο το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση των πειραμάτων της έρευνας. Αρχικά παρουσιάζεται η δομή ενός συνελκτικού νευρωνικού δικτύου και στη συνέχεια η έννοια της υπολειμματικής μάθησης [3]. Κατόπιν παρουσιάζεται το μοντέλο ResNet50 [3] που αξιοποιεί αυτή την ιδιότητα και αποδεικνύεται η μαθηματική του σύνδεση με τα υπόλοιπα συνελκτικά νευρωνικά δίκτυα. Τέλος, γίνεται αναφορά στην μηχανική μάθηση και μαθηματική ανάλυση των μοντέλων μηχανικής μάθησης που χρησιμοποιούνται στην έρευνα.

3.1 Συνελκτικά νευρωνικά δίκτυα

Τα συνελκτικά νευρωνικά δίκτυα είναι το κύριο συστατικό των συστημάτων βαθιάς μάθησης στον τομέα της όρασης υπολογιστών. Αποτελούν ιδιαίτερο είδος νευρωνικού δικτύου με πολλαπλά στρώματα και η λειτουργία τους βασίζεται στον μηχανισμό όρασης έμβιων όντων. Το όνομά τους οφείλεται στα συνελκτικά στρώματα που διαθέτουν. Στα στρώματα αυτά χρησιμοποιούνται διάφορα φίλτρα που εντοπίζουν αντικείμενα μέσα στην εικόνα. Με την χρήση πολλαπλών επιπέδων μπορεί το δίκτυο να εξάγει βασικά χαρακτηριστικά και από τα εικονοστοιχεία και από μεγαλύτερες περιοχές της εικόνας. Με αυτό τον τρόπο εξάγονται από τα δεδομένα εισόδου πίνακες χαρακτηριστικών οι οποίοι χρησιμοποιούνται στην τελική ταξινόμηση [42]. Μαθηματικά η διαδικασία αυτή για ένα τυχαίο συνελκτικό νευρωνικό δίκτυο μπορεί να εκφραστεί ως εξής:

Έστω ότι η εικόνα εισόδου είναι η:

$$\mathbf{I} \in \mathbb{R}^{H \times W \times C} \quad (3.1)$$

όπου H είναι το ύψος, W το πλάτος και C ο αριθμός καναλιών της εικόνας.

Η διαδικασία προώθησης (forward pass) μέσα από ένα Συνελικτικό Νευρωνικό Δίκτυο για ταξινόμηση εικόνας είναι:

$$\mathbf{x}^{(l)} = f^{(l)} (\mathbf{W}^{(l)} * \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, 2, \dots, L \quad (3.2)$$

Με αρχική τιμή:

$$\mathbf{x}^{(0)} = \mathbf{I} \quad (3.3)$$

όπου:

- $*$ είναι η συνελικτική πράξη (convolution),
- $\mathbf{W}^{(l)}$ τα βάρη του l -στρώματος,
- $\mathbf{b}^{(l)}$ όροι μετατόπισης,
- $f^{(l)}(\cdot)$ η μη γραμμική συνάρτηση ενεργοποίησης (π.χ Rectified Linear Unit, ReLU)
- L είναι ο συνολικός αριθμός στρωμάτων.

Μετά από όλα τα συνελικτικά στρώματα και στρώματα υποδειγματοληψίας, πραγματοποιείται μετατροπή σε διάνυσμα:

$$\mathbf{h} = \text{Flatten}(\mathbf{x}^{(L)}) \quad (3.4)$$

Έπειτα, ακολουθεί πλήρως συνδεδεμένο στρώμα (fully connected layer):

$$\mathbf{z} = \mathbf{W}^{(fc)} \mathbf{h} + \mathbf{b}^{(fc)} \quad (3.5)$$

Τέλος, η έξοδος ταξινόμησης $\mathbf{y}$ προκύπτει συνήθως μέσω της πράξης Softmax:

$$\mathbf{y} = \text{Softmax}(\mathbf{z}) \quad (3.6)$$

δηλαδή:

$$y_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}, \quad i = 1, 2, \dots, K \quad (3.7)$$

όπου K είναι ο αριθμός των κλάσεων.

3.2 Μοντέλο ResNet50

Το μοντέλο ResNet50 [3] είναι ένα συνελικτικό νευρωνικό δίκτυο με βαθιά αρχιτεκτονική που στηρίζεται στην υπολειμματική μάθηση (residual learning), προκειμένου να αντιμετωπίσει το πρόβλημα μείωσης αποδοτικότητας σε βαθιές αρχιτεκτονικές.

3.2.1 Υπολειμματική μάθηση

Με την ανάπτυξη μοντέλων CNN με ολοένα και πιο βαθιές αρχιτεκτονικές παρατηρείται ένα καινούριο πρόβλημα μείωσης απόδοσης. Η ταχύτητα με την οποία ένα μοντέλο συγκλίνει σε μία τιμή ακριβείας κατά την εκπαίδευσή του εξαρτάται άμεσα από το βάθος του μοντέλου, καθώς έχει ανακαλυφθεί πως βαθύτερες αρχιτεκτονικές συγκλίνουν ταχύτερα καλύπτοντας τις δυνατότητες μάθησης του μοντέλου. Μάλιστα, αφού φτάσει στην μέγιστη απόδοσή του το μοντέλο, ακολουθεί μια κατακόρυφη πτώση της ακριβείας του και ο ρυθμός αυτής της πτώσης εξαρτάται επίσης από την πολυπλοκότητα του μοντέλου. Αυτές οι απώλειες δεν οφείλονται στην υπερβολική προσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης, αλλά στην δυσκολία βελτιστοποίησης τέτοιων συστημάτων [3].

Λύση σε αυτό το φαινόμενο φαίνεται να προσφέρει η υπολειμματική μάθηση. Σύμφωνα με αυτή, δεν μαθαίνει κάθε μέρος του δικτύου έναν επιθυμητό μετασχηματισμό $\mathcal{H}(\mathbf{x})$, αλλά μαθαίνει ένα υπόλοιπο, γνωστό και ως residual, $\mathcal{F}(\mathbf{x}, \mathbf{W})$, δηλαδή τη διαφορά μεταξύ της εισόδου και της επιθυμητής εξόδου, όπως φαίνεται στην μαθηματική αναπαράσταση που ακολουθεί:

Έστω ότι $\mathbf{x}^{(l-1)}$ είναι η είσοδος σε ένα στρώμα υπολειμματικής μάθησης (residual block) l . Αυτό μαθαίνει μια συνάρτηση $\mathcal{F}^{(l)}(\mathbf{x}^{(l-1)}, \mathbf{W}^{(l)})$, όπου $\mathbf{W}^{(l)}$ είναι τα εκπαιδευόμενα βάρη του στρώματος. Τότε, η έξοδος $\mathbf{x}^{(l)}$ του block είναι η:

$$\mathbf{x}^{(l)} = \mathcal{F}^{(l)}(\mathbf{x}^{(l-1)}, \mathbf{W}^{(l)}) + \mathbf{x}^{(l-1)}, \quad l = 1, 2, \dots, L \quad (3.8)$$

όπου:

- $\mathcal{F}^{(l)}(\mathbf{x}^{(l-1)}, \mathbf{W}^{(l)})$: είναι η απεικόνιση που μαθαίνει το residual block, που συνήθως περιλαμβάνει δύο ή περισσότερες συνελκτικές στρώσεις, κανονικοποίηση και μη γραμμική συνάρτηση ενεργοποίησης.
- $\mathbf{x}^{(l-1)}$: είναι η είσοδος του block

Αν οι διαστάσεις της εξόδου $\mathbf{x}^{(l)}$ και της εισόδου $\mathbf{x}^{(l-1)}$ διαφέρουν, τότε χρησιμοποιείται μια συντόμευση-προβολή, γνωστή και ως projection shortcut, μέσω του γραμμικού μετασχηματισμού:

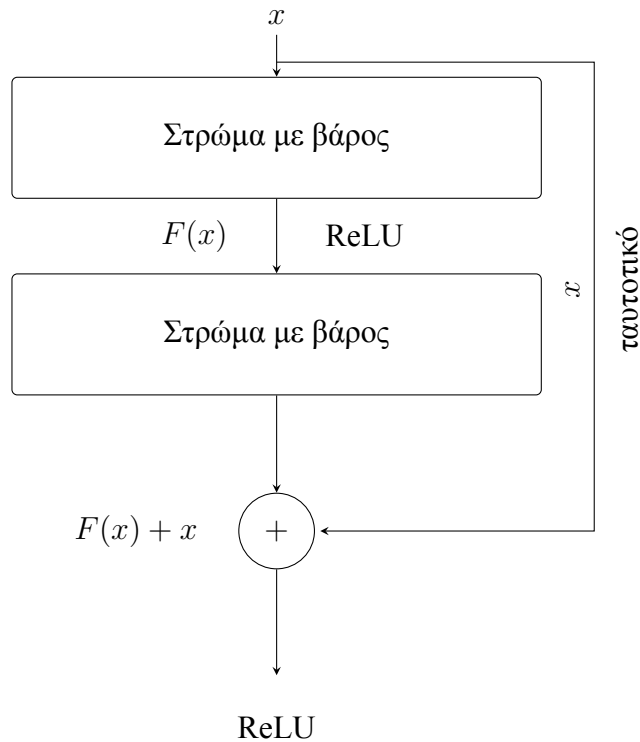
$$\mathbf{x}^{(l)} = \mathcal{F}^{(l)}(\mathbf{x}^{(l-1)}, \mathbf{W}^{(l)}) + \mathbf{W}_s^{(l)} \mathbf{x}^{(l-1)} \quad (3.9)$$

όπου $\mathbf{W}_s^{(l)}$ είναι τα βάρη της προβολής (συνήν 1×1 συνελίξεις).

Η κλασική διαδικασία προώθησης όπως αναφέρθηκε στην εξίσωση 3.2 μπορεί να επεκταθεί, ώστε να συμπεριλάβει την υπολειμματική μάθηση, δηλαδή να συμπεριληφθεί στο residual block η πρόσθεση της εισόδου. Έστω ότι το $\mathcal{F}^{(l)}(\cdot)$ ακολουθεί ίδια μορφή με την εξίσωση 3.2. Τότε, έχουμε:

$$\mathbf{x}^{(l)} = f^{(l)} (\mathbf{W}^{(l)} * \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}) + \mathbf{x}^{(l-1)} \quad (3.10)$$

Με αυτόν τον τρόπο, η υπολειμματική μάθηση επιτρέπει στο δίκτυο να μαθαίνει πιο εύκολα τις διαφορές.

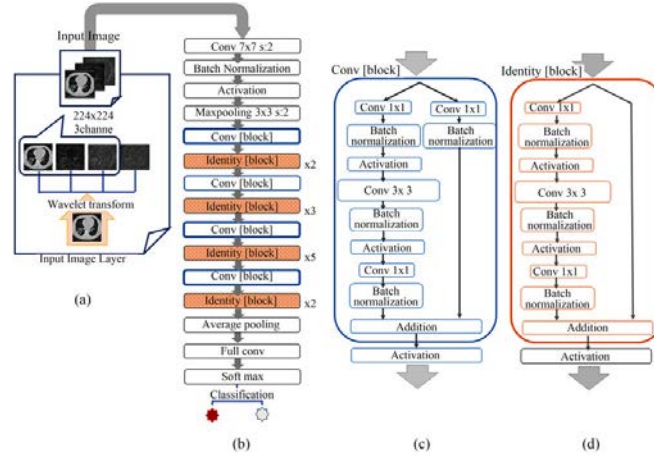


Σχήμα 3.1: Η δομή ενός στρώματος υπολειμματικής μάθησης residual block (σχήμα προσαρμοσμένο από [3]).

3.2.2 Δομή ResNet50

Το ResNet50 [3] αρχικά δέχεται ως είσοδο εικόνες μεγέθους 224×224 εικονοστοιχείων με 3 κανάλια χρώματος. Στην αρχή η εικόνα περνάει από ένα συνελκτικό στρώμα με πυρήνα 7×7 και βήμα 2. Έπειτα, πραγματοποιείται κανονικοποίηση, χρησιμοποιείται συνάρτηση ενεργοποίησης ReLU και γίνεται υποδειγματοληψία με πυρήνα 3×3 και βήμα 2. Μετά το πέρας αυτού του “σταδίου” ακολουθούν διαδοχικά και εναλλάξ ομάδες από συνελκτικά στρώματα (convolutional blocks) και ομάδες από στρώματα που προσφέρουν απεικόνιση για

την ταυτοτική είσοδο (identity blocks). Στο τέλος, υπάρχει ένα στρώμα υποδειγματοληψίας και ένα τελικό πλήρως συνδεδεμένο στρώμα και στο αποτέλεσμα εφαρμόζεται η συνάρτηση κανονικοποίησης softmax [3], [4].



Σχήμα 3.2: Η δομή του ResNet50. Το a) περιγράφει την επεξεργασία της εικόνας κατά την είσοδο. Το b) περιγράφει την συνολική δομή του ResNet50 και τα σύμβολα $\times 2$, $\times 3$, $\times 5$ αντιπροσωπεύουν τον αριθμό των στρωμάτων. Το c) δείχνει την αρχιτεκτονική των convolutional blocks όπου οι διαστάσεις εισόδου διαφέρουν και το d) δείχνει την αρχιτεκτονική των identity blocks όπου οι διαστάσεις εισόδου είναι σταθερές (σχήμα από [4]).

Το μοντέλο ακολουθεί την γενική αρχιτεκτονική και λογική προώθησης ενός CNN, όπως περιγράφεται στην εξίσωση 3.2, όμως στη συγκεκριμένη περίπτωση η προώθηση πραγματοποιείται μέσω residual blocks, όπως περιγράφεται από τις εξισώσεις 3.8 και 3.9.

Η είσοδος $\mathbf{I} \in \mathbb{R}^{224 \times 224 \times 3}$ περνά αρχικά από ένα συνελκτικό επίπεδο 7×7 με βήμα 2 και κατάλληλη συμπλήρωση της εισόδου με μηδενικά (padding). Η πράξη της συνέλιξης για τρισδιάστατα διανύσματα ορίζεται ως εξής:

$$Y_{i,j,k} = \sum_{n=1}^N \sum_u \sum_v X_{i+u,j+v,n} \cdot W_{u,v,n,k} + b_k \quad (3.11)$$

όπου:

- X : Ο τρισδιάστατος πίνακας εισόδου,
- W : τα φίλτρα συνέλιξεων (πυρήνες),
- b_k : οι μετατοπίσεις της εξόδου για το κανάλι k ,
- N : ο αριθμός καναλιών,

- i, j : χωρικές συντεταγμένες γραμμής και στήλης,
- k : δείκτης καναλιού εξόδου.

Στη συνέχεια εφαρμόζεται κανονικοποίηση (batch normalization):

$$\hat{x} = \frac{x - \mu_{\text{batch}}}{\sqrt{\sigma_{\text{batch}}^2 + \epsilon}} \quad (3.12)$$

και η τελική έξοδος της κανονικοποίησης είναι:

$$\text{BN}(x) = \gamma \hat{x} + \beta \quad (3.13)$$

Μετά, χρησιμοποιείται ReLU:

$$f(x) = \max(0, x) \quad (3.14)$$

Τέλος εφαρμόζεται συγκέντρωση μεγίστων (max pooling):

$$Y_{i,j,k} = \max_{(u,v) \in \text{window}} X_{i+u,j+v,k} \quad (3.15)$$

όπου το παράθυρο, window, είναι συνήθως μεγέθους 3×3 με βήμα 2.

Έπειτα ακολουθούν τα εναλλάξ επίπεδα convolution και identity, όπως φαίνεται και στο σχήμα 3.2. Τα Convolutional Blocks λόγω διαφορετικών διαστάσεων εισόδου χρησιμοποιούν την εξίσωση 3.9 για να γίνει προβολή στις διαστάσεις, ενώ τα Identity Blocks ακολουθούν την εξίσωση 3.8. Η εσωτερική απεικόνιση $\mathcal{F}^{(l)}$ κάθε μπλοκ περιγράφεται από την εξίσωση 3.10.

Τέλος, ακολουθεί υποδειγματοληψία (global average pooling):

$$y_k = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W x_{i,j,k} \quad (3.16)$$

Ακολουθεί το πλήρως συνδεδεμένο στρώμα, όπως περιγράφεται στην εξίσωση 3.5 και τέλος με την εφαρμογή της softmax 3.7 παράγεται η έξοδος.

Άρα συνοψίζοντας, η λογική της προώθησης του ResNet50 [3] είναι προέκταση της λειτουργίας προώθησης των CNN με την ενσωμάτωση υπολειμματικών συνδέσεων.

3.3 Μηχανική μάθηση για ταξινόμηση εικόνων

3.3.1 Μηχανές διανυσματικής υποστήριξης

Τα SVMs αναπτύχθηκαν στα πλαίσια της θεωρίας της στατιστικής μάθησης και έχουν καθιερωθεί ως ένα από τα πιο δημοφιλή μέσα μηχανικής μάθησης αφού χρησιμοποιούνται

σε πολλούς τομείς και ιδίως στην όραση υπολογιστών. Τα SVMs και κυρίως οι ταξινομητές SVM προσπαθούν να επιλύσουν το πρόβλημα μάθησης βρίσκοντας ένα βέλτιστο υπερεπίπεδο, τα όρια του οποίου ισαπέχουν από τους δύο τουλάχιστον χώρους διανυσμάτων [22], [43]. Προκειμένου να επιτευχθεί αυτό, τα σημεία του συνόλου ταξινομούνται σε διαφορετικές κλάσεις και υπολογίζεται το περιθώριο μεταξύ των κλάσεων. Με αυτό τον τρόπο παράγονται διανύσματα που οριοθετούν και στηρίζουν την περιοχή μεταξύ των περιθωρίων. Η περιοχή αυτή ονομάζεται margin region και τα διανύσματα αυτά support vectors. Στην περίπτωση που οι κλάσεις είναι περισσότερες από δύο μπορεί να χρησιμοποιηθεί η προσέγγιση “one versus all”, κατά την οποία μια κλάση συγκρίνεται και οριοθετείται σε σύγκριση με τις υπόλοιπες. Τα support vectors των σημείων των δεδομένων που είναι πλησιέστερα στο υπερεπίπεδο διαχωρισμού καθορίζουν την θέση και κατεύθυνση του υπερεπιπέδου [22].

Γενικά ένα SVM βρίσκει ένα υπερεπίπεδο που είναι δυνατόν να ανήκει σε διαφορετικό χώρο από αυτόν των δεδομένων. Αυτός ο χώρος παράγεται από έναν πυρήνα, ο οποίος ορίζει το εσωτερικό γινόμενο των διανυσμάτων στο χώρο. Μέσω του πυρήνα ο χώρος της υπόθεσης ορίζεται ως ένα σύνολο υπερεπιπέδων στον χώρο χαρακτηριστικών που επάγεται από τον πυρήνα [43]. Υπάρχουν διάφοροι πυρήνες και σε αυτήν την εργασία χρησιμοποιούνται δύο είδη, ο γραμμικός (linear) πυρήνας και η συνάρτηση ακτινωτής βάσης (Radial Basis Function - RBF). Στο πρώτο στάδιο του τρίτου πειράματος υπάρχει μόνο SVM με linear πυρήνα, καθώς ο πυρήνας RBF απαιτεί υπερβολικά μεγάλο αριθμό πράξεων και δεν επαρκούσαν οι πόροι για τους απαραίτητους υπολογισμούς.

Ο υπολογισμός υπερεπιπέδου με γραμμικό πυρήνα αποτελεί την πιο απλή περίπτωση για ένα SVM. Μαθηματικά ισχύει αρχικά για την δυαδική ταξινόμηση:

Έστω σύνολο εκπαίδευσης:

$$Dataset = \{ (\mathbf{x}^{(i)}, y_i) \mid \mathbf{x}^{(i)} \in \mathbb{R}^d, y_i \in \{-1, +1\} \}_{i=1}^n$$

και γραμμικός πυρήνας K :

$$K(x_1, x_2) = x_1^\top x_2 \quad (3.17)$$

Η συνάρτηση απόφασης του linear SVM είναι:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (3.18)$$

υπό τους περιορισμούς:

$$y_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (3.19)$$

Το περιθώριο είναι:

$$\gamma = \frac{2}{\|\mathbf{w}\|} \quad (3.20)$$

Έτσι, για τα support vectors ισχύει:

$$y_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b) = 1 \quad \text{ή} \quad \xi_i > 0 \quad (3.21)$$

Η συνάρτηση κόστους ορίζεται ως:

$$\mathcal{L}_{\text{hinge}}(y_i, \hat{y}_i) = \max(0, 1 - y_i \cdot \hat{y}_i) = \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b)) \quad (3.22)$$

Επεκτείνοντας την υλοποίηση για κλάσεις με μέθοδο one vs. all ισχύει:

Για M κλάσεις εκπαιδεύονται M ανεξάρτητα γραμμικά μοντέλα. Για κάθε κλάση $m \in \{1, \dots, M\}$ επανακωδικοποιούνται οι ετικέτες ως εξής:

$$y_i^{(m)} = \begin{cases} +1 & \text{αν } y_i = m \\ -1 & \text{αν } y_i \neq m \end{cases}$$

Πλέον, το κάθε SVM λύνει το εξής πρόβλημα:

$$\min_{\mathbf{w}_m, b_m, \xi^{(m)}} \frac{1}{2} \|\mathbf{w}_m\|^2 + C \sum_{i=1}^n \xi_i^{(m)} \quad (3.23)$$

με περιορισμούς:

$$y_i^{(m)} (\mathbf{w}_m^\top \mathbf{x}^{(i)} + b_m) \geq 1 - \xi_i^{(m)}, \quad \xi_i^{(m)} \geq 0 \quad (3.24)$$

Επομένως, η τελική πρόβλεψη για νέο δείγμα $\mathbf{x} \in \mathbb{R}^d$ είναι:

$$\hat{y} = \arg \max_{m \in \{1, \dots, M\}} (\mathbf{w}_m^\top \mathbf{x} + b_m) \quad (3.25)$$

Ακολουθεί η μαθηματική περιγραφή για δυαδικό ταξινομητή SVM με πυρήνα RBF. Ορίζεται η συνάρτηση πυρήνα ως:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2), \quad \gamma > 0 \quad (3.26)$$

Πλέον το πρόβλημα δυαδικής ταξινόμησης είναι:

$$\max_{\lambda} \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad (3.27)$$

υπό τους περιορισμούς:

$$0 \leq \lambda_i \leq C, \quad \sum_{i=1}^n \lambda_i y_i = 0 \quad (3.28)$$

Έτσι, η συνάρτηση απόφασης για νέο δείγμα $\mathbf{x} \in \mathbb{R}^d$ προκύπτει από:

$$f(\mathbf{x}) = \sum_{i=1}^n \lambda_i y_i K(\mathbf{x}^{(i)}, \mathbf{x}) + b \quad (3.29)$$

και η πρόβλεψη:

$$\hat{y} = \text{sign}(f(\mathbf{x})) \quad (3.30)$$

Επεκτείνοντας την υλοποίηση για M κλάσεις, εκπαιδεύονται M ανεξάρτητα μοντέλα SVM και για κάθε μοντέλο $m \in \{1, \dots, M\}$ επανακωδικοποιούνται και οι ετικέτες όπως και για Linear SVM με M κλάσεις.

Κάθε SVM λύνει το πρόβλημα:

$$\max_{\lambda^{(m)}} \sum_{i=1}^n \lambda_i^{(m)} - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i^{(m)} \lambda_j^{(m)} y_i^{(m)} y_j^{(m)} K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad (3.31)$$

με περιορισμούς:

$$0 \leq \lambda_i^{(m)} \leq C, \quad \sum_{i=1}^n \lambda_i^{(m)} y_i^{(m)} = 0 \quad (3.32)$$

Έτσι η συνάρτηση διαχωρισμού για κάθε κλάση είναι:

$$f_m(\mathbf{x}) = \sum_{i=1}^n \lambda_i^{(m)} K(\mathbf{x}^{(i)}, \mathbf{x}) + b_m \quad (3.33)$$

Η τελική πρόβλεψη ισούται με:

$$\hat{y} = \arg \max_{m \in \{1, \dots, M\}} f_m(\mathbf{x}) \quad (3.34)$$

3.3.2 Random Forest ταξινομητές

Τα Random Forest είναι ένα διάσημο είδος ταξινομητή που πετυχαίνει υψηλή ακρίβεια ταξινόμησης και προκύπτει από την δημιουργία ενός συνδυαστικού μοντέλου που συνδυάζει πολλά δέντρα απόφασης, τα οποία έπειτα ψηφίζουν για την πιο πιθανή κλάση. Για να προστεθούν δέντρα απόφασης και να μεγαλώσει το μέγεθος της δομής, πολύ συχνά παράγονται τυχαία διανύσματα, τα οποία καθορίζουν την προσθήκη δέντρων. Πάντοτε το κάθε νέο τυχαίο διάνυσμα που παράγεται για κάθε νέο δέντρο είναι ανεξάρτητο από παλιά δέντρα και τυχαία διανύσματα αλλά ακολουθεί την ίδια κατανομή δεδομένων. Όλα τα δέντρα, επομένως, εκπαιδεύονται πάνω στο ίδιο σύνολο εκπαίδευσης και αναπτύσσονται με βάση αυτό. Όταν έχει παραχθεί μεγάλος αριθμός από δέντρα τότε ψηφίζουν για την πιο δημοφιλή κλάση [44].

3.3.3 XGBoost ταξινομητές

Το XGBoost βασίζεται στην σταδιακή ενίσχυση αδύναμων μοντέλων, με στόχο την αύξηση της απόδοσης μέσα από την βελτιστοποίηση μιας παραγωγίσιμης συνάρτησης κόστους. Όπως και στην ενότητα 3.3.1 ορίζουμε το σύνολο δεδομένων εκπαίδευσης.

Το μοντέλο αποτελείται από T δέντρα:

$$\hat{y}_i = \sum_{t=1}^T f_t(\mathbf{x}^{(i)}), \quad f_t \in \mathcal{F} \quad (3.35)$$

όπου $\mathcal{F}$ είναι το σύνολο όλων των δυνατών δέντρων απόφασης. Η συνάρτηση διαχωρισμού αποτελείται από τον μέσο όρο της απώλειας και έναν όρο κανονικοποίησης:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n \text{loss}\left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}^{(i)})\right) + \Omega(f_t) \quad (3.36)$$

Για τον κανονικοποιητή $\Omega(f_t)$ ισχύει:

$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

όπου:

- T : αριθμός φύλλων του δέντρου,
- w_j : τιμή εξόδου στο φύλλο j

Η συνάρτηση κόστους υπολογίζεται με ανάπτυγμα Taylor δεύτερης τάξης:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n \left[g_i f_t(\mathbf{x}^{(i)}) + \frac{1}{2} h_i f_t(\mathbf{x}^{(i)})^2 \right] + \Omega(f_t) \quad (3.37)$$

όπου:

- $g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$: είναι η πρώτη παράγωγος
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2}$: είναι η δεύτερη παράγωγος

Η βέλτιστη έξοδος w_j για κάθε φύλλο j υπολογίζεται ως:

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (3.38)$$

και η συνεισφορά προς τον τελικό στόχο:

$$\mathcal{L}_{\text{leaf}} = - \frac{1}{2} \sum_{j=1}^T \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \quad (3.39)$$

όπου I_j είναι το σύνολο δειγμάτων στο φύλλο j . Για περισσότερες από 2 κλάσεις το XGBoost εφαρμόζει softmax και επιλέγει την πρόβλεψη με την μεγαλύτερη πιθανότητα.

Κεφάλαιο 4

Προετοιμασία δεδομένων

Στο κεφάλαιο αυτό θα εξεταστούν τα δεδομένα που χρησιμοποιήθηκαν στα πειράματα, η αναγκαία προεπεξεργασία των δεδομένων, προκειμένου να τα χρησιμοποιήσει το μοντέλο, καθώς και η δημιουργία ενός διαχειριστή δεδομένων, ο CIFAKEDatasetHandler.

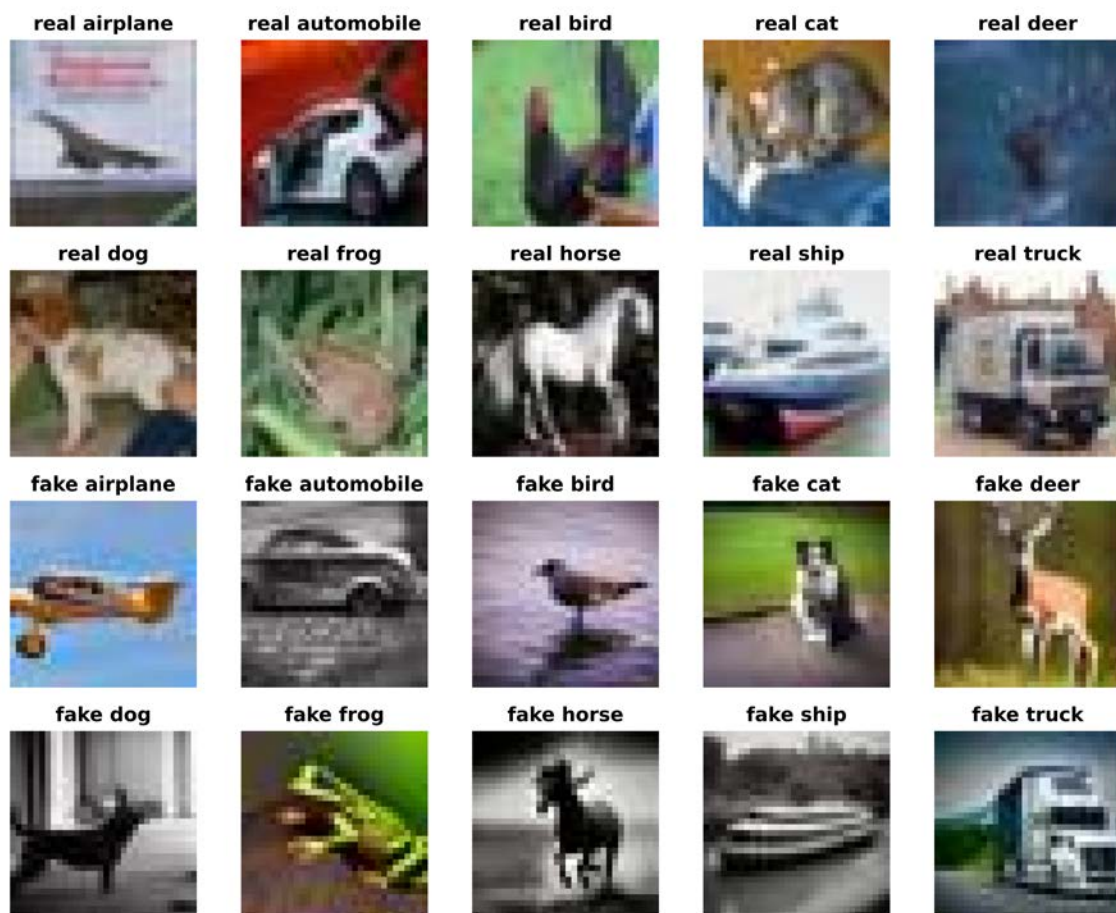
4.1 Βάση δεδομένων CIFAKE

Η βάση δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση, τον έλεγχο και την δοκιμή του μοντέλου είναι η CIFAKE [5], [6]. Η βάση αυτή περιέχει 120000 εικόνες, 60000 εικόνες από την βάση CIFAR-10 [6] και 60000 αντίστοιχες συνθετικές εικόνες από διάφορα μέσα παραγωγικής τεχνητής νοημοσύνης, όπως μοντέλα διάχυσης. Όλες οι εικόνες είναι της μορφής jpeg και έχουν μέγεθος 32×32 εικονοστοιχεία με 3 κανάλια χρώματος (RGB - κόκκινο, πράσινο και μπλε). Η βάση περιέχει 100000 εικόνες για εκπαίδευση και 20000 εικόνες για δοκιμή μοντέλου.

Είναι σημαντικό να αναφερθεί πως αρχικά η βάση δεδομένων έχει ετικέτες μόνο για τις 2 κλάσεις real και fake, αλλά παρόλα αυτά, επειδή η βάση στηρίζεται στο CIFAR-10 [6], ο αριθμός των εικόνων σε κάθε κλάση είναι ισορροπημένος και ισάριθμα κατανομημένος μεταξύ των κλάσεων της βάσης CIFAR-10[6]. Τα περιεχόμενα της βάσης δεδομένων CIFAKE παρουσιάζονται στον πίνακα 4.1.

Έπειτα, προκειμένου, να ελεγχθεί η πορεία του μοντέλου κατά την εκπαίδευση και να αποφευχθεί overfitting, είναι αναγκαίο να προστεθεί ρουτίνα ελέγχου και αντίστοιχα σύνολο ελέγχου που δεν υπάρχει στο αρχικό σύνολο. Υποχρεωτικά χωρίστηκε το σύνολο εκπαίδευσης εκ νέου σε ένα μικρότερο σύνολο εκπαίδευσης και σε ένα σύνολο ελέγχου. Ο διαχωρι-

σμός έγινε με λόγο 80/20, δηλαδή το νέο σύνολο εκπαίδευσης περιέχει το 80% των εικόνων του αρχικού (80000 εικόνες) και το σύνολο ελέγχου περιέχει το υπόλοιπο 20% δηλαδή 20000 εικόνες (στον πίνακα 4.2 παρουσιάζεται η νέα κατανομή των εικόνων). Προκειμένου να εκπαιδευτεί και να ελεγχθεί το μοντέλο δίκαια και σωστά, πρέπει η κατανομή των εικόνων στους διάφορους συνδυασμούς κλάσεων να παραμείνει ίδια με την αρχική. Επομένως όλες οι κλάσεις έχουν ίδιο αριθμό εικόνων στην εκπαίδευση, στον έλεγχο και στην δοκιμή. Ο κώδικας που πραγματοποιεί την διαδικασία ονοματοδοσίας των κλάσεων της βάσης δεδομένων, περιγράφεται στο σχήμα Β.1. Τέλος, στο σχήμα 4.1 παρουσιάζεται μια εικόνα από κάθε δυνατό συνδυασμό κλάσεων.



Σχήμα 4.1: Στο σχήμα αυτό παρουσιάζεται ένα παράδειγμα από κάθε δυνατό συνδυασμό κλάσεων για τη ΒΔ CIFAKE[5, 6]. Οι εικόνες είναι μεγέθους 32×32 και επομένως χαμηλής ποιότητας.

Πίνακας 4.1: Η κατανομή των εικόνων στην βάση CIFAKE [5], [6]. Στην στήλη κλάση για την ονομασία κάθε κλάσης πραγματοποιήθηκε το καρτεσιανό γινόμενο των δύο διαφορετικών κατηγοριών κλάσεων, όπου η 1η κατηγορία είναι real ή fake και η 2η είναι οι 10 κλάσεις του CIFAR-10 [6].

Κλάση	αριθμός εικόνων εκπαίδευσης	αριθμός εικόνων δοκιμής
real airplane	5000	1000
fake airplane	5000	1000
real automobile	5000	1000
fake automobile	5000	1000
real bird	5000	1000
fake bird	5000	1000
real cat	5000	1000
fake cat	5000	1000
real deer	5000	1000
fake deer	5000	1000
real dog	5000	1000
fake dog	5000	1000
real frog	5000	1000
fake frog	5000	1000
real horse	5000	1000
fake horse	5000	1000
real ship	5000	1000
fake ship	5000	1000
real truck	5000	1000
fake truck	5000	1000

Πίνακας 4.2: Η κατανομή των εικόνων στην νέα βάση CIFAKE [5], [6].

Κλάση	# εικόνων εκπαίδευσης	# εικόνων ελέγχου	# εικόνων δοκιμής
real airplane	4000	1000	1000
fake airplane	4000	1000	1000
real automobile	4000	1000	1000
fake automobile	4000	1000	1000
real bird	4000	1000	1000
fake bird	4000	1000	1000
real cat	4000	1000	1000
fake cat	4000	1000	1000
real deer	4000	1000	1000
fake deer	4000	1000	1000
real dog	4000	1000	1000
fake dog	4000	1000	1000
real frog	4000	1000	1000
fake frog	4000	1000	1000
real horse	4000	1000	1000
fake horse	4000	1000	1000
real ship	4000	1000	1000
fake ship	4000	1000	1000
real truck	4000	1000	1000
fake truck	4000	1000	1000

4.2 Διαχειριστής δεδομένων CIFAKEDatasetHandler

Η βιβλιοθήκη PyTorch [45] δεν παρέχει κάποιο διαχειριστή δεδομένων που να είναι σχεδιασμένος για την βάση δεδομένων της έρευνας. Επομένως, δημιουργείται η ανάγκη να κατασκευαστεί ένας νέος διαχειριστής δεδομένων dataset που να περιγράφει επαρκώς τη βάση δεδομένων, να καλύπτει τις ανάγκες των δεδομένων και να είναι σύμφωνη με τις προδιαγραφές των διαχειριστών δεδομένων `utils.data.Dataset` όπως τις ορίζει το PyTorch. Ο διαχειριστής δεδομένων `CIFAKEDatasetHandler` περιέχει τρεις συναρτήσεις, την `__init__`, την `__len__` και την `__getitem__`.

Η συνάρτηση `__init__` αρχικοποιεί σημαντικές παραμέτρους και χρησιμοποιείται για να διαπεραστούν οι φάκελοι που περιέχουν όλες τις εικόνες της βάσης. Δέχεται σαν όρισμα ένα string, το `root_directory` που αποτελεί το μονοπάτι του φακέλου, ο οποίος περιέχει τις εικόνες, και ένα όρισμα `transform` που είναι οι μεταχηματισμοί που θα εφαρμοστούν στις εικόνες. Αρχικά, επειδή η βάση περιέχει μόνο ετικέτες για την δυαδική ταξινόμηση, πρέπει να δημιουργηθούν και να τοποθετηθούν σωστά στις εικόνες οι κατάλληλες ετικέτες του CIFAR-10[6]. Για αυτό το λόγο με αφετηρία τον φάκελο `root_directory` διαπερνώνται όλοι οι υποφάκελοι, όπου ο καθένας περιέχει όλες τις εικόνες μιας κλάσης και δημιουργούνται δυναμικά οι πίνακες εικόνων και οι ετικέτες. Οι ετικέτες αποθηκεύονται σε λεξικά για γρήγορη πρόσβαση και ανάκτηση τιμής. Στο τέλος της `__init__` έχει δημιουργηθεί το πλήρες σύνολο δεδομένων με τις σωστές ετικέτες για την κάθε εικόνα. Ο αλγόριθμος 1 περιγράφει την λειτουργία της συνάρτησης `__init__`.

Η συνάρτηση `__len__` είναι μια απλή βοηθητική συνάρτηση που επιστρέφει το μήκος του πίνακα των εικόνων, δηλαδή το μέγεθος του συνόλου δεδομένων. Η συνάρτηση `__getitem__` δέχεται ως όρισμα ένα αριθμό `index`. Με αυτόν βρίσκει το μονοπάτι και τις ετικέτες της εικόνας η οποία βρίσκεται στην θέση `index` της λίστας. Έπειτα προσπαθεί να ανοίξει την εικόνα και εάν το επιτύχει, εφαρμόζει μετασχηματισμούς στην εικόνα, εφόσον υπάρχουν μετασχηματισμοί. Εάν δεν επιτύχει να ανοίξει την εικόνα, εμφανίζει μήνυμα λάθους και καλεί αναδρομικά τον εαυτό της για την επόμενη εικόνα στην λίστα. Η υλοποίηση των συναρτήσεων `__init__`, `__len__` και `__getitem__` παρουσιάζεται στο σχήμα B.2

Algorithm 1 Η αρχικοποίηση του CIFAKEDatasetHandler

Require: Αρχικό φάκελο με υποφακέλους *real* και *fake*, έκαστος με 10 υποφακέλους-κλάσεις.

```

1: Αρχικοποίηση λίστας εικόνων: images  $\leftarrow$  []
2: Ορισμός μετασχηματισμών: transform
3: Δημιουργία λεξικού ετικετών για την πρώτη ετικέτα: first_level_label_mapping  $\leftarrow$  {real  $\rightarrow$  0, fake  $\rightarrow$  1}
4: Δημιουργία κενού λεξικού για την δεύτερη ετικέτα: class_to_idx  $\leftarrow$  {}
5: for first_label in {real, fake} do
6:   path_first_level  $\leftarrow$  root_directory / first_label
7:   for class_name στο path_first_level do
8:     if class_name δεν είναι φάκελος then
9:       Συνέχισε στο επόμενο
10:    end if
11:    if class_name δεν υπάρχει στο class_to_idx then
12:      class_to_idx[class_name]  $\leftarrow$  τρέχων αριθμός κλάσεων
13:    end if
14:    for αρχείο fname μέσα στον φάκελο class_name do
15:      if το fname τελειώνει σε .png, .jpeg, .jpg then
16:        fpath  $\leftarrow$  Πλήρης διαδρομή αρχείου
17:        εγγραφή  $\leftarrow$  (fpath, first_level_label_mapping[first_label],
18:          class_to_idx[class_name])
19:        Πρόσθεσε την εγγραφή στην λίστα εικόνων
20:      end if
21:    end for
22:  end for
23: end for
24: Αποθήκευσε το class_to_idx

```

Ensure: Έτοιμη λίστα εικόνων με 2 ετικέτες ανά εικόνα

4.3 Μετασχηματισμοί, σύνολα δεδομένων και dataloaders

Αφού ορίστηκε ο διαχειριστής δεδομένων, πρέπει να οριστούν και οι μετασχηματισμοί για τις εικόνες. Αρχικά, αποθηκεύτηκαν ο μέσος όρος και η διασπορά των βαρών για το μοντέλο ResNet50[3] που είναι προεκπαιδευμένο στο ImageNet [35]. Η υλοποίηση περιγράφεται στο σχήμα B.3.

Στη συνέχεια δημιουργήθηκαν τρεις λίστες από μετασχηματισμούς, η `transform_train` που περιέχει τους μετασχηματισμούς για τις εικόνες εκπαίδευσης, η `transform_eval` που περιέχει τους μετασχηματισμούς για τις εικόνες ελέγχου και η `transform_test` που περιέχει τους μετασχηματισμούς για τις εικόνες δοκιμής. Σε όλες τις περιπτώσεις οι διαστάσεις των εικόνων μεγαλώνουν από $32 \times 32 \times 3$ σε $224 \times 224 \times 3$ προκειμένου το μέγεθος των εικόνων να είναι ίδιο με αυτό που δέχεται ως είσοδο το μοντέλο. Επίσης, παντού οι πίνακες των εικόνων μετατρέπονται σε tensors και οι τιμές τους κανονικοποιούνται με βάση τον μέσο όρο και την διασπορά των βαρών του ImageNet[35] που ορίστηκαν πριν.

Το `transform_train` σε αντίθεση με τις υπόλοιπες λίστες περιέχει μετασχηματισμό τυχαίας οριζόντιας περιστροφής της εικόνας, περιστρέφει γενικά τις εικόνες δέκα φορές και θέτει την φωτεινότητα, την αντίθεση και τον κορεσμό στο 0.2 των αρχικών τιμών. Ο λόγος που γίνονται αυτοί οι επιπλέον μετασχηματισμοί είναι για να εισαχθεί τυχαιότητα στα δεδομένα εισόδου και να δυσκολευτεί το μοντέλο περισσότερο στην εκμάθηση προτύπων. Έτσι, μειώνονται οι πιθανότητες για overfitting και ταυτόχρονα το μοντέλο έχει την δυνατότητα να μάθει πιο περίπλοκα μοτίβα στις εικόνες, ώστε να αυξήσει την ακρίβειά του. Ταυτόχρονα δεν υπάρχουν ιδιαίτερα μειονεκτήματα καθώς με επαρκή αριθμό εποχών το μοντέλο μαθαίνει και ξεπερνάει τις αρχικές δυσκολίες παρόλο που το σύνολο εκπαίδευσης έχει περισσότερους μετασχηματισμούς από το σύνολο ελέγχου. Οι λίστες μετασχηματισμών παρουσιάζονται στο σχήμα B.4.

Τέλος, αφού ορίστηκαν και οι μετασχηματισμοί, ορίζονται και αρχικοποιούνται τα σύνολα δεδομένων και τα dataloaders. Όλα τα dataloaders έχουν μέγεθος παρτίδας 64, δηλαδή σε κάθε παρτίδα υπάρχουν 64 εικόνες και αριθμό εργατών 8, δηλαδή 8 επεξεργαστές φορτώνουν τα δεδομένα. Επίσης, η σειρά με την οποία εμφανίζονται οι εικόνες του dataloader της εκπαίδευσης είναι τυχαία. Τα σύνολα δεδομένων και τα dataloaders παρουσιάζονται στο σχήμα B.5.

Κεφάλαιο 5

Μοντέλο ResNet50Classifier

Στο κεφάλαιο αυτό, εξετάζεται αρχικά η δομή και ο τρόπος λειτουργίας του μοντέλου ResNet50Classifier. Έπειτα, γίνεται περιγραφή της διαδικασίας εκπαίδευσης, ελέγχου και δοκιμής του μοντέλου.

5.1 Αντικείμενο ResNet50Classifier

Προκειμένου να υλοποιηθεί το μοντέλο ορίστηκε ένα αντικείμενο ResNet50Classifier στην Python και περιέχει την συνάρτηση `__init__`, η οποία υλοποιεί την αρχιτεκτονική του μοντέλου, και την συνάρτηση `__forward__` που περιγράφει την ροή δεδομένων στο μοντέλο, από την είσοδο έως την έξοδο. Ο κώδικας για τον ορισμό του ResNet50Classifier και τις συναρτήσεις `__init__` και `__forward__` παρουσιάζεται στο σχήμα Β.6.

5.1.1 Συνάρτηση `__init__`

Αρχικά, η μέθοδος λαμβάνει δύο ορίσματα, το `num_classes1` που είναι ο αριθμός των κλάσεων για την πρώτη κεφαλή και το `num_classes2` που είναι ο αριθμός των κλάσεων για την δεύτερη κεφαλή. Σε πρώτη φάση γίνεται αρχικοποίηση ενός μοντέλου ResNet50 [3] προεκπαιδευμένου στο ImageNet [35] και από αυτό αφαιρείται το τελευταίο πλήρως διασυνδεδεμένο στρώμα. Η αφαίρεση γίνεται προκειμένου να προσαρμοστεί το μοντέλο στις απαιτήσεις του πειράματος και να είναι ικανό να προβλέψει σωστά και τις δύο ετικέτες της βάσης δεδομένων CIFAKE [5], [6]. Έπειτα, στην θέση του τελικού στρώματος τοποθετείται ένα ακολουθιακό στρώμα, `final_layer`, που είναι κοινό και για τις δύο κεφαλές και κάνει τα εξής:

- μειώνει τη διάσταση των χαρακτηριστικών σε 512,
- κανονικοποιεί τα νέα διανύσματα,
- εφαρμόζει συνάρτηση ενεργοποίησης ReLU,
- εφαρμόζει τυχαία απενεργοποίηση του 30% των νευρώνων προκειμένου να αποφευχθεί η υπερβολική προσαρμογή στα δεδομένα εκπαίδευσης.

Ακολουθούν οι δύο κεφαλές του μοντέλου. Η κεφαλή 1 δηλαδή η head1 είναι υπεύθυνη για την δυαδική ταξινόμηση των εικόνων ανάμεσα σε αληθινές και σε ψεύτικες ή αλλιώς κατασκευασμένες από μέσα παραγωγικής τεχνητής νοημοσύνης. Η κεφαλή 2, δηλαδή η head2, είναι υπεύθυνη για την ταξινόμηση των εικόνων στις δέκα κλάσεις της βάσης δεδομένων CIFAR-10 [6] ανάλογα με το περιεχόμενο της εικόνας. Και οι δύο κεφαλές έχουν ίδια αρχιτεκτονική και το μόνο στο οποίο διαφέρουν είναι το μέγεθος του τελικού διανύσματος που υπολογίζεται. Οι κεφαλές αυτές υλοποιούν την εξής λειτουργία:

- Αρχικά, μειώνουν τη διάσταση των χαρακτηριστικών από 512 σε 256,
- εφαρμόζουν συνάρτηση ενεργοποίησης ReLU,
- εφαρμόζουν τυχαία απενεργοποίηση του 20% των νευρώνων,
- μειώνουν ξανά τη διάσταση σε 2 ή 10 ανάλογα με την κεφαλή.

Η λειτουργία του μοντέλου ResNet50 [3] περιγράφεται στο κεφάλαιο 3 και είναι ίδια και σε αυτό το μοντέλο με μοναδική διαφορά πως δεν χρησιμοποιείται το τελικό πλήρως διασυνδεδεμένο στρώμα. Συνεχίζοντας, επομένως, από την μέση υποδειγματοληψία του ResNet50 [3] έχουμε διάνυσμα χαρακτηριστικών: Feature vector: $\underline{x}_{\text{feat}} \in \mathbb{R}^d$, το οποίο επεξεργάζεται το πλήρως συνδεδεμένο επίπεδο final_layer με τον εξής τρόπο:

$$\mathbf{h} = \text{Dropout}(\text{ReLU}(\text{BN}(\mathbf{W}_1 \underline{x}_{\text{feat}} + \mathbf{b}_1))) \quad (5.1)$$

όπου:

- $\mathbf{W}_1 \in \mathbb{R}^{512 \times d}$,
- $\mathbf{b}_1 \in \mathbb{R}^{512}$,
- BN: Batch Normalization,

- ReLU: συνάρτηση ενεργοποίησης ReLU,
- Dropout: Dropout με πιθανότητα $p = 0.3$.

Κατόπιν, το $\mathbf{h}$ περνάει σαν είσοδος στις δύο κεφαλές και έχουμε για την κεφαλή 1:

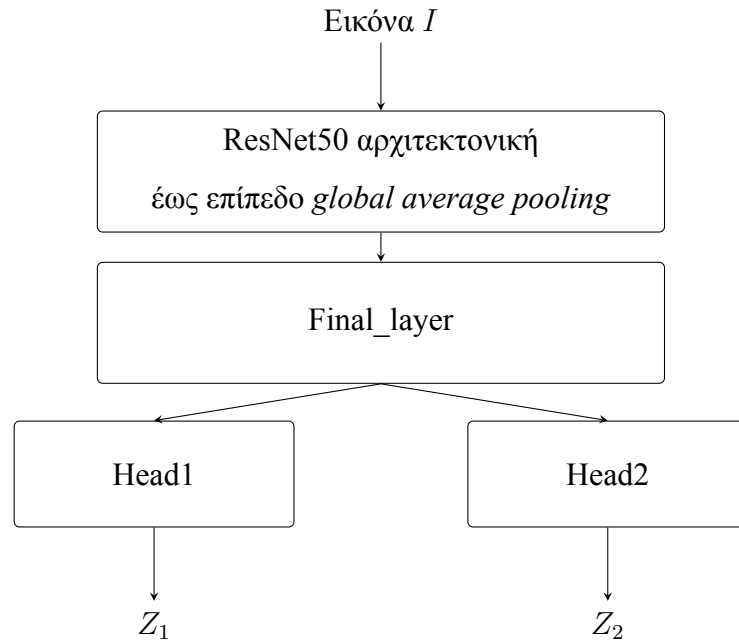
$$\mathbf{h}_1^{(1)} = \text{Dropout}(\text{ReLU}(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2)) \quad (5.2)$$

$$\mathbf{z}_1 = \mathbf{W}_3 \mathbf{h}_1^{(1)} + \mathbf{b}_3 \quad (5.3)$$

και για την κεφαλή 2:

$$\mathbf{h}_2^{(1)} = \text{Dropout}(\text{ReLU}(\mathbf{W}_4 \mathbf{h} + \mathbf{b}_4)) \quad (5.4)$$

$$\mathbf{z}_2 = \mathbf{W}_5 \mathbf{h}_2^{(1)} + \mathbf{b}_5 \quad (5.5)$$



Σχήμα 5.1: Η δομή του ResNet50Classifier

5.1.2 Συνάρτηση `__forward__`

Η συνάρτηση `__forward__` λαμβάνει ως όρισμα μια εικόνα $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$. Όπως περιγράφηκε προηγουμένως, η εικόνα αυτή εισέρχεται στο μοντέλο ResNet50Classifier και διέρχεται από τα στρώματα του ResNet50 [3] μέχρι το στάδιο του global average pooling. Ύστερα, το διάνυσμα χαρακτηριστικών που έχει προκύψει επιπεδοποιείται και διέρχεται από το final_layer και τις 2 κεφαλές. Στο τέλος παράγεται μία έξοδος της μορφής $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)$, όπου $\mathbf{z}_1 \in \mathbb{R}^2$ και $\mathbf{z}_2 \in \mathbb{R}^{10}$.

Algorithm 2 Η διαδικασία forward pass στο μοντέλο ResNet50Classifier**Require:** Εικόνα Εισόδου $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$

- 1: $\mathbf{x}_{\text{feat}} \leftarrow \text{Global Average Pooling}(\text{ResNet50}(\mathbf{I}))$
- 2: $\mathbf{h} \leftarrow \text{Dropout}(\text{ReLU}(\text{BN}(\mathbf{W}_1 \mathbf{x}_{\text{feat}} + \mathbf{b}_1)))$
- 3: $\mathbf{h}_1^{(1)} \leftarrow \text{Dropout}(\text{ReLU}(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2))$
- 4: $\mathbf{z}_1 \leftarrow \mathbf{W}_3 \mathbf{h}_1^{(1)} + \mathbf{b}_3$
- 5: $\mathbf{h}_2^{(1)} \leftarrow \text{Dropout}(\text{ReLU}(\mathbf{W}_4 \mathbf{h} + \mathbf{b}_4))$
- 6: $\mathbf{z}_2 \leftarrow \mathbf{W}_5 \mathbf{h}_2^{(1)} + \mathbf{b}_5$

Ensure: $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)$

5.2 Εκπαίδευση μοντέλου

Προκειμένου το μοντέλο να ταξινομεί σωστά εικόνες πρέπει πρώτα να εκπαιδευτεί. Στην συγκεκριμένη περίπτωση, επειδή χρησιμοποιείται ήδη ένα προεκπαιδευμένο μοντέλο ως βάση είναι δυνατή η μεταφορά γνώσης από το ήδη προεκπαιδευμένο στο δίκτυο του πειράματος με κατάλληλη λεπτομερή προσαρμογή των υπερπαραμέτρων εκμάθησης.

5.2.1 Αλγόριθμος εκπαίδευσης

Το μοντέλο τυπικά εκπαιδύεται για εκατό εποχές, όμως γίνεται πρόωρος τερματισμός της εκπαίδευσης, εάν δεν υπάρξει βελτίωση της ακρίβειας ελέγχου για πέντε διαδοχικές εποχές. Η συνάρτηση απώλειας που χρησιμοποιείται και για τις δύο κεφαλές είναι η συνάρτηση διασταυρούμενης εντροπίας (Cross Entropy Loss) και ο αλγόριθμος βελτιστοποίησης που χρησιμοποιείται είναι ο αλγόριθμος Adam [46]. Στην αρχή όλα τα στρώματα του μοντέλου ResNet50 [3] είναι “παγωμένα”, δηλαδή δεν γίνονται αλλαγές στα βάρη των στρωμάτων αυτών. Τα μόνα στρώματα που μπορούν να αλλάξουν βάρη ανάλογα με τις ανάγκες του μοντέλου είναι τα `final_layer`, `head1` και `head2`.

Λόγω της διαφορετικής δυσκολίας των δύο εργασιών ταξινόμησης χρησιμοποιείται διαφορετικός ρυθμός μάθησης (learning rate - LR) στις κεφαλές, με την κεφαλή `head1`, που είναι υπεύθυνη για τον πιο εύκολο στόχο της δυαδικής ταξινόμησης, να έχει $LR = 3 \times 10^{-4}$ και την κεφαλή `head2` με την ταξινόμηση σε 10 κλάσεις να έχει $LR = 5 \times 10^{-4}$. Το επίπεδο `final_layer` που είναι κοινό και για τις δύο κεφαλές έχει $LR = 3 \times 10^{-4}$. Επίσης, η απώλεια βαρών `weight_decay` είναι ίση με 5×10^{-6} . Προκειμένου να αποφευχθεί η υπερπροσαρμογή του μοντέλου στα δεδομένα εκπαίδευσης, χρησιμοποιείται χρονοδρομολογητής

ρυθμού εκμάθησης (scheduler) που ακολουθεί ημιτονοειδή απόσβεση (cosine annealing) με μέγιστες εποχές περιόδου τις εκατό. Τέλος, στην εποχή 11 “ξεπαγώνει” το τελευταίο συνελικτικό block του ResNet50 [3] και στην εποχή 31 το τρίτο συνελικτικό block του ResNet50, δηλαδή αρχίζουν και μαθαίνουν και εκείνα τα στρώματα από τα δεδομένα εκπαίδευσης. Σε κάθε φάση “ξεπαγώματος” προσαρμόζεται κατάλληλα ο ρυθμός μάθησης με τα στρώματα να έχουν $LR = 10^{-5}$. Ταυτόχρονα προσαρμόζεται κατάλληλα και ο scheduler. Ο κώδικας που ρυθμίζει τις υπερπαραμέτρους του πειράματος παρουσιάζεται στο σχήμα B.7

Ξεκινώντας από την συνάρτηση κόστους, την Cross-Entropy Loss και με τις εξής παραδοχές,

- έξοδος μοντέλου ταξινόμησης: $\hat{y}_i$,
- δείγμα εκπαίδευσης: x_i ,
- ετικέτα για δείγμα εκπαίδευσης x_i : y_i

έχουμε:

- Για την πρόβλεψη του μοντέλου ισχύει:

$$\hat{y}_i = f(x_i, \theta) \quad (5.6)$$

όπου θ είναι όλες οι παράμετροι του δικτύου.

- Η συνάρτηση κόστους για το δείγμα i είναι:

$$\mathcal{L}_i = \mathcal{L}_{\text{output}}(\hat{y}_i, y_i) \quad (5.7)$$

- Άρα η συνάρτηση κόστους Cross Entropy γίνεται:

$$\mathcal{L}_{\text{output}}(\hat{y}_i, y_i) = - \sum_{n=1}^N \mathbf{1}_{\{y_i=n\}} \cdot \log \left(\hat{p}_i^{(n)} \right) \quad (5.8)$$

Το $\hat{p}_i^{(n)}$ είναι η εκτιμώμενη πιθανότητα ότι το δείγμα i ανήκει στην κλάση n και προκύπτει με την χρήση softmax, όπως ορίστηκε στην εξίσωση softmax για CNN. Είναι δυνατόν να επαναδιατυπωθεί η εξίσωση αυτή απλά αντικαθιστώντας $K = N$ κλάσεις. Ισχύει λοιπόν:

$$\hat{p}_i^{(n)} = \frac{e^{z_i^{(n)}}}{\sum_{k=1}^N e^{z_i^{(k)}}} \quad (5.9)$$

Αντικαθιστώντας την εξίσωση 5.9 στην εξίσωση 5.8 προκύπτει:

$$\mathcal{L}_i = - \sum_{n=1}^N \mathbf{1}_{\{y_i=n\}} \cdot \log \left(\frac{e^{z_i^{(n)}}}{\sum_{k=1}^N e^{z_i^{(k)}}} \right) \quad (5.10)$$

και συνεχίζοντας τις πράξεις τελικά ισχύει:

$$\mathcal{L}_i = -z_i^{(y_i)} + \log \left(\sum_{k=1}^N e^{z_i^{(k)}} \right) \quad (5.11)$$

Αφού υπολογίστηκε η εξίσωση για γενική μορφή της Cross Entropy Loss, για τα δεδομένα του πειράματος ισχύει:

Για το μοντέλο υπάρχουν για κάθε δείγμα i δύο έξοδοι logits:

- $\mathbf{z}_i^{(1)}$: Logits για την κεφαλή head1,
- $\mathbf{z}_i^{(2)}$: Logits για την κεφαλή head2

Για κάθε κεφαλή εφαρμόζεται Softmax όπως στην εξίσωση 5.9 και στη συνέχεια υπολογίζεται Cross Entropy Loss. Η συνολική απώλεια για δείγμα i είναι το άθροισμα των επιμέρους απωλειών:

$$\mathcal{L}_i = \mathcal{L}_i^{(1)} + \mathcal{L}_i^{(2)} \quad (5.12)$$

όπου,

$$\mathcal{L}_i^{(1)} = -z_i^{(1,y_i^{(1)})} + \log \left(\sum_{k=1}^{N_1} e^{z_i^{(1,k)}} \right) \quad (5.13)$$

και

$$\mathcal{L}_i^{(2)} = -z_i^{(2,y_i^{(2)})} + \log \left(\sum_{k=1}^{N_2} e^{z_i^{(2,k)}} \right) \quad (5.14)$$

Στις εξισώσεις αυτές ισχύει:

- N_1, N_2 : αριθμός κλάσεων ανά κεφαλή,
- $y_i^{(1)}, y_i^{(2)}$: Οι σωστές κλάσεις για κάθε κεφαλή ανά δείγμα i ,
- $z_i^{(1,k)}, z_i^{(2,k)}$: Τα logits για την κλάση k στην κεφαλή 1 και κεφαλή 2 αντίστοιχα.

Ανά παρτίδα μεγέθους M η απώλεια δίνεται ως μέσος όρος:

$$\mathcal{L}_{\text{batch}} = \frac{1}{M} \sum_{i=1}^M \mathcal{L}_i \quad (5.15)$$

Αρα με $M = 64$ και $N_1 = 2$ και $N_2 = 10$ ο τελικός υπολογισμός απώλειας ανά παρτίδα είναι:

$$\mathcal{L}_{\text{batch}} = \frac{1}{64} \sum_{i=1}^{64} \left(-z_i^{(1, y_i^{(1)})} + \log \left(\sum_{k=1}^2 e^{z_i^{(1, k)}} \right) - z_i^{(2, y_i^{(2)})} + \log \left(\sum_{k=1}^{10} e^{z_i^{(2, k)}} \right) \right) \quad (5.16)$$

και η συνολική απώλεια ανά εποχή είναι ο μέσος όρος των απωλειών ανά παρτίδα:

$$\mathcal{L}_{\text{epoch}} = \frac{1}{B} \sum_{b=1}^B \mathcal{L}_{\text{batch}}^{(b)} \quad (5.17)$$

όπου B είναι ο συνολικός αριθμός παρτίδων.

Στόχος κατά την εκπαίδευση σε κάθε εποχή είναι η ελαχιστοποίηση της εξίσωσης 5.17, δηλαδή:

$$\min_{\theta} \mathcal{L}_{\text{epoch}} \quad (5.18)$$

Η στρατηγική βελτιστοποίησης είναι η Adam (Adaptive Moment Estimation) και για κάθε παρτίδα b , ο υπολογισμός έχει ως εξής:

$$m_b = \beta_1 \cdot m_{b-1} + (1 - \beta_1) \cdot \nabla_{\theta} \mathcal{L}_{\text{batch}}^{(b)} \quad (5.19)$$

$$v_b = \beta_2 \cdot v_{b-1} + (1 - \beta_2) \cdot \left(\nabla_{\theta} \mathcal{L}_{\text{batch}}^{(b)} \right)^2 \quad (5.20)$$

όπου:

- m_b : εκθετικός κινητός μέσος όρος (Exponential Moving Average - EMA) πρώτης τάξης για το b ,
- v_b : EMA δεύτερης τάξης για το b ,
- $\nabla_{\theta} \mathcal{L}_{\text{batch}}^{(b)}$: η κλίση της απώλειας $\mathcal{L}$ της παρτίδας b ως προς τις παραμέτρους θ του μοντέλου,
- β_1 : συντελεστής εξομάλυνσης για το EMA πρώτης τάξης,
- β_2 : συντελεστής εξομάλυνσης για το EMA δεύτερης τάξης.

Αφού υπολογιστούν τα EMA, ο αλγόριθμος Adam προσπαθεί να διορθώσει την μεροληψία υπολογίζοντας τις ακόλουθες μη μεροληπτικές εκτιμήσεις:

$$\hat{m}_b = \frac{m_b}{1 - \beta_1^b} \quad (5.21)$$

$$\hat{v}_b = \frac{v_b}{1 - \beta_2^b} \quad (5.22)$$

Τέλος, ενημερώνονται οι τιμές των παραμέτρων θ :

$$\theta_{b+1} = \theta_b - \alpha_b \cdot \frac{\hat{m}_b}{\sqrt{\hat{v}_b} + \epsilon} \quad (5.23)$$

όπου:

- α_b : Ο ρυθμός εκμάθησης στο batch b ,
- ϵ : Σταθερά για αριθμητική σταθερότητα

Αυτή η διαδικασία επαναλαμβάνεται για όλα τα b σε μια εποχή [46].

Ο ρυθμός εκμάθησης μεταβάλλεται και αυτός δυναμικά κατά τη διάρκεια της εκπαίδευσης λόγω του χρονοδρομολογητή. Συγκεκριμένα:

$$\forall b \in \{1, 2, \dots, B \cdot T_{\max}\}, \quad t = \lfloor \frac{b}{B} \rfloor$$

ισχύει:

$$\alpha_t = \alpha_{\min} + \frac{1}{2} (\alpha_{\max} - \alpha_{\min}) \left(1 + \cos \left(\frac{\pi \cdot t}{T_{\max}} \right) \right) \quad (5.24)$$

με:

- $\alpha_{\max}$: η μέγιστη αρχική τιμή LR ,
- $\alpha_{\min}$: η ελάχιστη επιτρεπτή τιμή LR ,
- t : η εποχή,
- $T_{\max}$: ο συνολικός αριθμός εποχών του scheduler.

Σημειωτέον, ότι καθώς αυξάνεται το t το α_t μειώνεται.

Επίσης, ισχύει $\theta = \theta_{\text{frozen}}^{(t)} \cup \theta_{\text{trainable}}^{(t)}$, όπου:

$$\theta_{\text{trainable}}^{(t)} = \begin{cases} \theta_{\text{heads, final_layer}} & \text{αν } t < e_1 \\ \theta_{\text{heads, final_layer}} \cup \theta_{\text{layer4}} & \text{αν } e_1 \leq t < e_2 \\ \theta_{\text{heads, final_layer}} \cup \theta_{\text{layer4}} \cup \theta_{\text{layer3}} & \text{αν } t \geq e_2 \end{cases}$$

όπου:

- e_1, e_2 : οι εποχές στις οποίες γίνεται ξεπάγωμα,
- $\theta_{\text{heads, final_layer}}$: παράμετροι ταξινομητών κεφαλών και final_layer,
- $\theta_{\text{layer4}}, \theta_{\text{layer3}}$: παράμετροι των τελευταίων στρωμάτων του ResNet50 [3].

Σε κάθε εποχή που γίνεται “ξεπάγωμα” έχουμε τροποποίηση των παραμέτρων που εκπαιδεύονται και κατά συνέπεια τροποποίηση στο σετ παραμέτρων που αξιοποιούνται στον υπολογισμό των κλίσεων.

Στο τέλος κάθε εποχής υπολογίζεται η ακρίβεια ελέγχου από την ρουτίνα ελέγχου. Αν η ακρίβεια ελέγχου είναι μεγαλύτερη από την καλύτερη καταγεγραμμένη ακρίβεια ελέγχου, τότε αποθηκεύεται το μοντέλο αυτής της εποχής ως το καλύτερο μοντέλο και στην μεταβλητή της καλύτερης καταγεγραμμένης ακριβείας ελέγχου αποθηκεύεται η τιμή της ακριβείας ελέγχου αυτής της εποχής. Έτσι η επιλογή των παραμέτρων θ^* του καλύτερου μοντέλου εκφράζεται από την:

$$\text{Acc}_{\text{val}}^{(t)} > \max_{k < t} \left(\text{Acc}_{\text{val}}^{(k)} \right) \quad (5.25)$$

όπου το $\text{Acc}_{\text{val}}^{(t)}$ είναι η ακρίβεια ελέγχου για την εποχή t . Άρα ο συνολικός μαθηματικός στόχος συνοψίζεται ως:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{\text{epochs}} [\mathcal{L}_{\text{epoch}}] \quad (5.26)$$

με τον περιορισμό για το θ^* της εξίσωσης 5.25. Ο αλγόριθμος 3 περιγράφει την διαδικασία εκπαίδευσης του μοντέλου.

5.2.2 Μετρικές κατά την εκπαίδευση και σημασία “ξεπαγώματος”

Καθόλη την εκπαίδευση του μοντέλου αποθηκεύτηκαν σημαντικές μετρικές ανά εποχή, όπως συνολική απώλεια εκπαίδευσης ανά εποχή και ακρίβεια (accuracy) κεφαλής ανά εποχή και κοινό accuracy ανά εποχή. Η ακρίβεια κεφαλής δείχνει την δυνατότητα του μοντέλου να ταξινομεί σωστά τις εικόνες ανάλογα με τις ετικέτες της εκάστοτε εργασίας ταξινόμησης. Μαθηματικά οι απώλειες εκφράστηκαν στην προηγούμενη ενότητα από την εξίσωση 5.17. Για τον υπολογισμό των μετρικών accuracy λαμβάνονται οι εξής παραδοχές:

- N_{total} : το συνολικό πλήθος δειγμάτων στο σύνολο εκπαίδευσης,
- $N_{\text{correct}}^{(1)}$: ο αριθμός σωστών προβλέψεων πρώτης κεφαλής,
- $N_{\text{correct}}^{(2)}$: ο αριθμός σωστών προβλέψεων δεύτερης κεφαλής

Η ακρίβεια της πρώτης κεφαλής ισούται με:

$$\text{Acc}_{\text{train}}^{(1)} = \frac{N_{\text{correct}}^{(1)}}{N_{\text{total}}} \times 100 \quad (5.27)$$

Algorithm 3 Αλγόριθμος εκπαίδευσης

Require: σύνολο εκπαίδευσης $\mathcal{D}_{\text{train}}$, σύνολο ελέγχου $\mathcal{D}_{\text{val}}$, αρχικές παράμετροι θ , ρυθμοί μάθησης, αριθμός εποχών T_{max} , εποχές ξεπαγώματος e_1, e_2

Ensure: Βέλτιστες παράμετροι θ^*

```

1: Αρχικοποίησε Adam και cosine annealing scheduler
2: for εποχή  $t = 1$  έως  $T_{\text{max}}$  do
3:   if  $t = e_1$  then
4:     Ξεπάγωσε layer4
5:     Ρύθμισε ρυθμούς μάθησης
6:     Επανεκκίνησε scheduler
7:   end if
8:   if  $t = e_2$  then
9:     Ξεπάγωσε layer3
10:    Ρύθμισε ρυθμούς μάθησης
11:    Επανεκκίνησε scheduler
12:  end if
13:  Θέσε  $\mathcal{L}_{\text{epoch}} \leftarrow 0$ 
14:  for παρτίδα  $b \in \mathcal{D}_{\text{train}}$  do
15:    Φόρτωσε δεδομένα παρτίδας
16:    Υπολόγισε έξοδο από μοντέλο
17:    Υπολόγισε την απώλεια  $\mathcal{L}_{\text{batch}}$ 
18:    Υπολόγισε κλίσεις
19:    Ενημέρωσε παραμέτρους μέσω Adam
20:    Ενημέρωσε ρυθμό μάθησης μέσω scheduler
21:    Πρόσθεσε  $\mathcal{L}_{\text{batch}}$  στο  $\mathcal{L}_{\text{epoch}}$ 
22:  end for
23:  Υπολόγισε μέση απώλεια εποχής
24:  Εκτέλεσε ρουτίνα ελέγχου και υπολόγισε  $\text{Acc}_{\text{val}}^{(t)}$ 
25:  if  $\text{Acc}_{\text{val}}$  ικανοποιεί την εξίσωση 5.25 then
26:    Αποθήκευσε το τρέχον μοντέλο ως καλύτερο
27:  end if
28: end for
29: return  $\theta^*$ 

```

$$\text{Acc}_{\text{train}}^{(2)} = \frac{N_{\text{correct}}^{(2)}}{N_{\text{total}}} \times 100 \quad (5.28)$$

όπου κάθε $N_{\text{correct}}^{(h)}$ ισούται με:

$$N_{\text{correct}}^{(h)} = \sum_{i=1}^{N_{\text{total}}} \mathbf{1}_{\{\hat{y}_i^{(h)} = y_i^{(h)}\}}, \quad h = 1, 2 \quad (5.29)$$

και:

$$\hat{y}_i^{(h)} = \arg \max_k \left(\hat{p}_i^{(h,k)} \right) \quad (5.30)$$

όπου $\hat{p}_i^{(h,k)}$ είναι η εκτιμώμενη πιθανότητα του μοντέλου για την κλάση k της κεφαλής h στο δείγμα i .

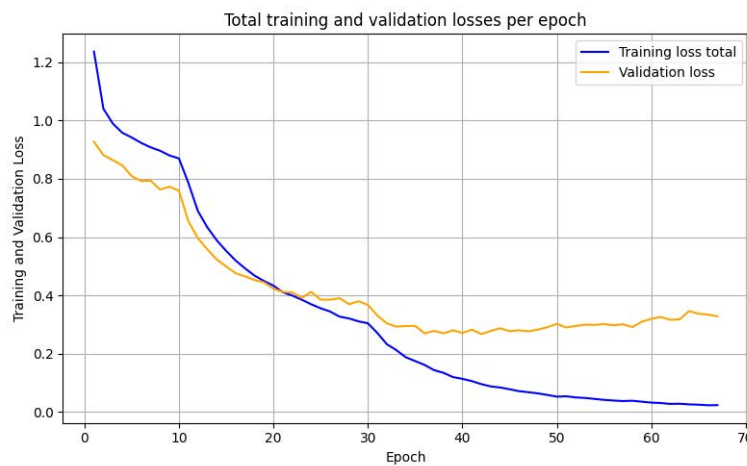
Επίσης, παρόλο που δεν υπολογίζεται στην εκπαίδευση, η κοινή ακρίβεια είναι μια σημαντική μετρική που προσφέρει συνολική εικόνα για την ευστοχία του μοντέλου. Δείχνει την ικανότητα του μοντέλου να ταξινομεί σωστά και τις δύο ετικέτες μιας εικόνας. Για να θεωρηθεί, δηλαδή, πως μια εικόνα ταξινομείται σωστά, πρέπει και οι δύο κεφαλές να τοποθετούν τις σωστές ετικέτες ταυτόχρονα, αλλιώς θεωρείται λάθος ταξινόμηση.

Η κοινή ακρίβεια ορίζεται ως:

$$\text{Acc}^{(\text{joint})} = \frac{N_{\text{correct}}^{(\text{joint})}}{N_{\text{total}}} \times 100 \quad (5.31)$$

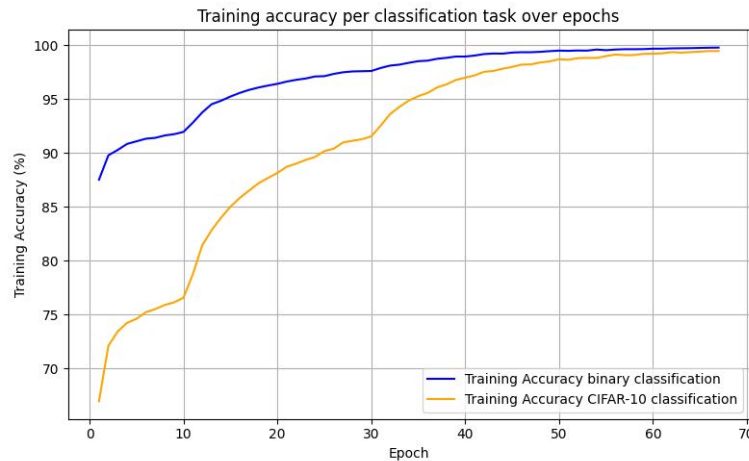
όπου:

$$N_{\text{correct}}^{(\text{joint})} = \sum_{i=1}^{N_{\text{total}}} \mathbf{1}_{\{\hat{y}_i^{(1)} = y_i^{(1)}\}} \cdot \mathbf{1}_{\{\hat{y}_i^{(2)} = y_i^{(2)}\}} \quad (5.32)$$



Σχήμα 5.2: Το γράφημα των απωλειών εκπαίδευσης και ελέγχου

Αρχικά, όπως φαίνεται και από τα σχήματα 5.2 και 5.3, το μοντέλο εκπαιδεύεται για 67 αντί για 100 εποχές, άρα η κοινή ακρίβεια ελέγχου έφτασε την μέγιστη τιμή της την εποχή 62.



Σχήμα 5.3: Τα accuracy κάθε κεφαλής ανά εποχή κατά την εκπαίδευση

Επίσης, σύμφωνα με το γράφημα απωλειών και οι απώλειες εκπαίδευσης και ελέγχου μειώνονται στις 35 πρώτες εποχές. Έπειτα, οι απώλειες εκπαίδευσης συνεχίζουν να μειώνονται σε κάθε εποχή σχεδόν ως το τέλος, σε αντίθεση με τις απώλειες ελέγχου που σταθεροποιούνται γύρω στην τιμή 0.3 και προς το τέλος της εκπαίδευσης παρουσιάζουν μια μικρή αύξηση. Μάλιστα, οι απώλειες εκπαίδευσης στις πρώτες εποχές έχουν υψηλότερη τιμή από τις απώλειες ελέγχου μέχρι περίπου την εποχή 20. Οι μεγαλύτερες αυτές τιμές αποδεικνύουν την μεγαλύτερη αρχική δυσκολία του μοντέλου να ταξινομήσει τα δεδομένα εκπαίδευσης, η οποία οφείλεται κυρίως στους επιπλέον μετασχηματισμούς των εικόνων εκπαίδευσης. Τέλος, είναι εμφανείς στο διάγραμμα οι εποχές που “ξεπαγώνουν” στρώματα, καθώς μετά την 10η και την 30η εποχή υπάρχει απότομη πτώση και των δύο απωλειών.

Από το διάγραμμα accuracy του σχήματος 5.3 προκύπτουν 3 συμπεράσματα. Πρώτον, η εργασία της δεύτερης κεφαλής για την ταξινόμηση εικόνων σε δέκα κλάσεις είναι πολύ δυσκολότερη της δυαδικής ταξινόμησης, ειδικά στις πρώτες εποχές. Αυτό είναι αναμενόμενο, καθώς με περισσότερες κλάσεις αυξάνονται οι πιθανότητες το μοντέλο να συγχέει ορισμένες κλάσεις, ειδικά στην αρχή που δεν έχει ρυθμίσει ακόμα σωστά τα βάρη του. Δεύτερον, οι 67 εποχές εκπαίδευσης φαίνεται να είναι αρκετές, καθώς το μοντέλο και στα δύο σύνολα πλησιάζει το 100% accuracy εγείροντας μάλιστα ανησυχίες για πιθανό overfitting. Τρίτον, φαίνεται η σημασία του “ξεπαγώματος” στρωμάτων στην βελτίωση της αποτελεσματικότητας του μοντέλου, ειδικά για ταξινόμηση σε πολλές κλάσεις. Μέχρι την 11η εποχή που αρχίζει και το layer4 του ResNet50 [3] να αλλάζει βάρη η ακρίβεια του head2 αυξάνεται έως 76.5% ενώ του head1 έως 92%. Την 11η εποχή η ακρίβεια αυξάνεται απότομα και στις δύο κεφαλές φτάνοντας το 93% για το head1 και το 79% για το head2. Συνεχίζει μάλιστα να

είναι ανοδική η πορεία και για τις δύο μετρικές μέχρι την εποχή 31 που αρχίζει και εκπαιδεύεται το layer3. Μετά την 31η εποχή υπάρχει ξανά άλμα στην ακρίβεια αν και μικρότερο σε μέγεθος. Το σταδιακό “ξεπάγωμα” επιτρέπει στο μοντέλο να προσαρμόσει λεπτομερώς τα βάρη των στρωμάτων που εκπαιδεύονται χωρίς να αλλοιώσει πιθανόν χρήσιμα πρότυπα που έχει μάθει από το σύνολο στο οποίο είναι προεκπαιδευμένο. Με αυτόν τον τρόπο αυξάνεται η αποτελεσματικότητά του και μειώνεται σημαντικά ο χρόνος εκπαίδευσης γιατί πρέπει να ανανεωθούν λιγότερα βάρη.

5.3 Έλεγχος μοντέλου

Πριν ολοκληρωθεί κάθε εποχή το μοντέλο διερχόταν από μία φάση ελέγχου και κατά την διάρκειά της συλλέχθηκαν σημαντικά στατιστικά για την απόδοση του μοντέλου σε εικόνες στις οποίες δεν είχε εκπαιδευτεί. Τα στατιστικά αυτά μπορούν να αναδείξουν αδυναμίες της αρχιτεκτονικής και της μεθόδου εκπαίδευσης και επιτρέπουν στο σχεδιαστή του νευρωνικού μοντέλου να το βελτιστοποιήσει. Επίσης, εφόσον σε κάθε εποχή το μοντέλο έχει διαφορετική επίδοση, τα στατιστικά αυτά επιτρέπουν να απομονωθεί η πιο αποτελεσματική έκδοση με στόχο να αποθηκευτεί και να χρησιμοποιηθεί ως σημείο σύγκρισης για μελλοντικές προσπάθειες εκπαίδευσης.

5.3.1 Αλγόριθμος ελέγχου

Κατά τον έλεγχο του μοντέλου, αρχικά δεν εφαρμόζεται οπισθοδιάδοση, για να μην αλλάξουν τα βάρη των στρωμάτων, σε αντίθεση με την εκπαίδευση. Το μοντέλο με τους νευρώνες έτσι όπως έχουν διαμορφωθεί από την αντίστοιχη εποχή εκμάθησης καλείται να ταξινομήσει κατάλληλα τις εικόνες στις ετικέτες, προκειμένου να δοθεί μια πρόωμη εκτίμηση για τις δυνατότητές του. Για αυτό τον λόγο, για κάθε παρτίδα ελέγχει πόσες και ποιες ετικέτες τοποθετήθηκαν σωστά στις εικόνες και υπολογίζει τις απώλειες ελέγχου, την κοινή ακρίβεια και την ακρίβεια ανά κεφαλή. Ο αλγόριθμος 4 είναι ο αλγόριθμος ελέγχου.

5.3.2 Μετρικές απωλειών και ακρίβειας στον έλεγχο

Μαθηματικά οι μετρικές απωλειών ελέγχου, η ακρίβεια κεφαλών και η κοινή ακρίβεια υπολογίζονται παρόμοια με μετρικές εκπαίδευσης, με μικρές μόνο διαφορές. Καταρχήν και οι τρεις μετρικές χρησιμοποιούν δείγματα ελέγχου αντί για δείγματα εκπαίδευσης και στον

Algorithm 4 Αλγόριθμος ελέγχου**Require:** Σύνολο ελέγχου D_{val} , εκπαιδευμένο μοντέλο f_{θ} , συνάρτηση κόστους $\mathcal{L}_{\text{output}}$ **Ensure:** Ακρίβεια ανά κεφαλή, κοινή ακρίβεια, μέσες απώλειες ανά κεφαλή, συνολική απώλεια

```

1:  $total\_loss \leftarrow 0$ 
2:  $total\_loss^{(1)} \leftarrow 0, total\_loss^{(2)} \leftarrow 0$ 
3:  $correct^{(1)} \leftarrow 0, correct^{(2)} \leftarrow 0, correct^{(join)} \leftarrow 0$ 
4:  $N_{\text{examined}} \leftarrow 0$ 
5: Θέσε το μοντέλο σε evaluation mode
6: for παρτίδα  $b \in \mathcal{D}_{\text{val}}$  do
7:   Φόρτωσε  $images, labels^{(1)}, labels^{(2)}$ 
8:   Υπολόγισε τις εξόδους των δύο κεφαλών:  $output^{(1)}, output^{(2)} \leftarrow f_{\theta}(images)$ 
9:   Υπολόγισε  $\mathcal{L}^{(1)} \leftarrow \mathcal{L}_{\text{output}}(\hat{y}^{(1)}, y^{(1)}), \mathcal{L}^{(2)} \leftarrow \mathcal{L}_{\text{output}}(\hat{y}^{(2)}, y^{(2)})$ 
10:  Ενημέρωσε  $total\_loss^{(1)}, total\_loss^{(2)}, total\_loss$ 
11:  Υπολόγισε προβλέψεις  $\hat{y}^{(1)}, \hat{y}^{(2)}$  με argmax
12:  Ενημέρωσε  $correct^{(1)}, correct^{(2)}, correct^{(join)}$ 
13:  Ενημέρωσε  $N_{\text{examined}}$ 
14: end for
15: if  $N_{\text{examined}} = 0$  then
16:
17:   return (0,0,0,0,0)
18: end if
19: Υπολόγισε:

```

$$Acc_{\text{val}}^{(1)} = \frac{correct^{(1)}}{N_{\text{examined}}} \times 100$$

$$Acc_{\text{val}}^{(2)} = \frac{correct^{(2)}}{N_{\text{examined}}} \times 100$$

$$Acc_{\text{val}}^{(join)} = \frac{correct^{(join)}}{N_{\text{examined}}} \times 100$$

$$Avgloss_{\text{val}}^{(1)} = \frac{total_loss^{(1)}}{|\mathcal{D}_{\text{val}}|}$$

$$Avgloss_{\text{val}}^{(2)} = \frac{total_loss^{(2)}}{|\mathcal{D}_{\text{val}}|}$$

20:

21: **return** $Acc_{\text{val}}^{(1)}, Acc_{\text{val}}^{(2)}, Avgloss_{\text{val}}^{(1)}, Avgloss_{\text{val}}^{(2)}, Acc_{\text{val}}^{(join)}$

υπολογισμό των απωλειών δεν γίνεται ενημέρωση παραμέτρων, γιατί δεν υπάρχει οπισθοδιάδοση. Η απώλεια ανά εποχή είναι:

$$\mathcal{L}_{\text{val}} = \frac{1}{B_{\text{val}}} \sum_{b=1}^{B_{\text{val}}} \mathcal{L}_{\text{batch}}^{(b)} \quad (5.33)$$

Η ακρίβεια πρώτης κεφαλής ορίζεται ως:

$$\text{Acc}_{\text{val}}^{(1)} = \frac{N_{\text{correct}}^{(1)}}{N_{\text{val}}} \times 100 \quad (5.34)$$

Η ακρίβεια δεύτερης κεφαλής ορίζεται ως:

$$\text{Acc}_{\text{val}}^{(2)} = \frac{N_{\text{correct}}^{(2)}}{N_{\text{val}}} \times 100 \quad (5.35)$$

και η κοινή ακρίβεια είναι:

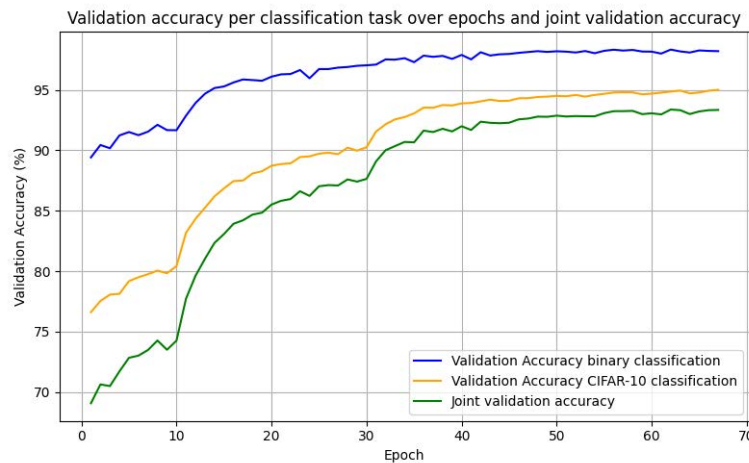
$$\text{Acc}_{\text{val}}^{(\text{joint})} = \frac{N_{\text{correct}}^{(\text{joint})}}{N_{\text{val}}} \times 100 \quad (5.36)$$

όπου:

$$N_{\text{correct}}^{(h)} = \sum_{i=1}^{N_{\text{val}}} \mathbf{1}_{\{\hat{y}_i^{(h)} = y_i^{(h)}\}}, \quad h = 1, 2$$

και

$$N_{\text{correct}}^{(\text{joint})} = \sum_{i=1}^{N_{\text{val}}} \mathbf{1}_{\{\hat{y}_i^{(1)} = y_i^{(1)}\}} \cdot \mathbf{1}_{\{\hat{y}_i^{(2)} = y_i^{(2)}\}}$$



Σχήμα 5.4: Η ακρίβεια ανά κεφαλή και η κοινή ακρίβεια ελέγχου

Από το σχήμα 5.4 παρατηρείται ότι η κοινή ακρίβεια και η ακρίβεια της δεύτερης κεφαλής αυξάνονται σχετικά γρήγορα μέχρι την εποχή 50 και η ακρίβεια πρώτης κεφαλής αυξάνεται σχετικά γρήγορα μέχρι την εποχή 40. Η ακρίβεια δυαδικής ταξινόμησης σταθεροποιείται περίπου στο 98-99%, η ακρίβεια ταξινόμησης στις κλάσεις του CIFAR-10 [6] πλησιάζει

μέγιστη τιμή λίγο μεγαλύτερη από 95% και η κοινή ακρίβεια φτάνει 93% περίπου. Οι χαμηλότερες τιμές της κοινής ακρίβειας είναι αναμενόμενες λόγω του ίδιου του ορισμού της μετρικής που επιτάσσει να είναι σωστές ταυτόχρονα και οι δύο ετικέτες. Επίσης, στις δύο τελευταίες μετρικές είναι φανερός ο αντίκτυπος του “ξεπαγώματος” στις εποχές 11 και 31 κατά τις οποίες παρατηρείται αλματώδης αύξηση στις τιμές. Επιπλέον, σε σχέση με το σχήμα 5.3 φανερώνεται ότι τελικά δεν υπάρχει overfitting, διότι αυξάνονται όλες οι μετρικές ακριβείας ελέγχου.

5.4 Δοκιμή μοντέλου

Αφού ολοκληρώθηκε η εκπαίδευση και ο έλεγχος του μοντέλου και αποθηκεύτηκε το μοντέλο με την καλύτερη κοινή ακρίβεια ελέγχου, ακολουθεί η δοκιμή του. Κατά την διάρκεια της δοκιμής αξιολογείται η ικανότητά του να ταξινομεί σωστά τις εικόνες ανάλογα με τις ετικέτες που τοποθετεί και εξάγονται μετρικές που περιγράφουν λεπτομερώς την λειτουργία του.

5.4.1 Αλγόριθμος δοκιμής

Αρχικά, κατά την ρουτίνα δοκιμής απενεργοποιείται η δυνατότητα οπισθοδιάδοσης κρατώντας σταθερές τις τιμές των παραμέτρων του μοντέλου. Το δίκτυο εξετάζει τις εικόνες του συνόλου δοκιμής ανά παρτίδα που λαμβάνει ως είσοδο και παράγει τα διανύσματα εξόδου. Ακολουθώντας, εφαρμόζει softmax σε αυτά τα διανύσματα και τοποθετεί την ετικέτα που αντιστοιχεί στην θέση του διανύσματος με την μέγιστη πιθανότητα. Αυτή τη διαδικασία την επαναλαμβάνει για όλες τις παρτίδες του συνόλου. Έπειτα, αφού έχει συγκεντρώσει όλες τις πιθανότητες, τις προβλέψεις, καθώς και τις σωστές ετικέτες για την κάθε εικόνα, συλλέγει μετρικές όπως accuracy, precision, recall, f1 score για κάθε κεφαλή και για κάθε συνδυασμό ετικετών (20 στο σύνολο). Επίσης, συγκεντρώνει κοινές μετρικές δηλαδή joint accuracy, joint precision, joint recall και joint f1 score. Ο αλγόριθμος δοκιμής είναι ο αλγόριθμος 5.

5.4.2 Συνάρτηση υπολογισμού μετρικών `__compute_metrics__`

Η συνάρτηση `__compute_metrics__` είναι υπεύθυνη για τον υπολογισμό των μετρικών μιας κεφαλής. Λαμβάνει ως ορίσματα την κεφαλή για την οποία πρέπει να γίνουν οι υπολογισμοί, τις ετικέτες των εικόνων, τις προβλέψεις του μοντέλου, καθώς και τις πιθανότητες

Algorithm 5 Ο αλγόριθμος της ρουτίνας δοκιμής**Require:** Εκπαιδευμένο μοντέλο f_θ , σύνολο δοκιμής $\mathcal{D}_{\text{test}}$ **Ensure:** Μετρικές ανά κεφαλή, μετρικές ανά συνδυασμό ετικετών και joint μετρικές

```

1: Θέσε το μοντέλο σε evaluation mode
2: Αρχικοποίησε κενές λίστες για  $all\_labels^{(1)}$ ,  $all\_labels^{(2)}$ ,  $all\_preds^{(1)}$ ,  $all\_preds^{(2)}$ ,
    $all\_probs^{(1)}$ ,  $all\_probs^{(2)}$ 
3: for παρτίδα  $b \in \mathcal{D}_{\text{test}}$  do
4:   Φόρτωσε  $inputs$ ,  $labels^{(1)}$ ,  $labels^{(2)}$ 
5:   Υπολόγισε τις εξόδους  $output^{(1)}$ ,  $output^{(2)} \leftarrow f_\theta(inputs)$ 
6:   Υπολόγισε τελικές πιθανότητες με χρήση Softmax σε κάθε κεφαλή
7:   Αποθήκευσε ετικέτες, προβλέψεις και πιθανότητες στις αντίστοιχες λίστες
8: end for
9: Υπολόγισε μετρικές για το  $head^{(1)}$  και για το  $head^{(2)}$  και αποθήκευσέ τις
10: Υπολόγισε joint accuracy
11: Δημιούργησε κωδικοποιήσεις ετικετών για αποθήκευση κοινών προβλέψεων
12: Υπολόγισε joint precision, joint recall, joint f1 score και αποθήκευσέ τις
13: for συνδυασμό κλάσης  $(h_1, h_2)$  από  $(0, 0)$  έως  $(1, 9)$  do
14:   Δημιούργησε ετικέτα και κλάση για το συγκεκριμένο συνδυασμό κλάσεων
15:   Υπολόγισε accuracy, precision, recall και f1 score για την κλάση και αποθήκευσε
       τις τιμές αυτές
16: end for
17: Αποθήκευσε τις μετρικές κάθε συνδυασμού κλάσεων
18:
19: return Μετρικές  $Head_{\text{metrics}}^{(1)} = \{\text{accuracy}_{\text{head}}^{(1)}, \text{precision}_{\text{head}}^{(1)}, \text{recall}_{\text{head}}^{(1)}, \text{f1 score}_{\text{head}}^{(1)}\}$ ,
        $Head_{\text{metrics}}^{(2)} = \{\text{accuracy}_{\text{head}}^{(2)}, \text{precision}_{\text{head}}^{(2)}, \text{recall}_{\text{head}}^{(2)}, \text{f1 score}_{\text{head}}^{(2)}\}$ , joint accuracy,
       joint precision, joint recall, joint f1 score

```

που έθεσε το μοντέλο. Με βάση αυτές τις πληροφορίες υπολογίζει τα accuracy, precision, recall, f1 score της κεφαλής. Η υλοποίηση της συνάρτησης παρουσιάζεται στο σχήμα B.8.

Για να υπολογιστούν μαθηματικά αυτές οι μετρικές, αρχικά ορίζονται οι εξής μεταβλητές: Έστω ότι:

- N : ο συνολικός αριθμός δειγμάτων,
- K : ο αριθμός κλάσεων που ταξινομεί η κεφαλή,
- y_i : η πραγματική ετικέτα του δείγματος $y = 1, 2, \dots, N$,
- $\hat{y}_i$: η προβλεπόμενη κλάση από το μοντέλο για το δείγμα i .

Έτσι, διαμορφώνονται τα:

$$N_{\text{correct}} = \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i=y_i\}} \quad (5.37)$$

$$TP_c = \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i=c\}} \cdot \mathbf{1}_{\{y_i=c\}} \quad (5.38)$$

$$FP_c = \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i=c\}} \cdot \mathbf{1}_{\{y_i \neq c\}} \quad (5.39)$$

$$TN_c = \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i \neq c\}} \cdot \mathbf{1}_{\{y_i \neq c\}} \quad (5.40)$$

$$FN_c = \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i \neq c\}} \cdot \mathbf{1}_{\{y_i=c\}} \quad (5.41)$$

Το True Positives για μια κλάση c (TP_c) εκφράζει τον αριθμό δειγμάτων που ανήκουν στην κλάση c και σωστά προβλέφθηκε ως c . Το False Positives για μια κλάση c (FP_c) εκφράζει τον αριθμό των δειγμάτων που λανθασμένα προβλέφθηκε πως ανήκουν στην c . Το True Negatives για μια κλάση c (TN_c) εκφράζει τον αριθμό των δειγμάτων που σωστά προβλέφθηκε πως δεν ανήκουν στην c και το False Negatives για μια κλάση c (FN_c) εκφράζει τον αριθμό των δειγμάτων που λανθασμένα προβλέφθηκε πως δεν ανήκουν στην c .

Με βάση αυτά μπορούν να οριστούν οι μετρικές:

1. Accuracy κεφαλής j :

$$\text{Accuracy}_{\text{head}}^{(j)} = \frac{N_{\text{correct}}^{(j)}}{N} \times 100 \quad (5.42)$$

2. Precision κεφαλής j :

$$\text{Precision}_{\text{head}}^{(j)} = \left(\frac{1}{K^{(j)}} \sum_{c=1}^{K^{(j)}} \frac{TP_c^{(j)}}{TP_c^{(j)} + FP_c^{(j)}} \right) \times 100 \quad (5.43)$$

3. Recall κεφαλής j :

$$\text{Recall}_{\text{head}}^{(j)} = \left(\frac{1}{K^{(j)}} \sum_{c=1}^{K^{(j)}} \frac{TP_c^{(j)}}{TP_c^{(j)} + FN_c^{(j)}} \right) \times 100 \quad (5.44)$$

4. f1 score κεφαλής j :

$$\text{f1_score}_{\text{head}}^{(j)} = \frac{2 \times \text{Precision}_{\text{head}}^{(j)} \times \text{Recall}_{\text{head}}^{(j)}}{\text{Precision}_{\text{head}}^{(j)} + \text{Recall}_{\text{head}}^{(j)}} \quad (5.45)$$

Όσον αφορά τον υπολογισμό των μετρικών αυτών στην πράξη υπάρχει μια μικρή διαφορά με τους ορισμούς για precision, recall, f1 score που έχουν δοθεί, καθώς το πλαίσιο `scikit_learn` που χρησιμοποιείται για τους υπολογισμούς προσθέτει ένα πολύ μικρό αριθμό $\varepsilon = 10^{-8}$ στον παρονομαστή, για να μην γίνει διαίρεση με το 0. Η μικρή τιμή του ε δεν επηρεάζει τα τελικά αποτελέσματα.

Η μετρική accuracy, όπως στην εκπαίδευση και στον έλεγχο, δείχνει το ποσοστό των σωστών προβλέψεων προς όλες τις προβλέψεις και το precision κεφαλής δείχνει το ποσοστό των περιπτώσεων που το μοντέλο ταξινόμησε σωστά μια εικόνα σε μια κλάση προς το σύνολο των φορών που ταξινόμησε μια εικόνα στην ίδια κλάση. Το precision είναι σημαντικό γιατί δείχνει πόσο αποτελεσματικό είναι το μοντέλο στο να εντοπίζει σωστά τις εικόνες που ανήκουν σε μια κλάση. Το recall κεφαλής δείχνει το ποσοστό των περιπτώσεων που το μοντέλο ταξινόμησε σωστά μια εικόνα σε μια κλάση προς το σύνολο όλων των εικόνων που ανήκουν στην ίδια κλάση. Με αυτή τη μετρική φαίνεται η αποτελεσματικότητα του μοντέλου στον εντοπισμό όλων των εικόνων μιας κλάσης. Τέλος, το f1 score κεφαλής είναι μια μετρική που δείχνει αν υπάρχει ισορροπία μεταξύ precision και recall, δηλαδή αν το μοντέλο μπορεί να ταξινομήσει σωστά τα δείγματα μιας κλάσης χωρίς να εντάξει στην κλάση αυτή εικόνες από άλλες κλάσεις.

5.4.3 Μετρικές ανά κλάση και συνάρτηση `__plot_creation__`

Εκτός από γενικές μετρικές για κάθε κεφαλή το μοντέλο καταγράφει και μετρικές precision, recall, f1 score για κάθε κλάση. Με αυτές τις μετρικές είναι δυνατή η καλύτερη και

σε βάθος κατανόηση της λειτουργίας του μοντέλου, καθώς φαίνονται οι κλάσεις που είναι δύσκολες και εύκολες. Οι μαθηματικοί ορισμοί των μετρικών αυτών είναι παρόμοιοι με εκείνους από την εξίσωση 5.43 έως την 5.45. Στην συγκεκριμένη περίπτωση όμως υπάρχουν οι εξής αλλαγές:

1. για το per-class precision:

$$\text{Precision}_{\text{class}}^{(c)} = \left(\frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \right) \times 100 \quad (5.46)$$

2. για το per-class recall:

$$\text{Recall}_{\text{class}}^{(c)} = \left(\frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \right) \times 100 \quad (5.47)$$

3. για το per class f1 score:

$$\text{F1_score}_{\text{class}}^{(c)} = \frac{2 \times \text{Precision}_{\text{class}}^{(c)} \times \text{Recall}_{\text{class}}^{(c)}}{\text{Precision}_{\text{class}}^{(c)} + \text{Recall}_{\text{class}}^{(c)}} \quad (5.48)$$

Όπως και με τους υπολογισμούς μετρικών κεφαλής έτσι και αυτοί έχουν στον παρονομαστή μια πολύ μικρή σταθερά $\varepsilon = 10^{-8}$ για να αποτραπεί διαίρεση με το 0. Οι μετρικές για τους συνδυασμούς κλάσεων είναι ακριβώς ίδιες, καθώς κάθε συνδυασμός αντιμετωπίζεται ως ξεχωριστή κλάση και ο κανόνας για τις συναρτήσεις-δείκτες που καθορίζουν τότε μια εικόνα ανήκει σε ένα συνδυασμό κλάσεων είναι ο εξής:

$$I_{(c_1, c_2)} = \mathbf{1}_{\{(\hat{y}_i^{(1)}, \hat{y}_i^{(2)}) = (c_1, c_2)\}} \quad (5.49)$$

Συνεπώς, όποτε η πρόβλεψη $\hat{y}_i^{(1)}$ της πρώτης κεφαλής είναι ίδια με την ετικέτα c_1 και η $\hat{y}_i^{(2)}$ της δεύτερης κεφαλής είναι ίδια με την ετικέτα c_2 για εικόνα i , τότε η τιμή της $I_{(c_1, c_2)}$ είναι 1 και έχει γίνει η πρόβλεψη σωστά, αλλιώς η τιμή της $I_{(c_1, c_2)}$ είναι 0 και έχει γίνει λάθος.

Στην εργασία αυτή, τον υπολογισμό των απλών μετρικών κάθε κλάσης τον αναλαμβάνει η συνάρτηση `__plot_creation__`. Η συνάρτηση αυτή λαμβάνει τις ετικέτες, τις εξόδους και τις προβλέψεις του μοντέλου για τους υπολογισμούς και δημιουργεί και αρχεία .csv στα οποία αποθηκεύει αναλυτικά για κάθε εικόνα ποια είναι η σωστή ετικέτα, ποια δόθηκε από το μοντέλο και τις πιθανότητες κάθε κλάσης. Επίσης, δημιουργεί τους πίνακες σύγχυσης (confusion matrices) για το μοντέλο. Ο κώδικας που υλοποιεί την συνάρτηση `__plot_creation__` παρουσιάζεται στο σχήμα B.9.

5.4.4 Υπολογισμός κοινών μετρικών

Τέλος υπολογίζονται και τα κοινά accuracy, precision, recall και f1 score και των 2 κεφαλών για να δημιουργηθεί μια συνολική εικόνα για την απόδοση των μοντέλων. Για τους υπολογισμούς αυτούς ενσωματώνουμε την indicator function της εξίσωσης 5.49 στις εξισώσεις για per class μετρικές. Έτσι έχουμε:

- K_{joint} : Συνολικός αριθμός joint κλάσεων,
- N : πλήθος δειγμάτων.

Ισχύει $\forall k \in \{1, 2, \dots, K\}$:

- $\hat{y}_i$: η προβλεπόμενη κλάση για το δείγμα i ,
- y_i : η πραγματική κλάση για δείγμα i .

$$\text{Accuracy}_{\text{joint}} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}_{\{\hat{y}_i = y_i\}} \quad (5.50)$$

$$\text{Precision}_{\text{joint}} = \frac{1}{K_{\text{joint}}} \sum_{k=1}^{K_{\text{joint}}} \text{Precision}_k \quad (5.51)$$

$$\text{Recall}_{\text{joint}} = \frac{1}{K_{\text{joint}}} \sum_{k=1}^{K_{\text{joint}}} \text{Recall}_k \quad (5.52)$$

$$\text{F1_score}_{\text{joint}} = \frac{2 \times \text{Precision}_{\text{joint}} \times \text{Recall}_{\text{joint}}}{\text{Precision}_{\text{joint}} + \text{Recall}_{\text{joint}}} \quad (5.53)$$

Κεφάλαιο 6

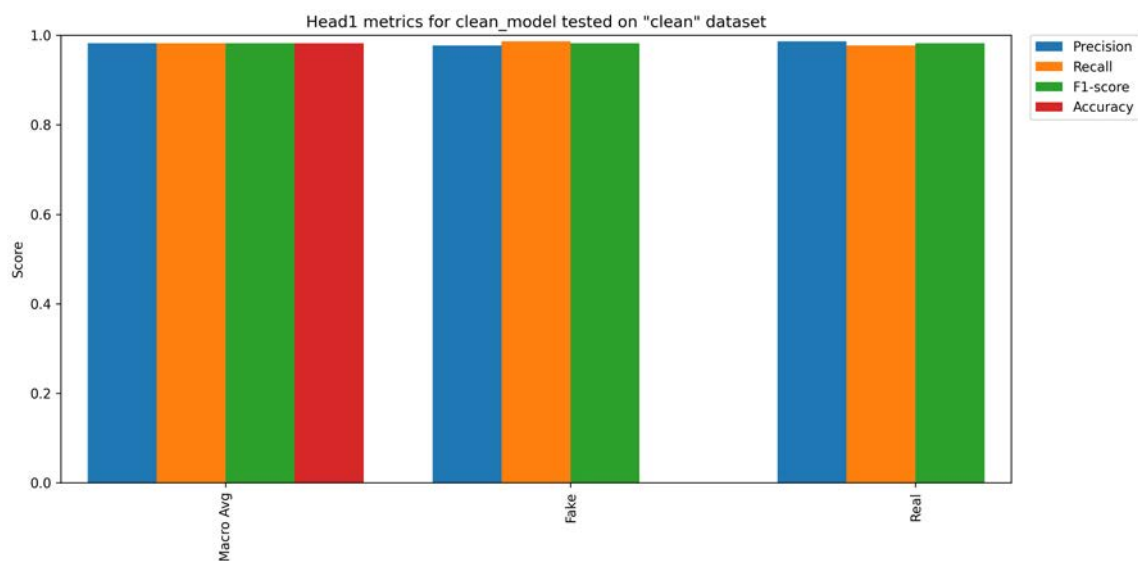
Πειραματικά αποτελέσματα

Στο κεφάλαιο αυτό εξετάζονται τα αποτελέσματα της δοκιμής του ResNet50Classifier. Αρχικά, ερμηνεύονται τα αποτελέσματα των κεφαλών ξεχωριστά και η δυνατότητα ταξινόμησης στις δύο ετικέτες. Τέλος, εξετάζεται η συνολική ικανότητα του μοντέλου να τοποθετεί σωστά ταυτόχρονα και τις δύο ετικέτες σε μια εικόνα.

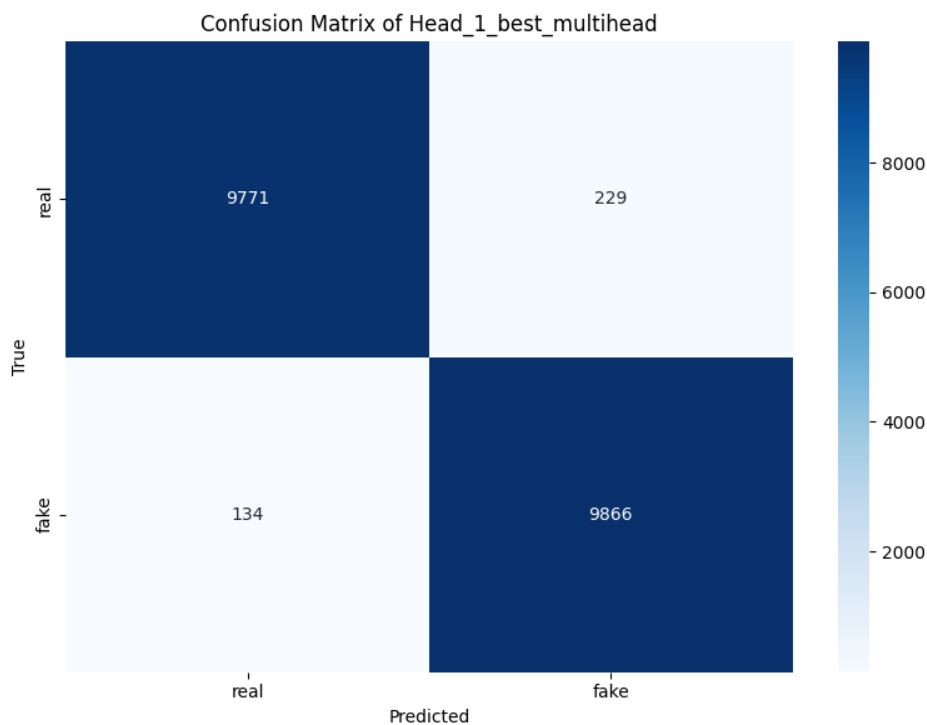
6.1 Αποτελέσματα κεφαλών

Αρχικά, τα σχήματα 6.1 και 6.2 αναφέρονται στην κεφαλή head1 που είναι υπεύθυνη για την δυαδική ταξινόμηση αληθινών και συνθετικών εικόνων. Στο πρώτο σχήμα αναφέρονται τα recall, precision, f1 score για κάθε κλάση και οι μέσοι όροι. Επίσης, καταγράφεται η μέση ακρίβεια. Στην κλάση real το recall ισούται με 97.71%, το precision είναι 98.65% και το f1 score ισούται με 98.18%, ενώ στην κλάση fake το precision ισούται με 97.73%, το recall ισούται με 98.66% και το f1 score ισούται με 98.18%. Από αυτές τις μετρήσεις φαίνεται πως το μοντέλο είναι αρκετά καλό στον εντοπισμό όλων των εικόνων των κλάσεων. Αυτό αποδεικνύεται και από το υψηλό ποσοστό accuracy που ισούται με 98.19%. Κατά μέσο όρο λοιπόν, ταξινομεί σωστά τις 19638 από τις 20000 εικόνες. Επίσης, το μέσο recall, precision και το μέσο f1 score για την κεφαλή 1 είναι περίπου 98.19%. Επομένως, η λειτουργία του μοντέλου είναι σωστή και η ικανότητα ταξινόμησης στις κλάσεις ισορροπημένη.

Από το δεύτερο σχήμα (6.2) παρατηρείται πως το μοντέλο παρουσιάζει ελαφρώς αυξημένη δυσκολία στο να ταξινομήσει σωστά τις αληθινές εικόνες, καθώς αναγνωρίζει λανθασμένα 229 αληθινές εικόνες ως συνθετικές. Αυτό υποστηρίζεται και από τις μετρικές, αφού το recall_{real} είναι μικρότερο από το recall_{fake} .

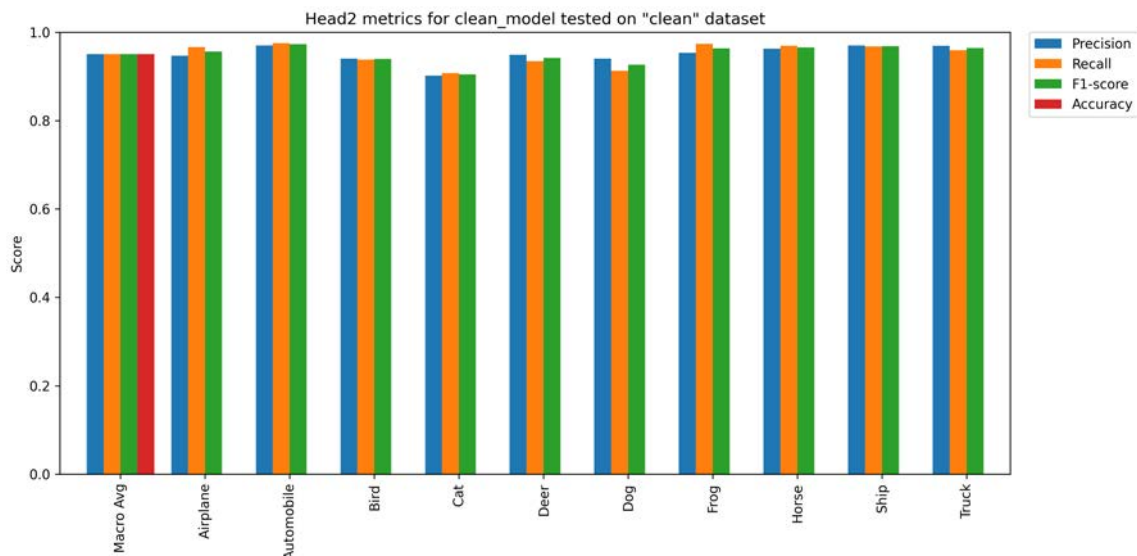


Σχήμα 6.1: Οι μετρικές precision, recall, f1 score της πρώτης κεφαλής αναλυτικά για κάθε κλάση και οι μέσοι όροι μαζί με μέση ακρίβεια.



Σχήμα 6.2: Confusion matrix πρώτης κεφαλής

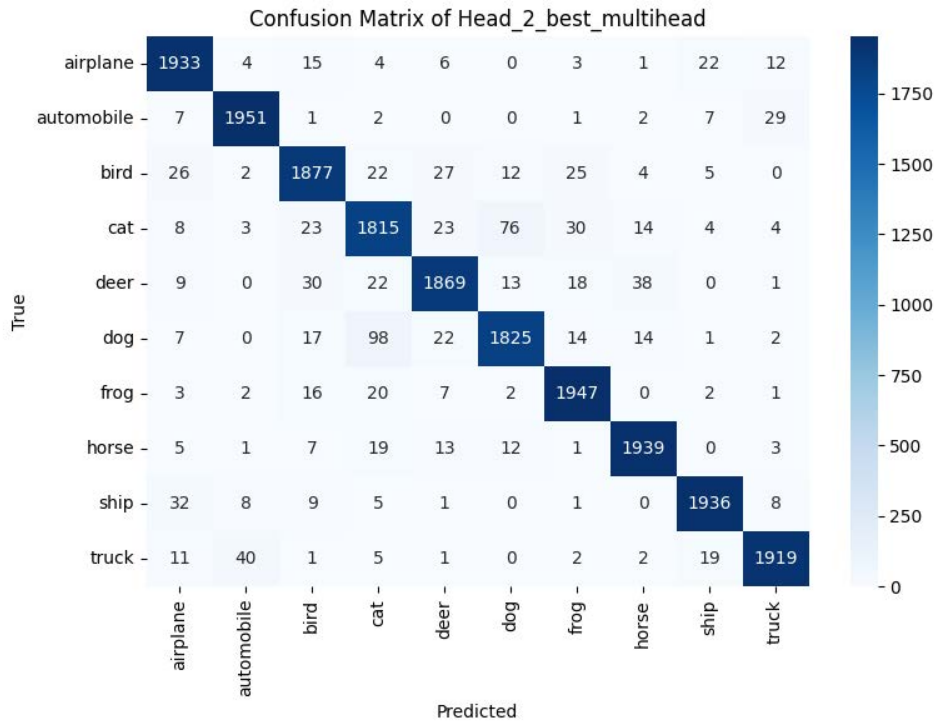
Παρατηρώντας και τα δύο σχήματα (6.3 και 6.4) για την δεύτερη κεφαλή είναι φανερό πως η ταξινόμηση σε δέκα κλάσεις είναι αρκετά πιο δύσκολη και το μοντέλο κάνει περισσότερα λάθη. Αρχικά ο μέσος όρος των accuracy, recall, precision και f1 score ισούται με 95.06%, που είναι περίπου 3% μικρότερο από τα αντίστοιχα αποτελέσματα της άλλης εργασίας ταξινόμησης. Μάλιστα, υπάρχει μεγάλη διασπορά μεταξύ των επιμέρους κλάσεων, αφού υπάρχουν κλάσεις, όπως automobile, frog και ship με precision, recall 1-2% υψηλότερο κατά μέσο όρο, ενώ κλάσεις, όπως cat και dog έχουν recall 90.75%, 91.25% και precision 90.2%, 94.07% αντίστοιχα, που είναι αρκετά χαμηλότερα από τους μέσους όρους. Με βάση το confusion matrix της κεφαλής το μοντέλο δυσκολεύεται να ξεχωρίσει εικόνες της κλάσης cat από εικόνες της κλάσης dog αφού 98 εικόνες dog αναγνωρίζονται ως cat και 76 εικόνες cat ως dog. Η δυσκολία αυτή οφείλεται πιθανότατα στην ύπαρξη πολλών παρόμοιων στοιχείων ανάμεσα στις εικόνες, όπως παρόμοια χρώματα και χαρακτηριστικά, που δυσχεραίνει την αναγνώριση προτύπων. Παρά τις δυσκολίες όμως εξακολουθεί να είναι αποτελεσματικό το μοντέλο και να πετυχαίνει υψηλά ποσοστά σωστών προβλέψεων.



Σχήμα 6.3: Οι μετρικές precision, recall, f1 score της δεύτερης κεφαλής αναλυτικά για κάθε κλάση και οι μέσοι όροι μαζί με μέση ακρίβεια.

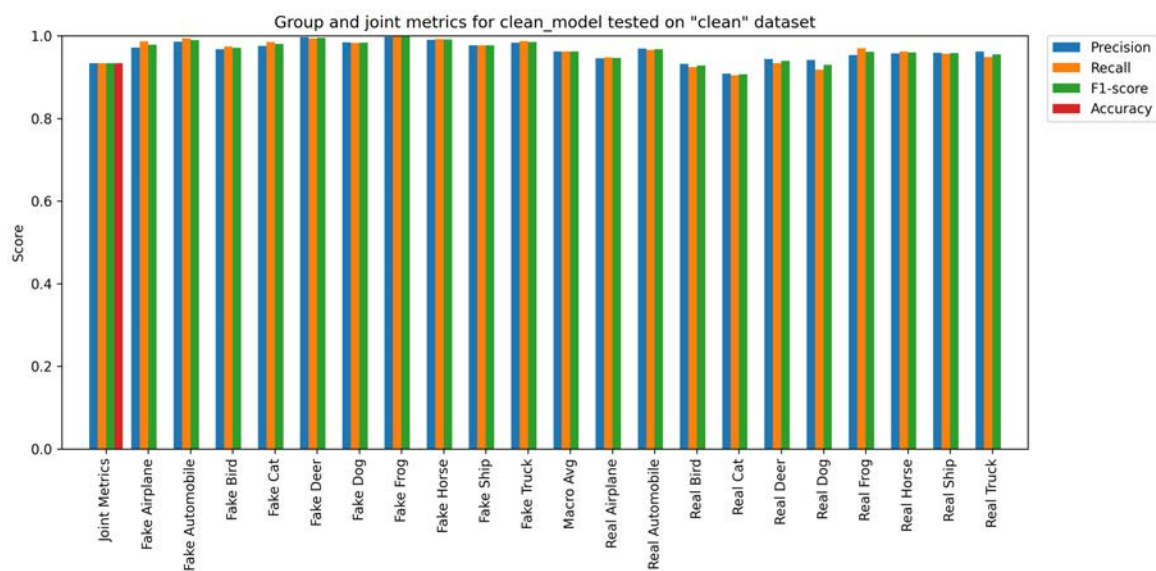
6.2 Αποτελέσματα συνδυασμών κλάσεων

Από το σχήμα 6.5 είναι φανερό πως οι συνθετικές εικόνες εντοπίζονται ευκολότερα από τις πραγματικές ανεξαρτήτως του αντικειμένου που απεικονίζουν. Ειδικότερα οι εικόνες που



Σχήμα 6.4: Confusion Matrix για την δεύτερη κεφαλή

ανήκουν στις κλάσεις fake και frog αναγνωρίζονται σωστά στο 99.9% των περιπτώσεων, δηλαδή μόλις μία εικόνα από αυτό το σύνολο δεν ταξινομήθηκε σωστά. Με βάση το σχήμα, επίσης, φαίνεται πως το μοντέλο ξεχωρίζει περισσότερο τις κλάσεις real cat και real dog από τις αντίστοιχες fake και πιθανότατα το υψηλό ποσοστό λάθος ταξινόμησης (misclassification) μεταξύ των κλάσεων dog, cat αφορά αληθινές εικόνες και όχι συνθετικές. Τέλος, με βάση τις κοινές μετρικές του ίδιου σχήματος 6.5, είναι φανερό πως το μοντέλο είναι πολύ αποτελεσματικό συνολικά, καθώς πετυχαίνει 93.45% accuracy, 93.38% precision, 93.41% recall και 93.38% f1 score, ταξινομώντας σωστά τις 18690 εικόνες.



Σχήμα 6.5: Οι κοινές μετρικές και οι μετρικές για κάθε συνδυασμό ετικετών των δύο κεφαλών

Κεφάλαιο 7

Αντοχή μοντέλου σε εικόνες με θόρυβο

Το ResNet50Classifier αποδείχθηκε αποτελεσματικό για ταξινόμηση εικόνων που δεν έχουν καθόλου θόρυβο. Στην πραγματικότητα όμως οι εικόνες έχουν συνήθως θόρυβο, το επίπεδο του οποίου διαφέρει. Μια εφαρμογή ταξινόμησης, προκειμένου να είναι πρακτική, πρέπει να είναι ικανή να αναγνωρίσει ακόμη και εικόνες που δεν είναι καθαρές, με σκοπό να αποτρέψει κενά ασφάλειας και την επιτηδευμένη εισαγωγή θορύβου που έχει ως στόχο την εξαπάτηση του συστήματος. Για τον λόγο αυτό διεξάγεται ένα ακόμα πείραμα στο οποίο δημιουργούνται νέα σύνολα δεδομένων που βασίζονται στο CIFAKE [5, 6] αλλά περιέχουν εικόνες με θόρυβο. Δημιουργούνται τέσσερα νέα μοντέλα ResNet50Classifier που έχουν ίδια δομή με το προηγούμενο, αλλά εκπαιδεύονται στα θορυβώδη σύνολα δεδομένων και κάθε ένα από αυτά τα καινούρια μοντέλα καθώς και αυτό του πρώτου πειράματος εξετάζονται σε πέντε διαφορετικά σύνολα δοκιμής, από τα οποία το ένα έχει εικόνες χωρίς θόρυβο και τα άλλα τέσσερα εικόνες με διαφορετικά είδη θορύβου. Στόχος του πειράματος αυτού είναι η αξιολόγηση της αποτελεσματικότητας του μοντέλου σε εικόνες με διαφορετικές συνθήκες θορύβου.

7.1 Είδη θορύβου υπό εξέταση

Στο πείραμα αυτό εξετάζονται δύο είδη θορύβου, ο Γκαουσιανός θόρυβος (Gaussian noise) και η συμπίεση εικόνων jpeg (JPEG compression), καθώς και ο αντίκτυπός τους στην επίδοση του δικτύου. Ο Γκαουσιανός θόρυβος υπάρχει στις φωτογραφίες, προκαλείται από φυσικά αίτια, όπως η θερμική ταλάντωση των ατόμων και η διακριτή φύση της ακτινοβολίας τους και επηρεάζει κυρίως τα εικονοστοιχεία [47]. Σε αυτό το είδος θορύβου οι τιμές

θορύβου ακολουθούν την Γκαουσιανή κατανομή και αυτή η φυσική κανονική κατανομή καθιστά τον θόρυβο ιδανικό για μοντελοποίηση απόκρισης συστημάτων σε θορυβώδη είσοδο. Μαθηματικά, εκφράζεται ως εξής:

Έστω η κατανομή για τον θόρυβο n :

$$n \sim \mathcal{N}(\mu, \sigma^2) \quad (7.1)$$

όπου:

- μ είναι ο μέσος όρος της κατανομής,
- σ^2 είναι η διασπορά που καθορίζει τον διαμοιρασμό των τιμών θορύβου.

Αυτή η κατανομή εκφράζεται από την συνάρτηση πυκνότητας πιθανότητας $p(n)$:

$$p(n) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(n-\mu)^2}{2\sigma^2}\right) \quad (7.2)$$

Η $p(n)$ εκφράζει την πιθανοφάνεια να εμφανιστεί το n και η κατανομή είναι συμμετρική γύρω από το μ . Έστω λοιπόν μια εικόνα $I \in \mathbb{R}^{W \times H}$ όπου W, H είναι οι διαστάσεις πλάτους και ύψους της εικόνας και έστω $I(x, y)$ η αναπαράσταση των τιμών των εικονοστοιχείων μιας καθαρής εικόνας για τα εικονοστοιχεία (x, y) . Επίσης, $\tilde{I}(x, y)$ η αναπαράσταση της αντίστοιχης εικόνας με θόρυβο. Η προσθήκη Γκαουσιανού θορύβου ορίζεται ως:

$$\tilde{I}(x, y) = I(x, y) + n(x, y), \quad \text{με} \quad n(x, y) \sim \mathcal{N}(0, \sigma^2) \quad (7.3)$$

Κάθε εικονοστοιχείο της εξίσωσης 7.3 επηρεάζεται ανεξάρτητα από μία μεταβλητή της κανονικής κατανομής με μέσο το 0. Τέλος, λόγω του τρόπου ορισμού ισχύει ότι:

$$\mathbb{E}[n] = \mu \quad (7.4)$$

και

$$\text{Var}(n) = \sigma^2 \quad (7.5)$$

Από αυτές τις εξισώσεις συνεπάγεται ότι το μ καθορίζει την μετατόπιση της τιμής της φωτεινότητας για όλη την εικόνα, άρα όταν $\mu = 0$ δεν υπάρχει μετατόπιση και η τιμή της σ επηρεάζει αναλόγως τις τιμές των εικονοστοιχείων, καθώς μεγαλύτερο σ ισοδυναμεί με μεγαλύτερη αλλαγή τιμών.

Η συμπίεση εικόνων jpeg, σε αντίθεση με τον προηγούμενο θόρυβο, δεν εμφανίζεται φυσικά αλλά εφαρμόζεται σε φωτογραφίες με σκοπό την μείωση του μεγέθους της εικόνας

και την σύμπτυξη της πληροφορίας. Σύμφωνα με αυτό οι λιγότερο σημαντικές πληροφορίες αφαιρούνται με χρήση διακριτού μετασχηματισμού συνημιτόνου (Discrete Cosine Transform - DCT) και έπειτα δημιουργείται μια νέα εικόνα μικρότερου μεγέθους. Επειδή η εικόνα εισόδου διαφέρει από αυτή της εξόδου ως προς το περιεχόμενο, έστω και ελάχιστα, το JPEG compression θεωρείται τεχνική σύμπτυξης με απώλειες [48]. Μπορούμε να ορίσουμε την διαδικασία ως εξής:

Έστω, όπως και προηγουμένως, μια εικόνα $I(x, y)$ που αντιπροσωπεύει τις τιμές των εικονοστοιχείων (x, y) μιας εικόνας $I \in \mathbb{R}^{W \times H}$ όπου (W, H) οι διαστάσεις μήκους, ύψους της εικόνας.

Ακολουθούν κατά σειρά τα βήματα:

1. Μετατροπή χρωματικού χώρου από RGB σε YCbCr με σκοπό την απομόνωση της φωτεινότητας
2. Η εικόνα διαιρείται σε μη επικαλυπτόμενα κομμάτια 8×8
3. Για κάθε κομμάτι B εφαρμόζεται ο DCT:

$$F(u, v) = \frac{1}{4}C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 I(x, y) \cos \left[\frac{(2x+1)u\pi}{16} \right] \cos \left[\frac{(2y+1)v\pi}{16} \right] \quad (7.6)$$

$$\text{με } C(k) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{αν } k = 0 \\ 1, & \text{αν } k > 0 \end{cases}$$

4. Έπειτα κάθε συντελεστής DCT $F(u, v)$ διαιρείται με προκαθορισμένο στοιχείο κβαντοποίησης $Q(u, v)$ και στρογγυλοποιείται:

$$\tilde{F}(u, v) = \text{round} \left(\frac{F(u, v)}{Q(u, v)} \right) \quad (7.7)$$

5. Τέλος, εφαρμόζεται σειριοποίηση και κωδικοποίηση εντροπίας για συμπίεση.

Η πληροφορία χάνεται στο επίπεδο κβαντοποίησης καθώς η πράξη της στρογγυλοποίησης δεν είναι αναστρέψιμη. Το τελικό μέρος ισούται με:

$$\hat{F}(u, v) = \tilde{F}(u, v) \cdot Q(u, v) \quad (7.8)$$

και με εφαρμογή αντίστροφου σχηματισμού διακριτού συνημιτόνου (Inverse Discrete Cosine Transform) ισχύει:

$$\hat{I}(x, y) = \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v) \hat{F}(u, v) \cos \left[\frac{(2x+1)u\pi}{16} \right] \cos \left[\frac{(2y+1)v\pi}{16} \right] \quad (7.9)$$

Η τελική εικόνα $\hat{I}(x, y)$ εμφανίζει απώλεια λεπτομερειών, όπως θόλωση σε περιοχές υψηλής συχνότητας και blocking artifacts.

7.1.1 Αντικείμενο GaussianNoise

Το αντικείμενο GaussianNoise είναι υπεύθυνο για την εισαγωγή κανονικού θορύβου σε εικόνες και διαθέτει δύο συναρτήσεις την `__init__` που αρχικοποιεί σημαντικές παραμέτρους και την `__call__` που πραγματοποιεί την προσθήκη θορύβου. Η `__init__` δέχεται τρία ορίσματα, τον μέσο όρο `mean`, την διασπορά `std` και την πιθανότητα `p`. Τα πρώτα δύο επηρεάζουν τα μ και σ της κανονικής κατανομής, όπως ορίστηκε από την συνάρτηση πιθανοφάνειας 7.2 και η τρίτη συνάρτηση καθορίζει το ποσοστό των εικονοστοιχείων της εικόνας που δέχονται αυτό τον μετασχηματισμό και κυμαίνεται από 0.0-1.0. Η συνάρτηση `__call__` χρησιμοποιεί το διάνυσμα της εικόνας και δημιουργεί ένα αντίγραφο αυτής, στο οποίο έχει εισαγάγει Γκαουσιανό θόρυβο. Έπειτα, αυτό το διάνυσμα περνά από τη μάσκα και επικαλύπτει το αρχικό, με αποτέλεσμα να επηρεάζεται μόνο το ποσοστό εικονοστοιχείων που είναι επιθυμητό. Η υλοποίηση του αντικειμένου GaussianNoise παρουσιάζεται στο σχήμα B.10

Μαθηματικά η λειτουργία αυτού του αντικειμένου με τη βοήθεια των εξισώσεων της ενότητας 7.1 μπορεί να εκφραστεί ως εξής:

Αρχικά, η είσοδος είναι της μορφής $I \in [0, 1]^{H \times W \times C}$, $H = W = 224$, $C = 3$ και ο θόρυβος εφαρμόζεται ανεξάρτητα σε κάθε θέση και κανάλι. Σε αντίθεση με την εξίσωση 7.3 δεν εφαρμόζεται ο θόρυβος καθολικά, αλλά μόνο σε τυχαίο υποσύνολο εικονοστοιχείων και καναλιών, βάσει δυαδικής μάσκας Bernoulli. Έτσι, ορίζεται:

$$\tilde{I}(x, y, c) = \text{clip}(I(x, y, c) + M(x, y, c) \cdot n(x, y, c), 0, 1) \quad (7.10)$$

όπου:

- $I(x, y, c)$: η αρχική τιμή του εικονοστοιχείου (x, y) στο κανάλι c ,
- $\tilde{I}(x, y, c)$: τελική τιμή μετά την προσθήκη θορύβου,
- $n(x, y, c) \sim \mathcal{N}(\mu, \sigma^2)$: τυχαία τιμή θορύβου όπως προκύπτει από συνάρτηση πιθανοφάνειας,
- $M(x, y, c) \sim \text{Bernoulli}(p)$: μάσκα που ορίζει εάν εισάγεται θόρυβος,
- $\text{clip}(\cdot, 0, 1)$: περιορίζει τελικές τιμές εντός του διαστήματος $[0, 1]$.

Algorithm 6 Εφαρμογή GaussianNoise σε εικόνα $I \in \mathbb{R}^{224 \times 224 \times 3}$

Require: Εικόνα ως διάνυσμα $H \times W \times C$, p ως ποσοστό εικόνας που δέχεται θόρυβο, μ ως μέση τιμή, σ ως τυπική απόκλιση.

```

1: Αρχικοποίησε  $\tilde{I} \leftarrow I$ 
2: for θέση  $(x, y)$  στην εικόνα do
3:   for κανάλι  $c \in \{1, 2, 3\}$  do
4:     Λάβε  $r \sim \mathcal{U}(0, 1)$ 
5:     if  $r < p$  then
6:       Λάβε  $n \sim \mathcal{N}(\mu, \sigma^2)$ 
7:        $\tilde{I}(x, y, c) \leftarrow I(x, y, c) + n$ 
8:     end if
9:   end for
10: end for
11: Περιορισμός τιμών:  $\tilde{I}(x, y, c) \leftarrow \min(\max(\tilde{I}(x, y, c), 0), 1) \forall (x, y, c)$ 
12: return  $\tilde{I}$ 

```

7.1.2 Αντικείμενο JPEGCompression

Για να μοντελοποιηθεί η σύμπτυξη jpeg αναπτύχθηκε το αντικείμενο JPEGCompression. Όπως και στο GaussianNoise δημιουργούνται δύο συναρτήσεις, η `__init__` που αρχικοποιεί σημαντικές παραμέτρους και η `__call__` που εφαρμόζει τον μετασχηματισμό στην εικόνα. Η `__init__` δέχεται σαν όρισμα έναν ακέραιο αριθμό quality και καθορίζει την σύμπτυξη της νέας εικόνας, αφού όσο μεγαλύτερο είναι το quality τόσο μικρότερη είναι η σύμπτυξη και η απώλεια πληροφοριών με μέγιστη τιμή το 100 όπου δεν γίνεται κάποια αλλαγή. Η μέθοδος `__call__` γράφει τα περιεχόμενα της εικόνας σε έναν buffer και εφαρμόζει JPEG Compression όπως περιγράφηκε στην ενότητα 7.1. Τέλος, επιστρέφει τον δείκτη του buffer στην αρχική θέση και ανοίγει την νέα εικόνα ως rgb εικόνα μέσω του αντιγράφου του buffer. Η υλοποίηση του αντικειμένου JPEGCompression παρουσιάζεται στο σχήμα B.11. Ο αλγόριθμος του JPEGCompression παρουσιάζεται στον αλγόριθμο 7.

Algorithm 7 Ο αλγόριθμος του JPEGCompression**Require:** I : είσοδο εικόνας, q : τιμή ποιότητας τελικής εικόνας.

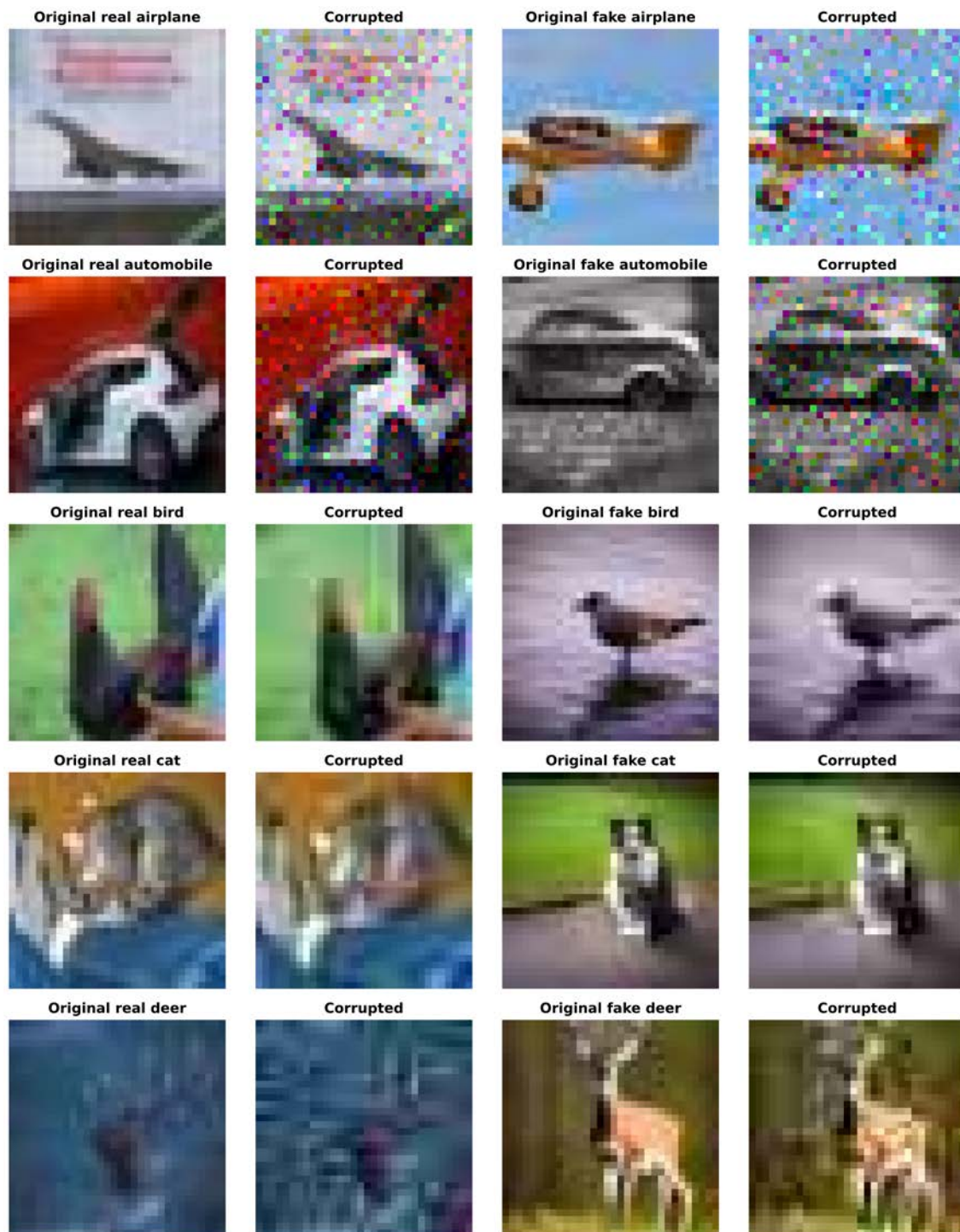
- 1: Δημιούργησε ένα προσωρινό buffer B .
- 2: Αποθήκευσε την I στο B με ποιότητα q .
- 3: Επαναφόρτωσε την $\hat{I}$ από το B .
- 4: **return** την παραμορφωμένη εικόνα $\hat{I}$ από το B .

7.2 Δημιουργία εικόνων με θόρυβο

Αφού δημιουργήθηκαν τα αντικείμενα θορύβου, πρέπει να δημιουργηθεί και το κατάλληλο πλαίσιο για την δημιουργία των συνόλων εκπαίδευσης, ελέγχου και δοκιμής με εικόνες με θόρυβο. Για να πραγματοποιηθεί αυτό, δημιουργήθηκαν τρεις νέες συναρτήσεις οι `gen_encoding_map`, `create_transforms_cache` και `decode_transform` καθώς και ένας διαχειριστής δεδομένων ο `CIFAKENoisyDatasetHandler` που χειρίζεται τα νέα σύνολα. Επίσης, πρέπει εκ νέου να δημιουργηθούν για κάθε περίπτωση τα νέα dataloaders. Στα πλαίσια του πειράματος κατασκευάζονται τέσσερα νέα σύνολα. Στο σχήμα 7.1 παρουσιάζονται κάποιες αρχικές εικόνες μαζί με τις αντίστοιχες παραμορφωμένες εικόνες. Προκειμένου ο αντίκτυπος του θορύβου να γίνει αισθητός στον άνθρωπο, ο κανονικός θόρυβος επηρεάζει το 30% της εικόνας και η ποιότητα του JPEG Compression είναι 50. Στα πειράματα χρησιμοποιούνται πιο ήπιες παράμετροι θορύβου που αναλύονται στην ενότητα 7.2.1.

7.2.1 Μελέτη συναρτήσεων

Αρχικά, η συνάρτηση `gen_encoding_map` λαμβάνει ως ορίσματα το αρχικό σύνολο δεδομένων και ένα string `noise_level` που καθορίζει τα είδη θορύβου που θα χρησιμοποιηθούν. Θέτει πάντα οι μισές εικόνες να είναι καθαρές, δηλαδή χωρίς θόρυβο και έπειτα ανάλογα με το `noise_level` κατασκευάζει την κατανομή των εικόνων με θόρυβο, έτσι ώστε να είναι ισομοιρασμένες σε όλες τις κλάσεις, για να μην υπάρξει διαφορά στην κατανομή δεδομένων σε σχέση με το σύνολο του αρχικού πειράματος. Αν το `noise_level` τεθεί στο επίπεδο `all`, τότε οι μη καθαρές εικόνες θα περιέχουν είτε Gaussian noise είτε JPEG Compression είτε και τις δύο παραμορφώσεις ταυτόχρονα. Συγκεκριμένα, το 16.66% περίπου των εικόνων θα υποστούν JPEG Compression με $q = 80$, το 16.66% Gaussian noise και το υπόλοιπο 16.66% θα έχει και τους 2 μετασχηματισμούς p, q . Αν το `noise_level` είναι `both` τότε το 25% των εικόνων θα έχουν JPEG Compression, το 25% Gaussian noise και το υπόλοιπο 50% θα είναι



Σχήμα 7.1: Παρουσιάζονται οι αρχικές εικόνες μαζί με τις αντίστοιχες παραμορφωμένες εικόνες (corrupted). Στις πρώτες 2 σειρές προστίθεται μόνο κανονικός θόρυβος, στις επόμενες 2 γίνεται μόνο JPEG Compression και στην τελευταία σειρά προστίθενται και τα 2 είδη θορύβου.

καθαρές. Αν το `noise_level` είναι `jpeg` τότε υπάρχει 50/50 διαχωρισμός ανάμεσα σε καθαρές εικόνες και JPEG Compression. Αντίστοιχα, αν `noise_level` είναι `Gaussian` το 50% των εικόνων έχουν `Gaussian noise` αντί για JPEG Compression. Αφού έχει μοιράσει σε όλες τις κλάσεις ίσο αριθμό εικόνων κάθε είδους θορύβου, κατασκευάζει στην ουσία μια απεικόνιση ή “χάρτη” που κωδικοποιεί την κατανομή των διαφορετικών ειδών θορύβου στις εικόνες, ώστε να συνδεθεί η θέση εγγραφής κάθε εικόνας στο νέο σύνολο δεδομένων με τους κατάλληλους μετασχηματισμούς. Η υλοποίηση της `gen_encoding_map` παρουσιάζεται στο σχήμα B.12.

Τα σύνολα δεδομένων δεν έχουν εικόνες μόνο με θόρυβο αλλά και καθαρές εικόνες για διάφορους λόγους. Αρχικά, με αυτό τον τρόπο δημιουργείται ένα πιο ρεαλιστικό σενάριο δοκιμής του μοντέλου σε ταξινόμηση εικόνων, καθώς δεν έχει πάντα το σύνολο των εικόνων θόρυβο. Επίσης, με την χρήση καθαρών δειγμάτων το δίκτυο μαθαίνει επιπλέον μοτίβα και όχι μόνο εκείνα του θορύβου, επιτυγχάνοντας δύο στόχους ταυτόχρονα. Πρώτον, μπορεί να αναγνωρίζει και να ταξινομεί σωστά αυτές τις εικόνες και δεύτερον εκπαιδεύεται καλύτερα στον διαχωρισμό μεταξύ των διαφόρων ειδών θορύβου, διότι δεν υπάρχει κίνδυνος να κάνει *overfitting* σε διάφορα πρότυπα θορύβου.

Μετά ακολουθούν οι συναρτήσεις `decode_transform` και η `create_transforms_cache`. Η πρώτη λαμβάνει ως όρισμα την κωδικοποίηση μιας εικόνας και ένα `string mode` που καθορίζει αν το σύνολο θα χρησιμοποιηθεί στην εκπαίδευση ή στον έλεγχο/δοκιμή του μοντέλου. Στην αρχή, αποκωδικοποιεί την απεικόνιση και έπειτα ελέγχει το `mode`. Αν η εικόνα ανήκει σε σύνολο εκπαίδευσης, τότε κατασκευάζει την λίστα μετασχηματισμών προσθέτοντας πρώτα τους περισσότερους μετασχηματισμούς εκπαίδευσης που χρησιμοποιούνται και στο καθαρό σύνολο εκπαίδευσης του πρώτου πειράματος. Οι μόνοι μετασχηματισμοί εκπαίδευσης που δεν προστίθενται ακόμη είναι οι `transforms.ToTensor()` και `transforms.Normalize(imagenet_mean, imagenet_std)` του σχήματος B.4. Έπειτα, από το αποτέλεσμα της αποκωδικοποίησης αν η εικόνα πρέπει να υποστεί JPEG Compression, τότε προστίθεται ο μετασχηματισμός αυτός στη λίστα μετασχηματισμών. Ακολουθεί, ανεξαρτήτως του `mode` και του `encoding`, η προσθήκη του μετασχηματισμού `transforms.ToTensor()`. Ακολούθως, αν στην εικόνα προστίθεται `Gaussian noise`, τότε προστίθεται ο αντίστοιχος μετασχηματισμός και τέλος, ανεξαρτήτως πάλι `mode` και `encoding`, προστίθεται ο μετασχηματισμός `transforms.Normalize(imagenet_mean, imagenet_std)`. Στην ουσία δηλαδή δημιουργήθηκε η λίστα μετασχηματισμών για την εικόνα, για να δοθεί μετά ως παράμετρος για τον κατάλληλο διαχειριστή δεδομένων. Οι βασικοί μετασχηματισμοί των εικόνων παρουσιάζον-

ται στο σχήμα B.13 και η υλοποίηση της συνάρτησης `decode_transform` παρουσιάζεται στο σχήμα B.14.

Τέλος, η `create_transforms_cache` λαμβάνει ως όρισμα το `mode` που είναι ίδιο με αυτό της `decode_transform`. Η `create_transforms_cache` είναι επί της ουσίας μια βοηθητική συνάρτηση που αρχικά αποθηκεύει όλες τις δυνατές απεικονίσεις σε μια λίστα. Έπειτα, δημιουργεί ένα hash map που περιέχει όλες τις κωδικοποιήσεις και σε κάθε bucket του hash map αντιστοιχεί την κατάλληλη λίστα μετασχηματισμών καλώντας την `decode_transform` με ορίσματα το `encoding` και το `mode`. Με αυτό τον τρόπο δημιουργείται μία δομή που λειτουργεί ως ψευδής μνήμη cache, αφού πλέον ο χρόνος πρόσβασης για κάθε κωδικοποίηση μιας εικόνας είναι $O(1)$. Επίσης, δεν επιβαρύνονται σημαντικά οι απαιτήσεις για μνήμη, καθώς η δομή απαιτεί $O(n)$ χώρο. Επομένως, η διαδικασία δημιουργίας των νέων συνόλων θα είναι ταχύτατη, αφού ο χρόνος απόκτησης των σωστών μετασχηματισμών είναι $O(1)$ αντί για $O(n)$ που ήταν αρχικά, άρα η επιτάχυνση για 120000 εικόνες θα είναι πολύ σημαντική. Η υλοποίηση της συνάρτησης `create_transforms_cache` παρουσιάζεται στο σχήμα B.15.

7.2.2 Διαχειριστής δεδομένων CIFAKENoisyDatasetHandler

Ο διαχειριστής δεδομένων `CIFAKENoisyDatasetHandler` είναι βοηθητικός για τον διαχειριστή δεδομένων `CIFAKEDatasetHandler`. Η λειτουργία του είναι να διαχειρίζεται τα σύνολα δεδομένων με θόρυβο που βασίζονται στα αρχικά σύνολα δεδομένων. Περιέχει τρεις συναρτήσεις την `__init__` που αρχικοποιεί σημαντικές παραμέτρους, την `__len__` που υπολογίζει το πλήθος εικόνων και την `__getitem__` που χειρίζεται μια εικόνα και τις απαραίτητες πληροφορίες της. Η `__init__` παίρνει ως ορίσματα ένα σύνολο δεδομένων `dataset` που είναι το αντίστοιχο `dataset` του `CIFAKEDatasetHandler` με καθαρά δεδομένα, το `encoding_map` που είναι ο χάρτης του συνόλου με τις κωδικοποιήσεις θορύβου για όλες τις εικόνες, όπως προέκυψε από την συνάρτηση `gen_encoding_map`, και ως τρίτο όρισμα το `trans_cache` που περιέχει την μνήμη cache με τις λίστες μετασχηματισμών. Η συνάρτηση αυτή αρχικοποιεί όλα αυτά τα ορίσματα.

Η συνάρτηση `__len__` καλεί την `__len__` του αντίστοιχου `dataset` και επιστρέφει το μήκος του αρχικού `dataset` το οποίο ισούται με το μήκος της βάσης που χρησιμοποιείται για δεδομένα με θόρυβο, αφού τα δύο `datasets` είναι παρόμοια με μόνη διαφορά πως στο θορυβώδες κάποιες εικόνες έχουν θόρυβο. Η συνάρτηση `__getitem__` δέχεται ως όρισμα έναν αριθμό `index` που είναι η θέση της υπό εξέταση εικόνας στο `dataset`. Λόγω της ομοιότητας

των datasets καλείται η αντίστοιχη `__getitem__` του βασικού dataset, για να αποκτηθούν οι ετικέτες της εικόνας και η ίδια η εικόνα. Στη συνέχεια, ανατίθεται στην εικόνα η κατάλληλη απεικόνιση από τον χάρτη απεικονίσεων, αποκτάται η σωστή λίστα μετασχηματισμών για την κωδικοποίηση και εφαρμόζονται οι μετασχηματισμοί στην εικόνα. Τέλος, επιστρέφονται οι ετικέτες και το μονοπάτι της εικόνας, ενώ αν υπάρξει λάθος στην μέθοδο, γίνεται επανάκληση της μεθόδου δείχνοντας στην επόμενη θέση στο σύνολο εικόνων. Η υλοποίηση του CIFAKENoisyDatasetHandler παρουσιάζεται στο σχήμα B.16.

7.2.3 Δημιουργία datasets και dataloaders

Για να δημιουργηθούν τα νέα σύνολα και τα αντίστοιχα dataloaders, αρχικά δημιουργήθηκαν οι αντίστοιχοι CIFAKEDatasetHandler που χειρίζονται τα αντίστοιχα καθαρά σύνολα. Έπειτα, για κάθε μία από τις 4 περιπτώσεις δημιουργήθηκε ξεχωριστός χάρτης με κωδικοποιήσεις και μνήμες cache. Όλα αυτά χρησιμοποιήθηκαν κατόπιν για την ανάπτυξη των CIFAKENoisyDatasetHandler. Στα σχήματα B.17, B.18, B.19 και B.20 παρουσιάζεται η δημιουργία των διαφορετικών συνόλων με θόρυβο.

7.3 Εκπαίδευση και έλεγχος μοντέλων

Εκπαιδεύτηκαν τέσσερα νέα μοντέλα ResNet50Classifier για την διερεύνηση της αντοχής του μοντέλου σε θόρυβο. Τα μοντέλα αυτά ονομάστηκαν:

1. `full_noise_model` που εκπαιδεύτηκε σε εικόνες με όλα τα είδη θορύβου και την παρουσία και των δύο ειδών θορύβου σε εικόνες ταυτόχρονα,
2. `both_noise_model` που εκπαιδεύτηκε σε εικόνες με όλα τα είδη θορύβου αλλά κάθε εικόνα είχε το πολύ ένα από τα δύο είδη θορύβου,
3. `jpeg_model` που εκπαιδεύτηκε σε εικόνες με JPEG Compression,
4. `gaussian_model` που εκπαιδεύτηκε σε εικόνες με Γκαουσιανό θόρυβο και
5. `clean_model` που είναι το μοντέλο του προηγούμενου πειράματος το οποίο εκπαιδεύτηκε σε καθαρά δεδομένα.

Είναι σημαντικό να αναφερθεί πως το `clean_model` δεν εκπαιδεύτηκε εκ νέου, αλλά χρησιμοποιείται για συγκριτική αξιολόγηση. Όλα τα μοντέλα ακολούθησαν τους αλγορίθμους

εκπαίδευσης και ελέγχου, όπως αναφέρονται στους αλγορίθμους 3 και 4 χωρίς καμία αλλαγή πέρα από το σύνολο δεδομένων που χρησιμοποιήθηκε. Επίσης, το σύνολο ελέγχου για την εκπαίδευση κάθε μοντέλου ακολουθούσε την ίδια κατανομή με αυτή του αντίστοιχου συνόλου εκπαίδευσης, για να εξασφαλιστεί η σωστή εκπαίδευση του μοντέλου. Τέλος, όπως και στην ενότητα 5.2.2, συλλέχθηκαν διαγράμματα απωλειών και ακριβείας για εκπαίδευση και έλεγχο και δημιουργήθηκαν και τρία επιπλέον διαγράμματα που απεικονίζουν τις απώλειες και τις ακρίβειες των μοντέλων μαζί.

7.3.1 Εκπαίδευση και έλεγχος του *full_noise_model*

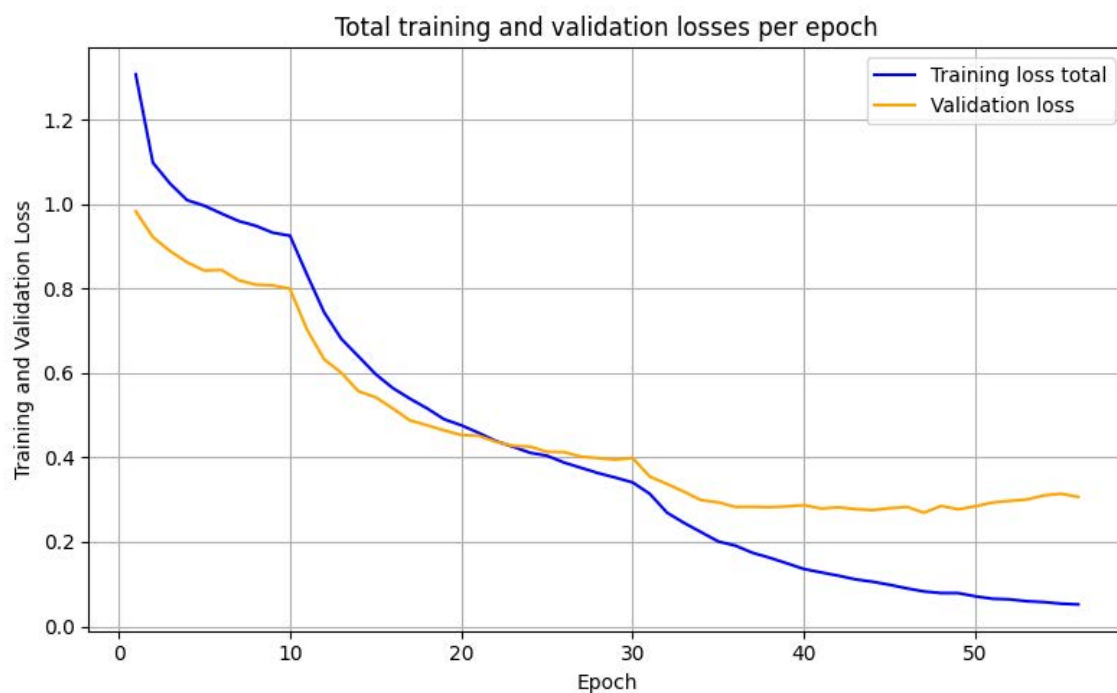
Στην ενότητα αυτή εξετάζονται τα διαγράμματα της εκδοχής *full_noise_model*. Με βάση τα σχήματα 7.2, 7.3 και 7.4 το μοντέλο αυτό εκπαιδεύτηκε για 56 εποχές, πετυχαίνοντας την μέγιστη κοινή ακρίβεια ελέγχου 92.7% την εποχή 51. Παρόλο που το μοντέλο εκπαιδεύτηκε για σημαντικά μικρότερο αριθμό εποχών από το *clean_model*, δεν παρατηρούνται σημαντικές διαφορές στις μετρικές της καλύτερης εποχής σε σύγκριση με τις αντίστοιχες του *clean_model*. Μάλιστα, η ικανότητα μάθησης του μοντέλου αυτού είναι θεαματική, καθώς, παρόλο που ο βαθμός δυσκολίας ταξινόμησης των δειγμάτων εκπαίδευσης και ελέγχου είναι μεγαλύτερος, το μοντέλο κατάφερε να πετύχει συγκρίσιμη ακρίβεια με αυτή του μοντέλου του πρώτου πειράματος.

7.3.2 Εκπαίδευση και έλεγχος του *both_noise_model*

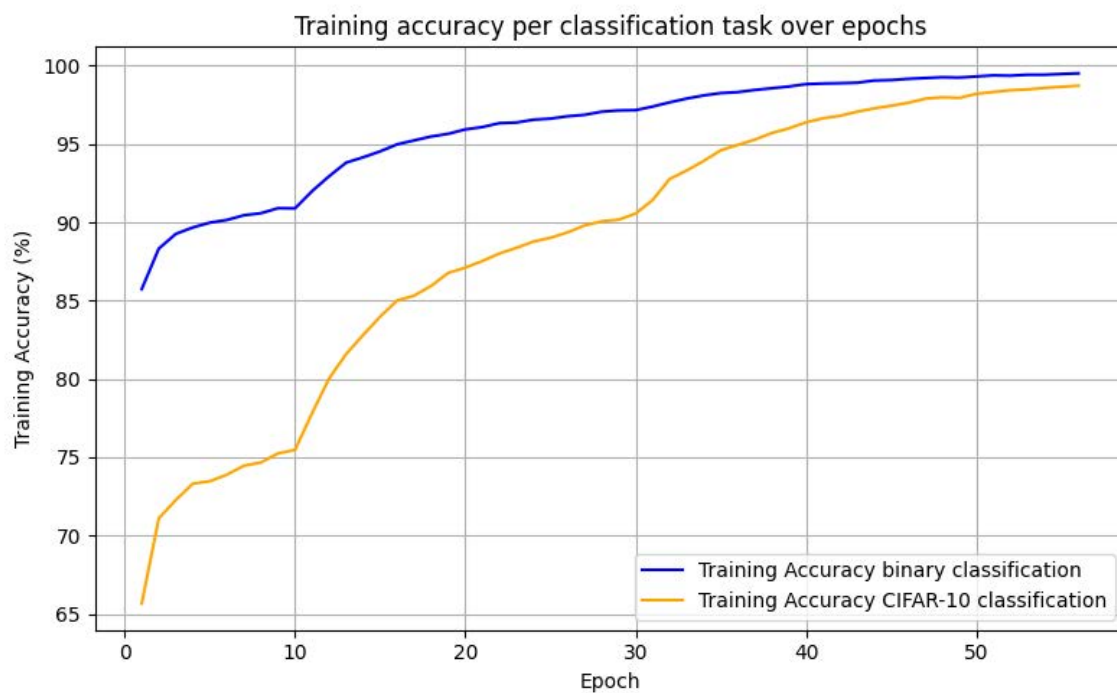
Στην ενότητα αυτή εξετάζονται τα διαγράμματα του μοντέλου *both_noise_model* (σχήματα 7.5, 7.6 και 7.7). Το μοντέλο αυτό εκπαιδεύτηκε για 63 εποχές και σημείωσε την καλύτερη επίδοση κοινής ακριβείας ελέγχου την εποχή 58 με ποσοστό 93.06%. Όπως και με τα άλλα δύο μοντέλα, έτσι και αυτό σημείωσε παρόμοια αποτελέσματα σε όλες τις μετρικές. Μάλιστα, σημείωσε μικρή βελτίωση στα αποτελέσματα σε σχέση με το *full_noise_model*, γεγονός που πιθανότατα οφείλεται στον μικρότερο βαθμό δυσκολίας των σετ.

7.3.3 Εκπαίδευση και έλεγχος του *jpeg_model*

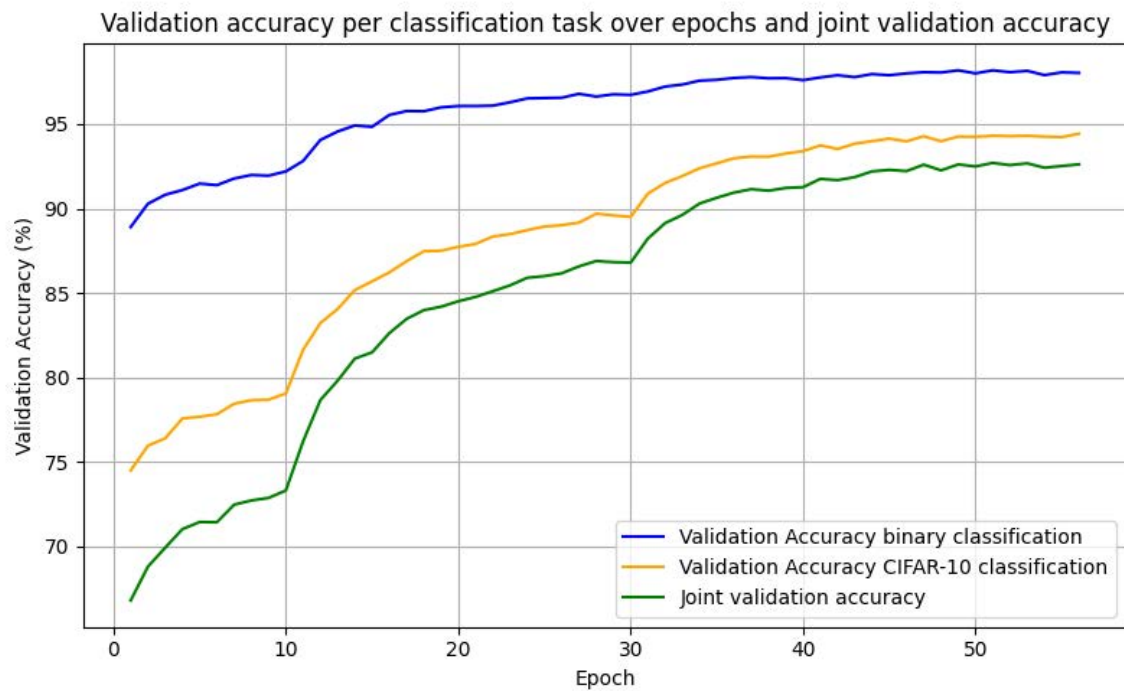
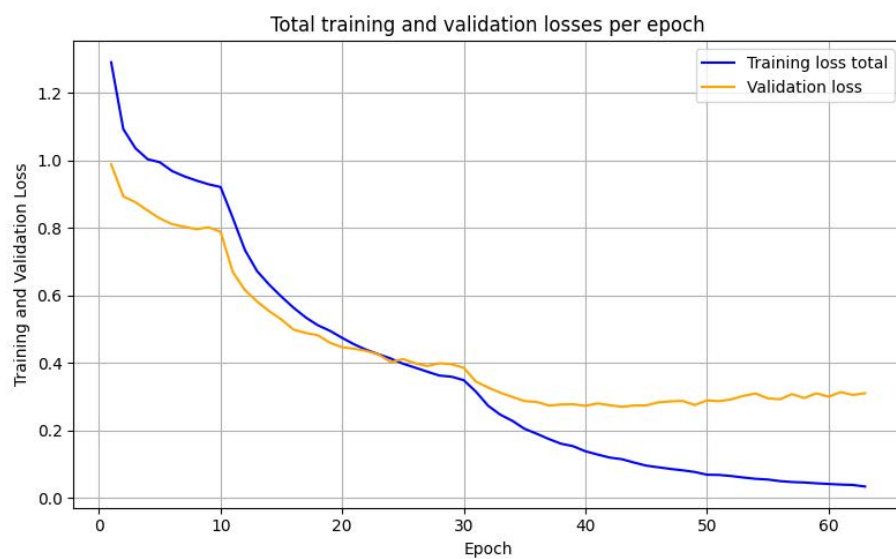
Στην ενότητα αυτή εξετάζονται τα διαγράμματα του μοντέλου *jpeg_model* (σχήματα 7.8, 7.9 και 7.10). Το μοντέλο εκπαιδεύτηκε για 67 εποχές και σημείωσε την καλύτερη επίδοση κοινής ακριβείας ελέγχου με 93.31% την εποχή 62. Αυτή είναι η δεύτερη καλύτερη τιμή

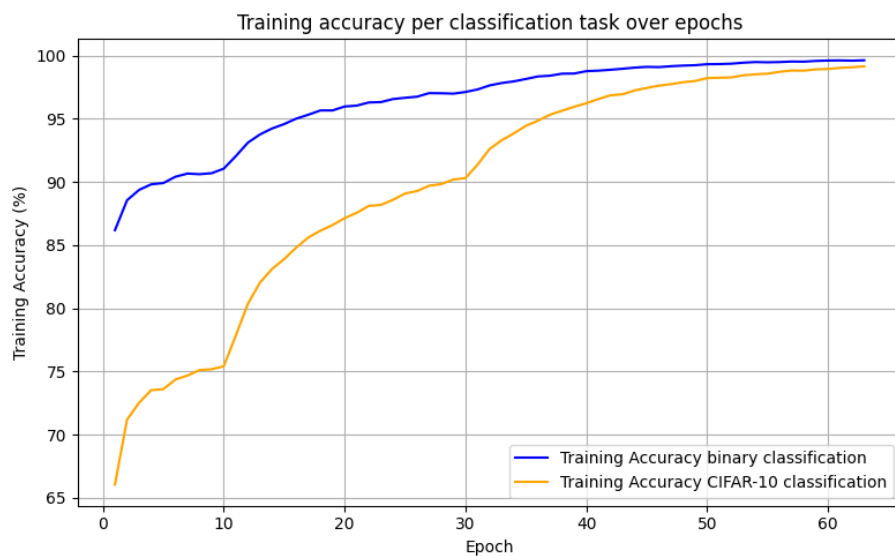
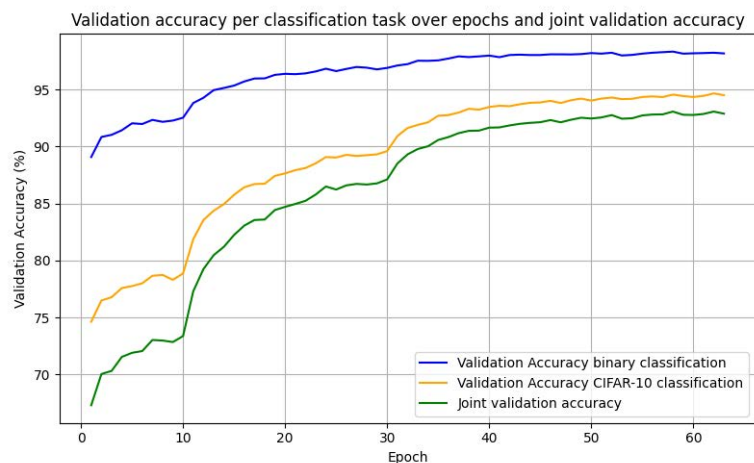


Σχήμα 7.2: Οι απώλειες του full_noise_model

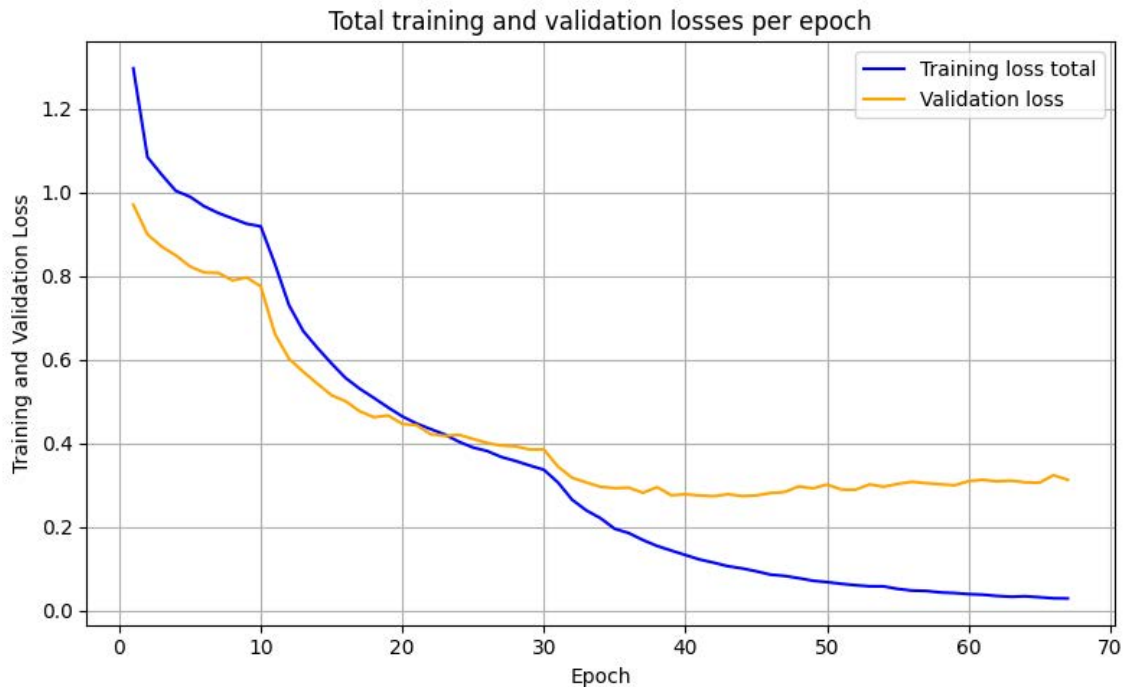


Σχήμα 7.3: Οι ακρίβειες εκπαίδευσης του full_noise_model

Σχήμα 7.4: Οι ακρίβειες ελέγχου του *full_noise_model*Σχήμα 7.5: Οι απώλειες του *both_noise_model*

Σχήμα 7.6: Οι ακρίβειες εκπαίδευσης του `both_noise_model`Σχήμα 7.7: Οι ακρίβειες ελέγχου του `both_noise_model`

μεταξύ των τεσσάρων μοντέλων που έχουν εξεταστεί μέχρι στιγμής και είναι ελάχιστα χειρότερη από την κοινή ακρίβεια ελέγχου του *clean_model*. Όλες οι μετρικές και σε αυτό το μοντέλο ακολουθούν παρόμοια πορεία εξέλιξης με τα προηγούμενα μοντέλα.



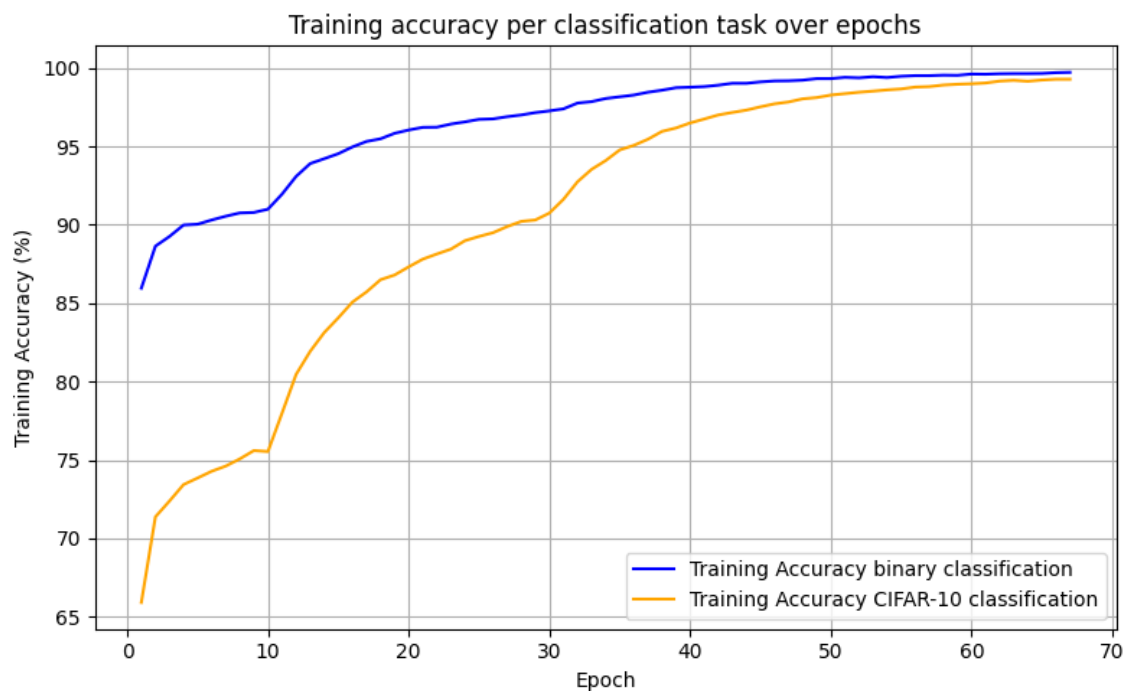
Σχήμα 7.8: Οι απώλειες του *jpeg_model*

7.3.4 Εκπαίδευση και έλεγχος του *gaussian_model*

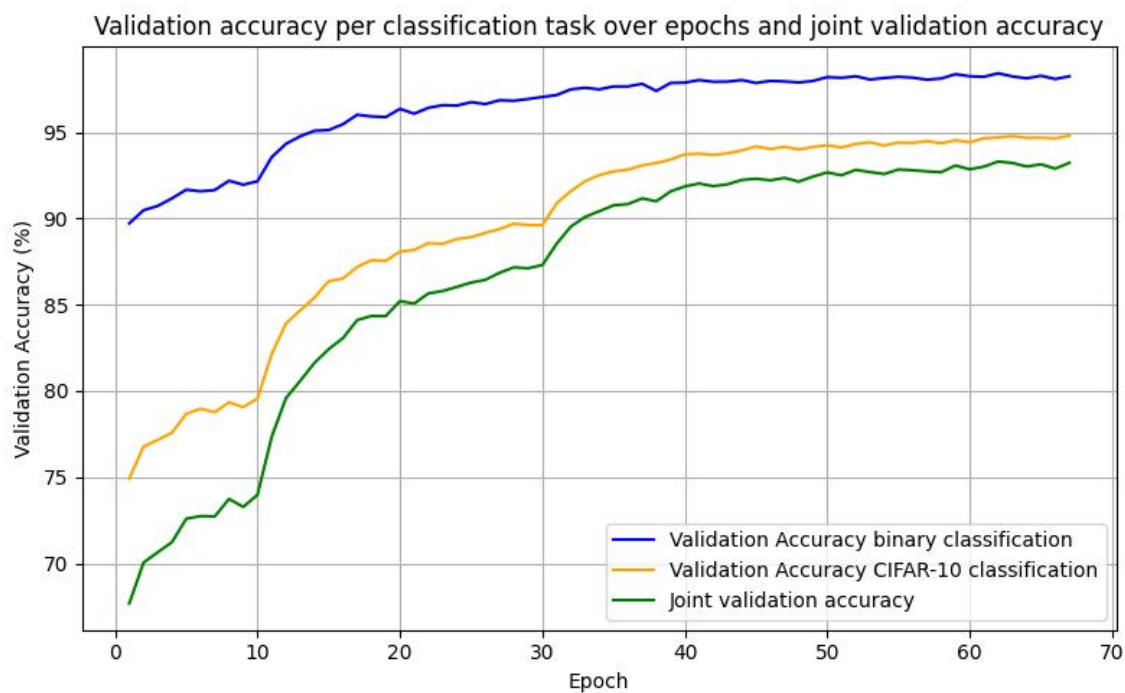
Στην ενότητα αυτή εξετάζονται τα διαγράμματα του μοντέλου *gaussian_model* (σχήματα 7.11, 7.12 και 7.13). Το μοντέλο αυτό εκπαιδεύτηκε για 74 εποχές και σημείωσε την καλύτερη κοινή ακρίβεια ελέγχου με 93.55% την εποχή 69. Αυτή η επίδοση είναι και η καλύτερη συγκριτικά με όλα τα μοντέλα περιλαμβανομένου και του *clean_model*. Οι υπόλοιπες μετρικές επίσης ακολουθούν παρόμοια πορεία με όλα τα υπόλοιπα μοντέλα.

7.3.5 Συνολική συγκριτική αξιολόγηση εκπαίδευσης και ελέγχου

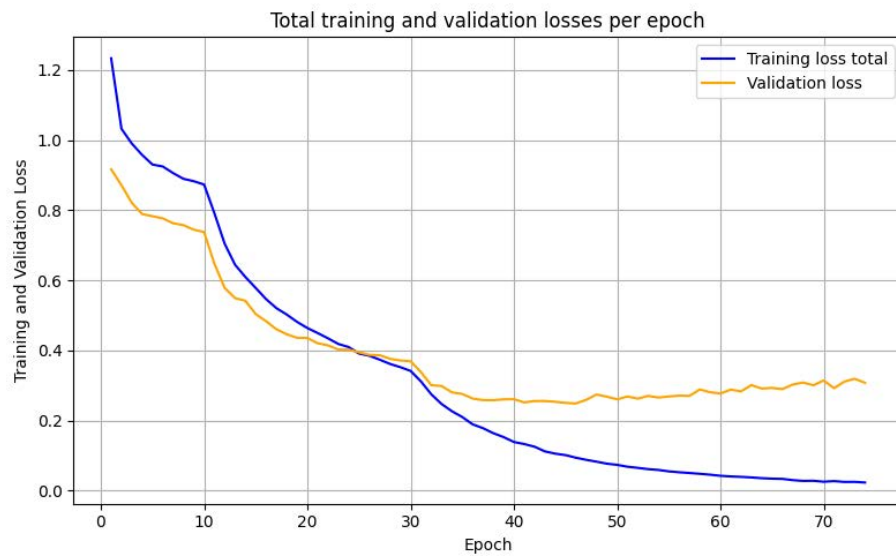
Από τα σχήματα 7.14, 7.15 και 7.16 φαίνεται πως όλα τα μοντέλα συγκλίνουν στις ίδιες τιμές για κάθε μετρική. Μάλιστα, η πορεία εξέλιξης είναι παρόμοια σε όλες τις μετρικές σε όλα τα μοντέλα και όσο περνούν οι εποχές τόσο πιο πολύ συγκλίνουν οι τιμές των μοντέλων. Από τα προαναφερθέντα σχήματα φαίνεται ότι η παρουσία θορύβου στα σύνολα εκπαίδευσης και ελέγχου δεν διαφοροποιεί σημαντικά την πορεία εκπαίδευσης ενός *ResNet50Classifier*.



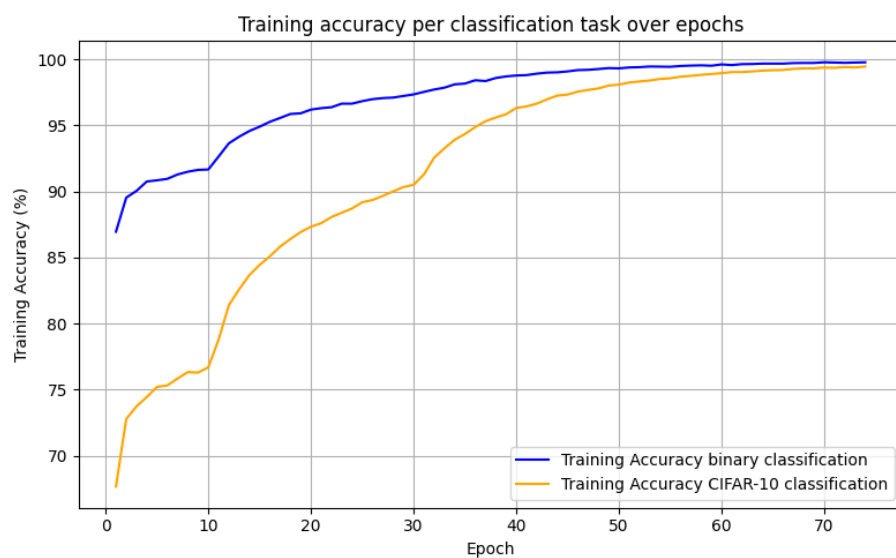
Σχήμα 7.9: Οι ακρίβειες εκπαίδευσης του jpeg_model



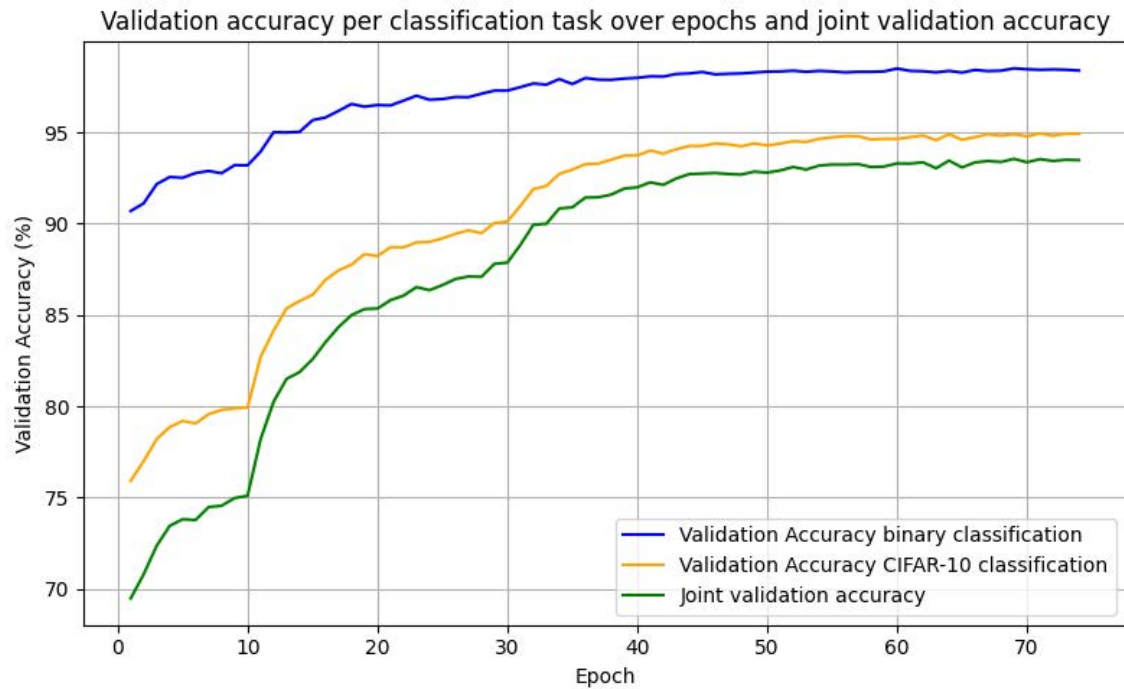
Σχήμα 7.10: Οι ακρίβειες ελέγχου του jpeg_model



Σχήμα 7.11: Οι απώλειες του gaussian_model

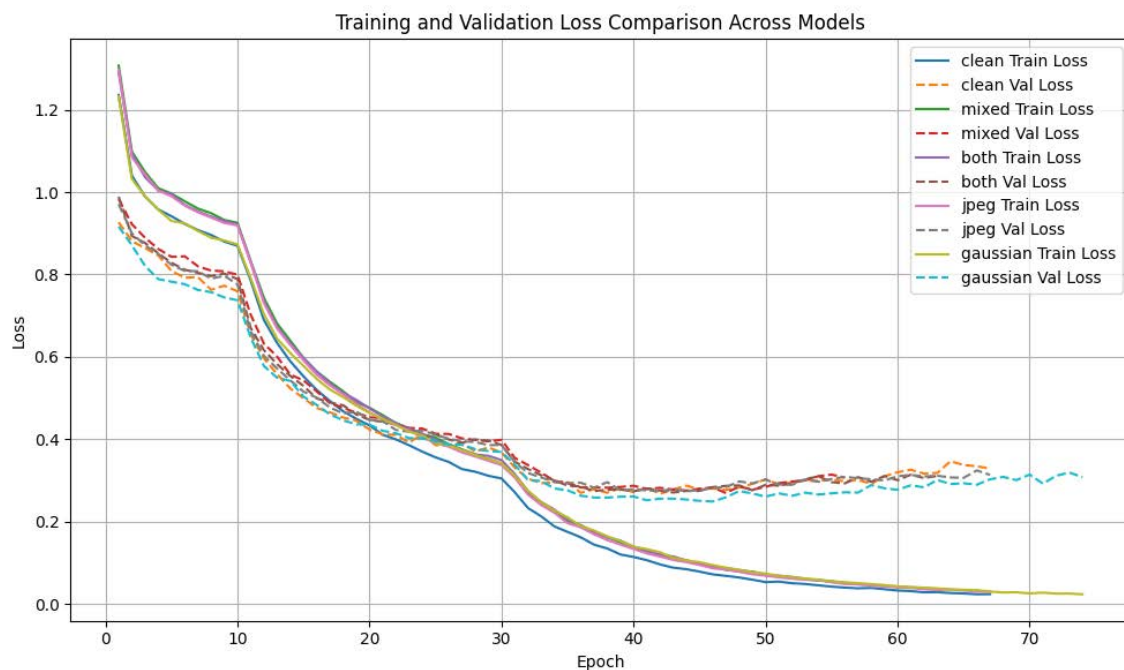


Σχήμα 7.12: Οι ακρίβειες εκπαίδευσης του gaussian_model

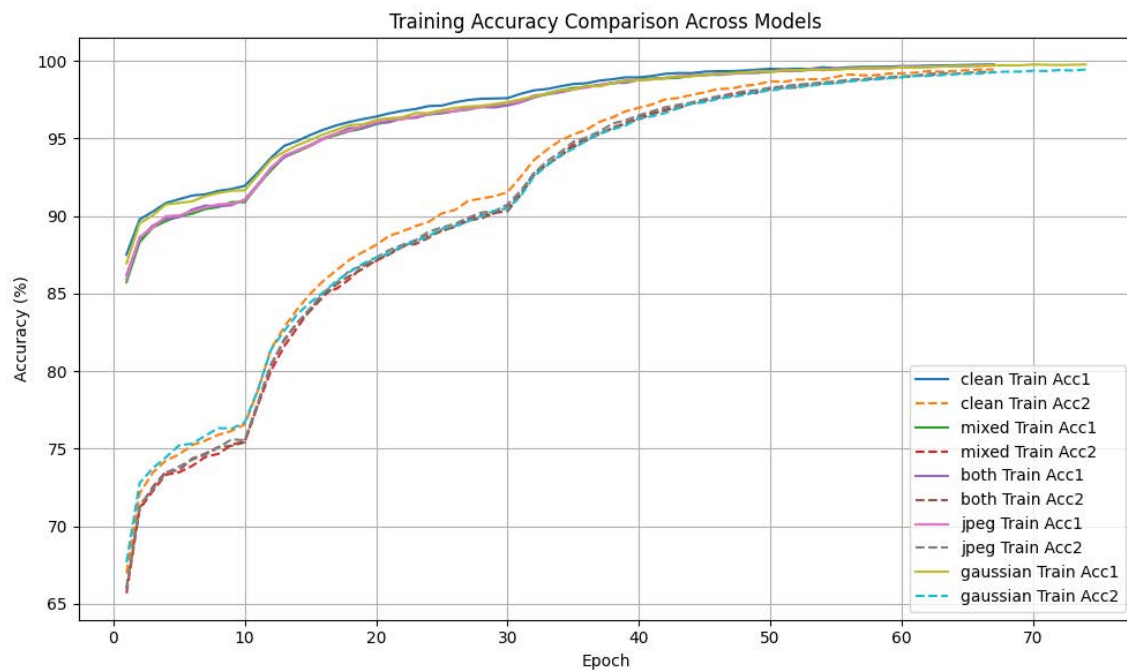


Σχήμα 7.13: Οι ακρίβειες ελέγχου του gaussian_model

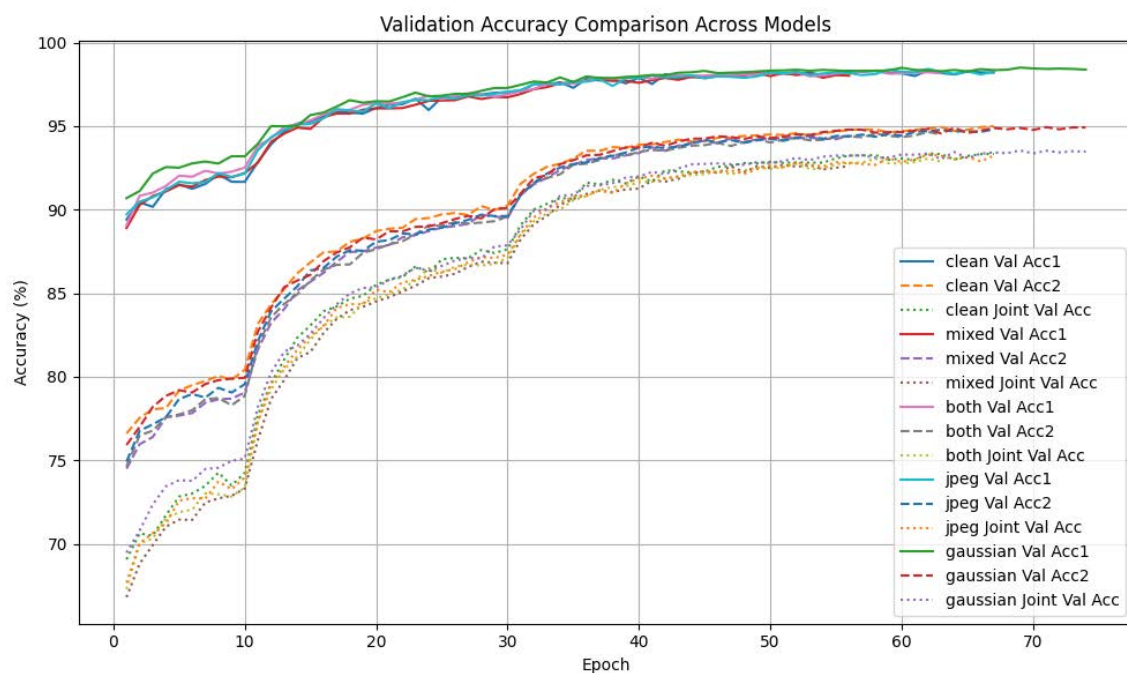
Αυτά τα στοιχεία αποτελούν πιθανώς πρόωγη ένδειξη πως το μοντέλο είναι όντως ανθεκτικό στον θόρυβο και θα μπορούσε να χρησιμοποιηθεί σε πραγματικές συνθήκες, όμως, για να εξαχθεί τελικό συμπέρασμα, πρέπει να συγκριθούν τα αποτελέσματα των δοκιμών.



Σχήμα 7.14: Οι απώλειες όλων των μοντέλων



Σχήμα 7.15: Οι ακρίβειες εκπαίδευσης όλων των μοντέλων



Σχήμα 7.16: Οι ακρίβειες ελέγχου όλων των μοντέλων

7.4 Ανάλυση δοκιμής των μοντέλων

Μετά την εκπαίδευση και τον έλεγχο των μοντέλων ακολουθεί η δοκιμή. Κάθε μοντέλο δοκιμάστηκε και στα πέντε σύνολα δοκιμής που δημιουργήθηκαν. Τα σύνολα δοκιμής ονομάστηκαν “clean”, “full”, “both”, “jpeg”, “gaussian” και αφορούν αντίστοιχα το καθαρό σύνολο χωρίς θόρυβο, το σύνολο με όλα τα είδη θορύβου στο οποίο οι εικόνες μπορούν να έχουν και τα δύο είδη ταυτόχρονα παρόντα, το σύνολο με όλα τα είδη θορύβου στο οποίο οι εικόνες έχουν μέχρι ένα είδος, το σύνολο με JPEG Compression και το σύνολο με Γκαουσιανό θόρυβο. Ο αλγόριθμος δοκιμής είναι ίδιος με τον αλγόριθμο 5. Από κάθε σενάριο δοκιμής (25 συνολικά) αποκτήθηκαν οι ίδιες μετρικές που αποκτήθηκαν και για το πρώτο πείραμα.

7.4.1 Δοκιμή clean_model και γραφήματα

Στην ενότητα αυτή παρουσιάζονται τα αποτελέσματα της δοκιμής του μοντέλου clean_model, εκτός της δοκιμής του σε καθαρό σύνολο δεδομένων, διότι παρουσιάστηκε στο κεφάλαιο 6. Αναλυτικά τα αποτελέσματα για το μοντέλο βρίσκονται στον πίνακα A.1 και τα περισσότερα σχήματα των γραφικών παραστάσεων παρουσιάζονται στην ενότητα A.2.1.

Το clean_model στο “full” dataset έχει κοινή ακρίβεια 86.09%, που είναι κατά 7 μονάδες χαμηλότερη από την κοινή ακρίβεια, όταν εξετάζεται σε καθαρό (“clean”) σύνολο δεδομένων. Το κοινό precision είναι 87.49%, το κοινό recall 86.09% και το κοινό f1 score 86.45%. Γενικότερα, όλες οι κοινές μετρικές είναι αρκετά χαμηλότερες από τις αντίστοιχες της δοκιμής με “clean”. Αυτό το αποτέλεσμα είναι αναμενόμενο, καθώς το μοντέλο εκπαιδεύτηκε σε δεδομένα χωρίς θόρυβο, επομένως δυσκολεύεται να ταξινομήσει κατάλληλα τις κλάσεις. Επίσης, οι μέσοι όροι των μετρικών της πρώτης κεφαλής έχουν εύρος από 95.76% έως 95.77%, ενώ της δεύτερης δεν αγγίζουν το 90%. Εξαίρεση αποτελεί ο μέσος όρος του precision για την δεύτερη κεφαλή που αγγίζει το 90.67%. Παρόλα αυτά, οι μετρικές και για τις 2 κεφαλές είναι μειωμένες σημαντικά. Τέλος, όπως και με το clean dataset, το μοντέλο δυσκολεύεται περισσότερο να ταξινομήσει τις αληθινές εικόνες, όμως σε αυτή την περίπτωση η δυσκολία είναι εμφανέστερη, ειδικά στην κλάση frog, όπου ταξινομεί σωστά το 73.6% των αληθινών εικόνων και αντίστοιχα το 97.5% των συνθετικών εικόνων (σχήμα 7.17.)

Στη περίπτωση “both” το μοντέλο επιτυγχάνει κοινή ακρίβεια 84.91%, κοινό precision 86.67%, κοινό recall 84.91% και κοινό f1 score 85.35% (σχήμα 7.18). Για άλλη μια φορά

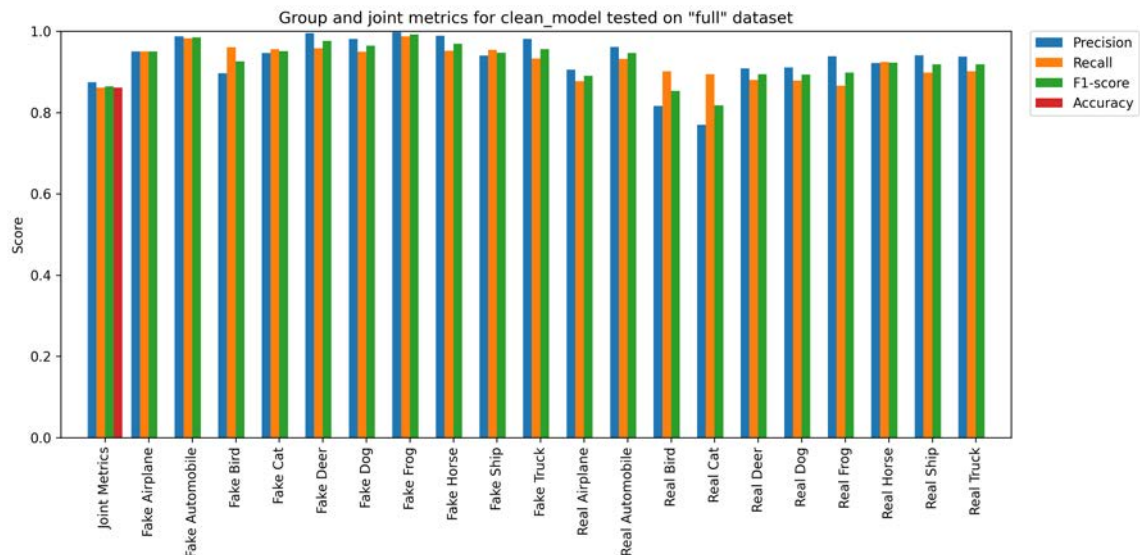
το *clean_model* σημειώνει μικρότερη επίδοση σε σύνολο δοκιμής διαφορετικό από σύνολο εκπαίδευσης. Στην κεφαλή 1 επιτυγχάνει μέση ακρίβεια, precision, recall, f1 score εύρους 95.37% έως 95.38% και στην κεφαλή 2 σημειώνει μέση ακρίβεια και recall 88.69%, μέσο precision 89.99% και μέσο f1 score 88.97%. Τέλος, επιβεβαιώνεται για άλλη μια φορά πως οι αληθινές εικόνες το δυσκολεύουν περισσότερο σε σχέση με τις συνθετικές για ταξινόμηση.

Στο dataset “jpeg” η επίδοση του μοντέλου είναι σχετικά βελτιωμένη, αφού επιτυγχάνει 88.11% κοινή ακρίβεια και όλες οι κοινές μετρικές κυμαίνονται από 88.11%-89.01% (σχήμα 7.19), παρόλα αυτά είναι αρκετά χαμηλότερη συγκριτικά με το *clean dataset*. Η ίδια παρατήρηση σχετικά με την ελαφρώς βελτιωμένη επίδοση εξάγεται και από τις μετρικές για τις κεφαλές του μοντέλου. Η μέση ακρίβεια για την κεφαλή 1 είναι 96.56% και για την κεφαλή 2 90.99%, τιμές εμφανώς βελτιωμένες συγκριτικά με τα datasets “full” και “both”. Η καλύτερη αυτή επίδοση οφείλεται στην μεγαλύτερη ευκολία του dataset, καθώς ο μετασχηματισμός JPEG Compression με ποιότητα $q = 80$ παράγει εικόνες που δεν έχουν χάσει σημαντικές πληροφορίες. Τέλος, παρατηρείται ξανά η μεγαλύτερη δυσκολία στις επιμέρους κλάσεις των αληθινών εικόνων.

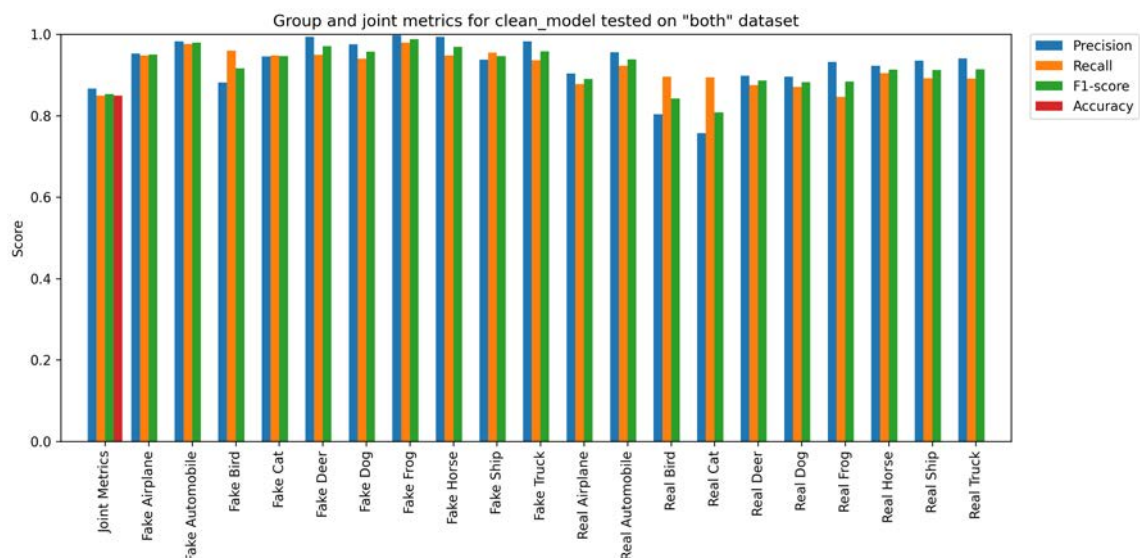
Το μοντέλο σημειώνει τη χειρότερη του επίδοση στο σενάριο δοκιμής “gaussian”, διότι πετυχαίνει κοινή ακρίβεια 82.66% και οι υπόλοιπες κοινές μετρικές κυμαίνονται σε εξίσου χαμηλά επίπεδα (σχήμα 7.20). Η μέση ακρίβεια της κεφαλής 1 είναι κάτω του 95%, συγκεκριμένα 94.78%, ενώ της κεφαλής 2 είναι μόλις 86.89%. Ενώ χρησιμοποιείται μόνο ένα είδος θορύβου και ο βαθμός δυσκολίας σε σχέση με τα datasets “full” και “both” θα έπρεπε να είναι θεωρητικά χαμηλότερος, παρατηρείται το αντίθετο. Αυτό οφείλεται στην ίδια την φύση του θορύβου, αφού η τυχαία εφαρμογή Gaussian noise στο 10% της εικόνας, μπορεί να οδηγήσει σε μικρή αλλά σημαντική απώλεια λεπτομέρειας, δυσχεραίνοντας έτσι την δυνατότητα του μοντέλου να ταξινομεί σωστά. Στα datasets “full” και “both”, ενώ υπάρχουν εικόνες με Gaussian noise, η κατανομή τους είναι διαφορετική, καθώς στο πρώτο dataset αυτός ο μετασχηματισμός υπάρχει περίπου στο 33.2% των εικόνων, στο δεύτερο στο 25% και στο υπό εξέταση σύνολο στο 50%. Αυτό σημαίνει ότι σε αυτή την περίπτωση υπάρχουν περίπου 3400 εικόνες περισσότερες με Γκαουσιανό θόρυβο, γεγονός που επηρεάζει σοβαρά την αποτελεσματικότητα του μοντέλου. Τέλος, παρατηρείται πως η ταξινόμηση αληθινών εικόνων στις κλάσεις του CIFAR-10 [6] είναι πιο δύσκολη από την ταξινόμηση των αντίστοιχων συνθετικών εικόνων.

Από το διάγραμμα του σχήματος 7.21 και από τις προηγούμενες παρατηρήσεις είναι προ-

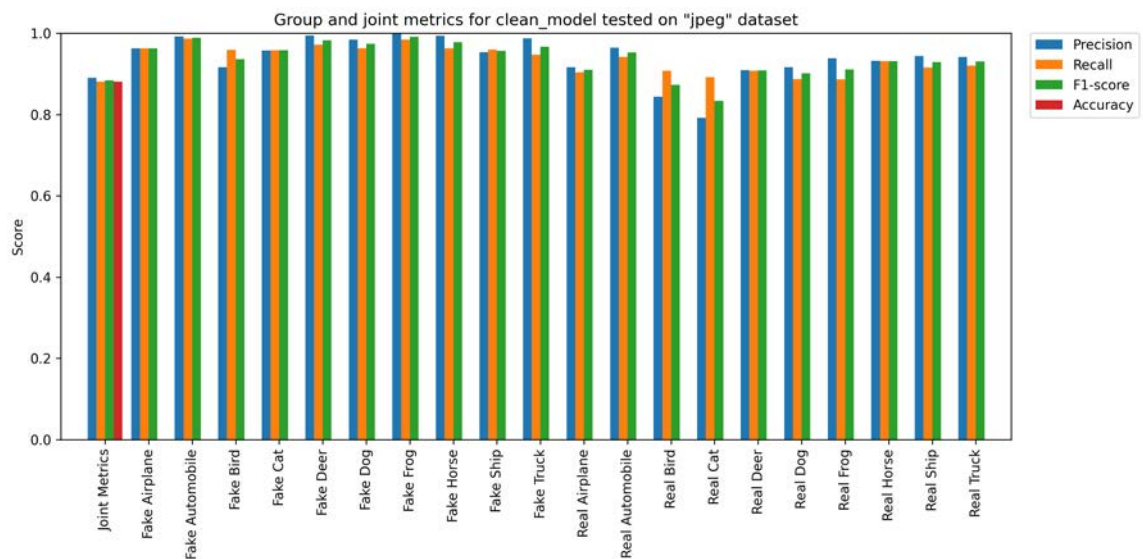
φανές πως το μοντέλο `clean_model` λειτουργεί βέλτιστα σε εικόνες χωρίς θόρυβο. Όταν αντιμετωπίζει εικόνες με θόρυβο, η απόδοσή του μειώνεται δραστικά, γεγονός αναμενόμενο καθώς δεν έχει εκπαιδευτεί σε αντίστοιχα παραδείγματα.



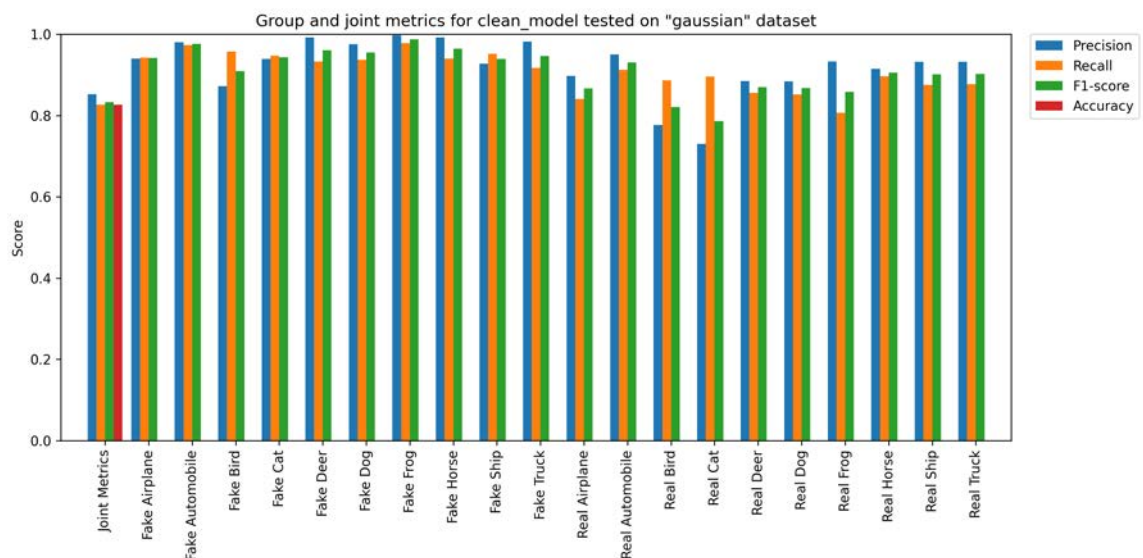
Σχήμα 7.17: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο “full”



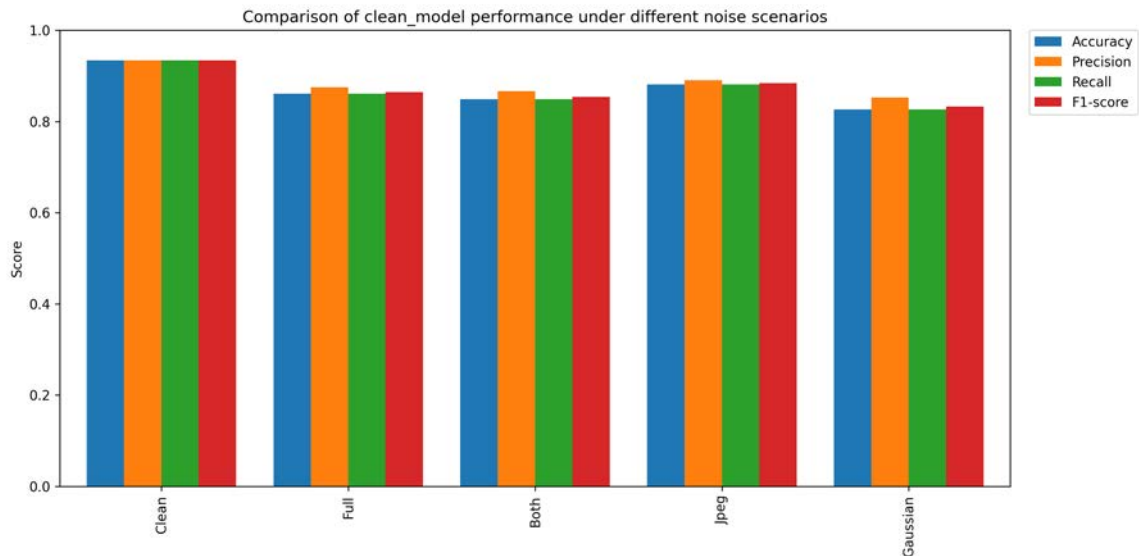
Σχήμα 7.18: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο “both”



Σχήμα 7.19: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με θόρυβο "jpeg"



Σχήμα 7.20: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset με Gaussian noise



Σχήμα 7.21: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων του μοντέλου για κάθε περίπτωση

7.4.2 Ανάλυση δοκιμής μοντέλου `full_noise_model` υπό διαφορετικές συνθήκες θορύβου

Όπως και το `clean_model` έτσι και το `full_noise_model` δοκιμάστηκε και στα 5 διαφορετικά σύνολα θορύβου. Ακολουθεί η ανάλυση της επίδοσής του. Τα αποτελέσματα βρίσκονται αναλυτικά στον πίνακα A.2 και τα περισσότερα σχήματα παρουσιάζονται στην ενότητα A.2.2.

Αρχικά, το μοντέλο κατέγραψε υψηλή επίδοση στο “clean” dataset σημειώνοντας κοινή ακρίβεια 92.05%. Οι υπόλοιπες κοινές μετρικές κυμαίνονται σε παρόμοια ποσοστά (σχήμα 7.22). Στην κεφαλή 1 οι μέσοι όροι όλων των μετρικών είχαν εύρος τιμών από 97.79% έως 97.8% και στην κεφαλή 2 αγγίζουν περίπου το 94.05%. Οι επιδόσεις αυτές είναι πολύ ικανοποιητικές δεδομένου ότι το μοντέλο εκπαιδεύτηκε σε δεδομένα με θόρυβο αλλά δοκιμάστηκε σε καθαρά δεδομένα. Επίσης, παρατηρήθηκε δυσκολία ταξινόμησης αληθινών εικόνων στις κλάσεις CIFAR-10 [6], αλλά ήταν λιγότερο εμφανής συγκριτικά με τα αποτελέσματα του `clean_model` σε datasets διαφορετικά του “clean”.

Στο σενάριο “full” το μοντέλο πέτυχε ακόμη καλύτερα αποτελέσματα σημειώνοντας περίπου 92.6% στις κοινές μετρικές (σχήμα 7.23). Στην κεφαλή 1 οι μέσοι όροι των μετρικών κυμαίνονται στο 98.12% και στην κεφαλή 2 στο 94.27%. Τα καλύτερα αποτελέσματα συγκριτικά με το clean dataset οφείλονται στο ότι το μοντέλο εκπαιδεύτηκε στο αντίστοιχο full

training dataset οπότε εκπαιδεύτηκε σε παρόμοια κατανομή δεδομένων.

Στο σενάριο “both” καταγράφηκε ελάχιστα χειρότερη επίδοση συγκριτικά με εκείνη του προηγούμενου. Συγκεκριμένα, οι κοινές μετρικές κυμαίνονται στο 92.46%-92.49% (σχήμα 7.24), για την κεφαλή 1 στο 98.19% και για την κεφαλή 2 στο 94.1%. Το μοντέλο παρότι εκπαιδεύτηκε σε διαφορετικά δεδομένα επιδεικνύει μέχρι στιγμής αντοχή στο θόρυβο.

Το μοντέλο και στο σενάριο “jreg” επιδεικνύει σωστή συμπεριφορά σημειώνοντας ποσοστά της τάξεως του 92.5% στις κοινές μετρικές (σχήμα 7.25), 98.23% στους μέσους όρους των μετρικών της κεφαλής 1 και 94.04%-94.07% στους μέσους όρους των μετρικών της κεφαλής 2.

Στο dataset “gaussian” καταγράφονται τα καλύτερα αποτελέσματα για το *full_noise_model*. Οι κοινές μετρικές αγγίζουν το 92.82%-92.85% (σχήμα 7.26), οι μέσοι όροι μετρικών για την κεφαλή 1 και την κεφαλή 2 τα 98.3% και 94.37%-94.38%, αντίστοιχα. Σε αντίθεση με το μοντέλο *clean_model*, αυτό παρουσιάζει δραστικά καλύτερη συμπεριφορά που αποδίδεται στο διαφορετικό σύνολο εκπαίδευσης. Η χρήση όλων των ειδών θορύβου επέτρεψε στο μοντέλο να αναπτύξει ισχυρότερη ικανότητα γενίκευσης σε εικόνες με θόρυβο και να μην επηρεάζεται από την κατανομή των διαφορετικών ειδών θορύβου σε κάθε dataset. Τέλος, ακόμη και στο *full_noise_model* είναι εμφανής η μεγαλύτερη δυσκολία ταξινόμησης αληθινών εικόνων σε σχέση με την ταξινόμηση των αντίστοιχων συνθετικών εικόνων.

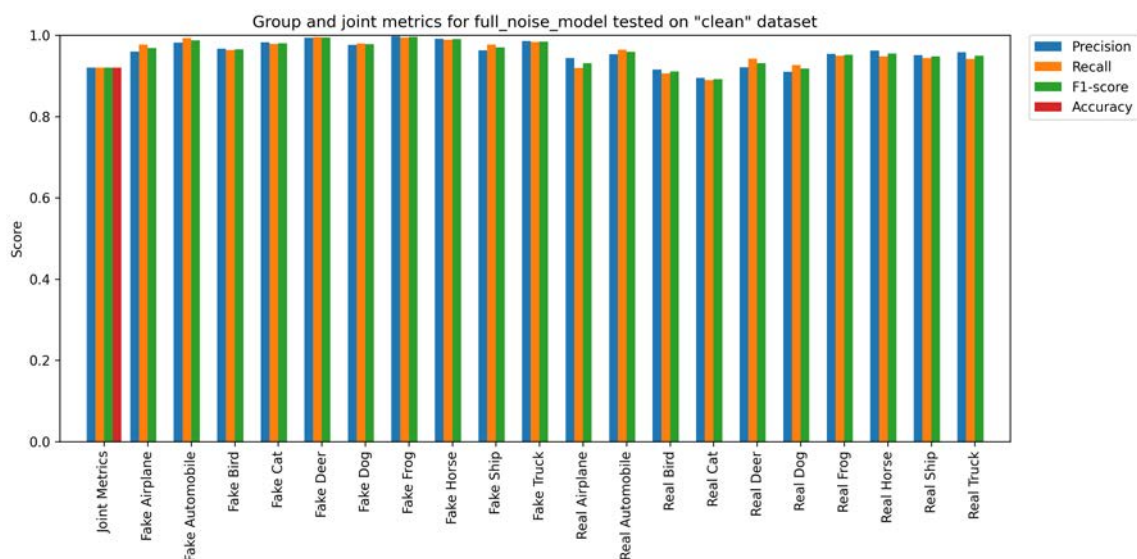
Στο μοντέλο αυτό παρατηρείται πως η απόδοσή του είναι σχετικά σταθερή ανεξαρτήτως θορύβου (σχήμα 7.27), αποδεικνύοντας έτσι την ισχυρή δυνατότητά του να ταξινομεί σωστά ακόμη και αλλοιωμένες εικόνες, αρκεί να εκπαιδευτεί καταλλήλως.

7.4.3 Ανάλυση δοκιμής μοντέλου *both_noise_model*

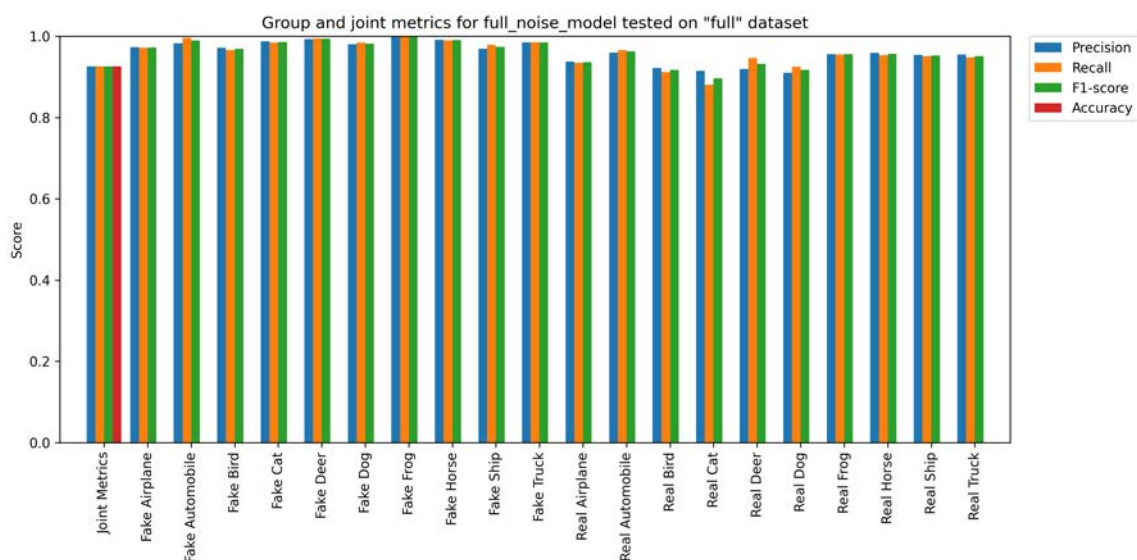
Το μοντέλο *both_noise_model* δοκιμάστηκε στα ίδια datasets με τα προηγούμενα. Τα αποτελέσματα των πειραμάτων καταγράφονται στον πίνακα A.3 και τα περισσότερα σχήματα βρίσκονται στην ενότητα A.2.3. Ακολουθεί η αξιολόγησή του.

Το μοντέλο σημειώνει 91.7% στις κοινές μετρικές (σχήμα 7.28), στους μέσους όρους των μετρικών της κεφαλής 1 και κεφαλής 2 επιτυγχάνει 97.5% και 93.9% αντίστοιχα, στη “clean” ΒΔ.

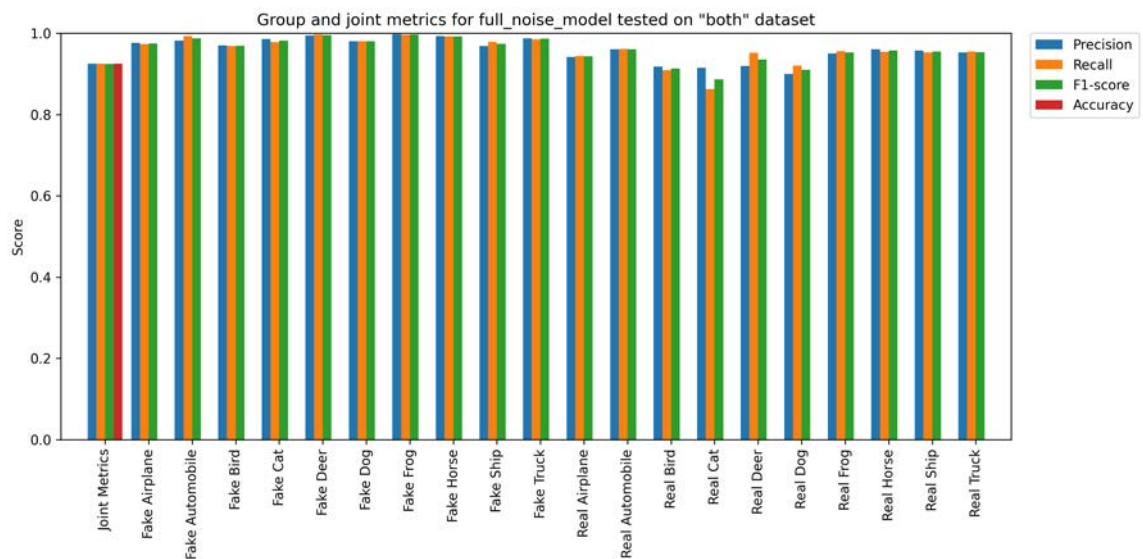
Στο dataset “full” το μοντέλο σημειώνει 92.8% στις κοινές μετρικές (σχήμα 7.29), 98.2% στους μέσους όρους των μετρικών της κεφαλής 1 και 94.4% στους μέσους όρους των μετρικών της κεφαλής 2. Η επίδοσή του στη ΒΔ “full” είναι ελαφρώς βελτιωμένη σε σχέση με



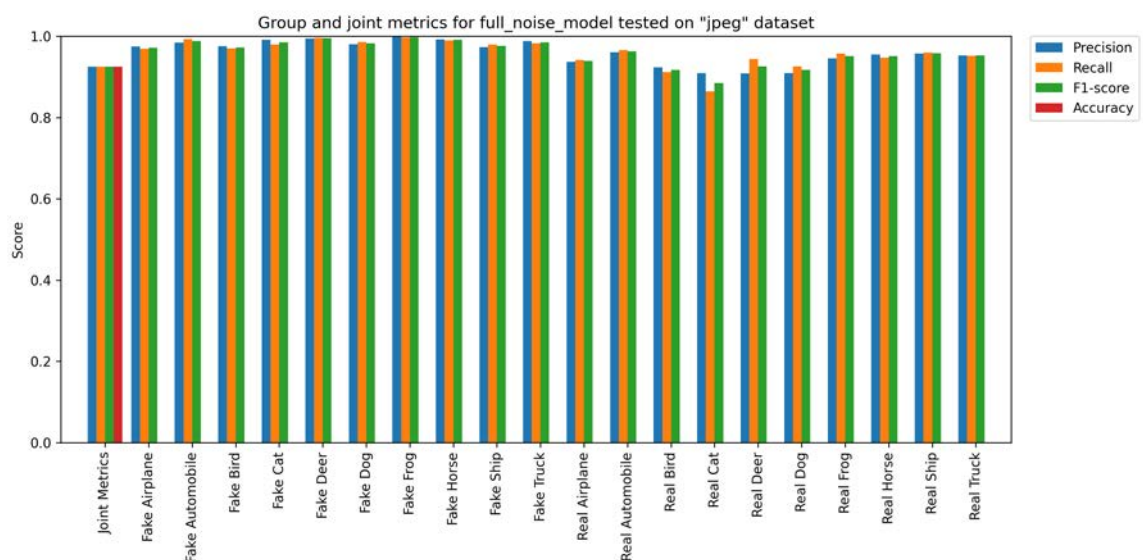
Σχήμα 7.22: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”



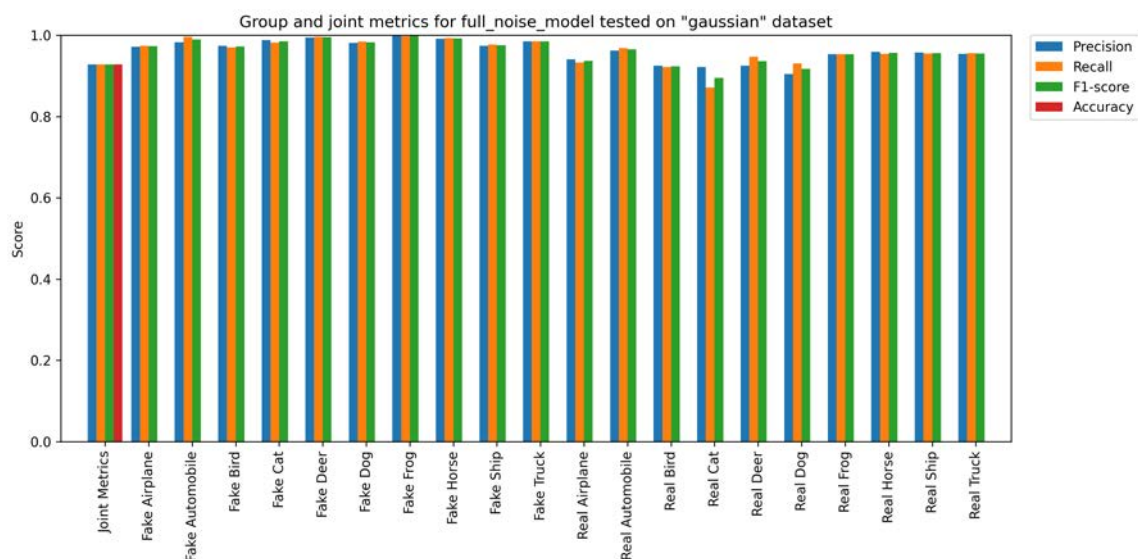
Σχήμα 7.23: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”



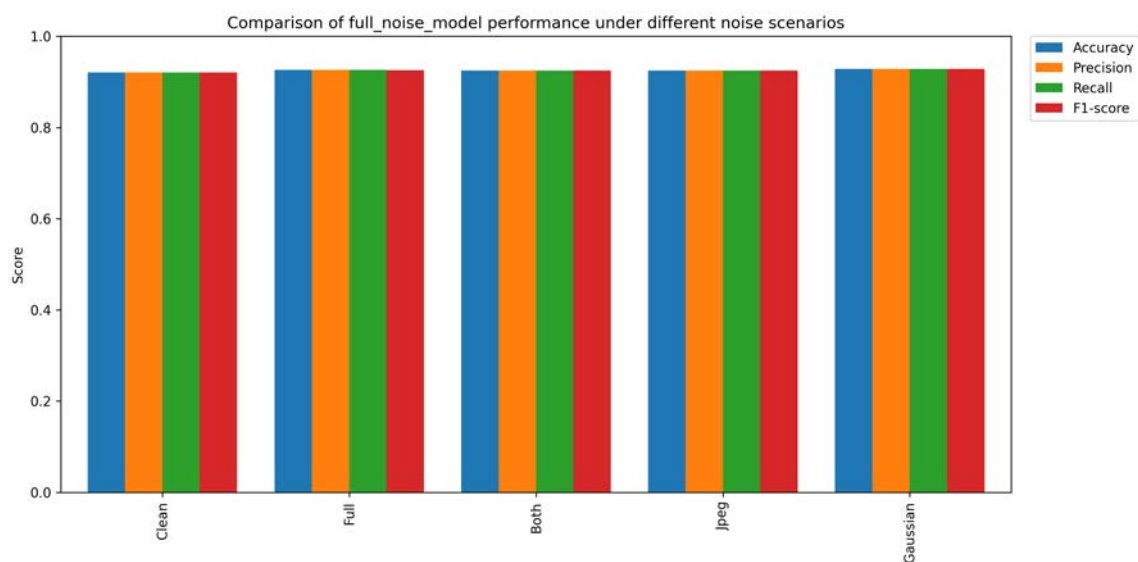
Σχήμα 7.24: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”



Σχήμα 7.25: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”



Σχήμα 7.26: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”



Σχήμα 7.27: Οι κοινές μετρικές του μοντέλου στα διαφορετικά σενάρια δοκιμής

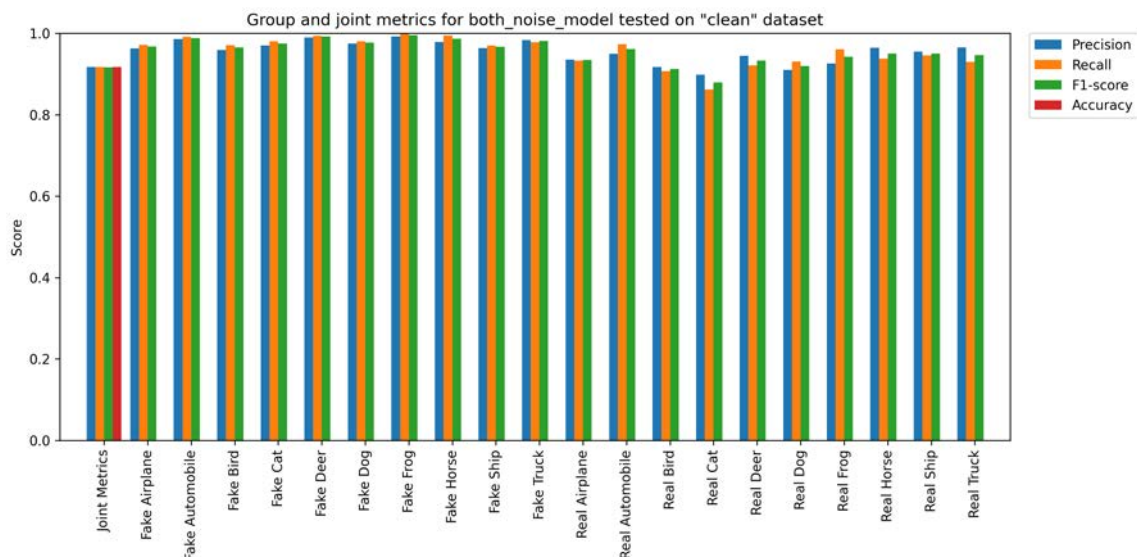
αυτή του dataset “clean”.

Στην περίπτωση “both” το μοντέλο σημειώνει παρόμοιες επιδόσεις. Συγκεκριμένα, στις κοινές μετρικές σημειώνει ποσοστά της τάξης του 92.85%-92.88% (σχήμα 7.30) και στους μέσους όρους των μετρικών στην κεφαλή 1 και 2 επιτυγχάνει 98.2% και 94.47% αντίστοιχα.

Και στο σενάριο “jpeg” το μοντέλο παρουσιάζει αντοχή στον θόρυβο και σταθερή απόδοση, αφού οι κοινές μετρικές αγγίζουν το 92.84-92.88% (σχήμα 7.31), οι μέσοι όροι των μετρικών κεφαλής 1 και 2 το 98.16% και 94.54% αντίστοιχα.

Το μοντέλο στη ΒΔ “gaussian” επιτυγχάνει υψηλή επίδοση σημειώνοντας 93.15%-93.17% (σχήμα 7.32) στις κοινές μετρικές και στους μέσους όρους των μετρικών της κεφαλής 1 και 2 σημειώνει 98.4% και 94.6% αντίστοιχα.

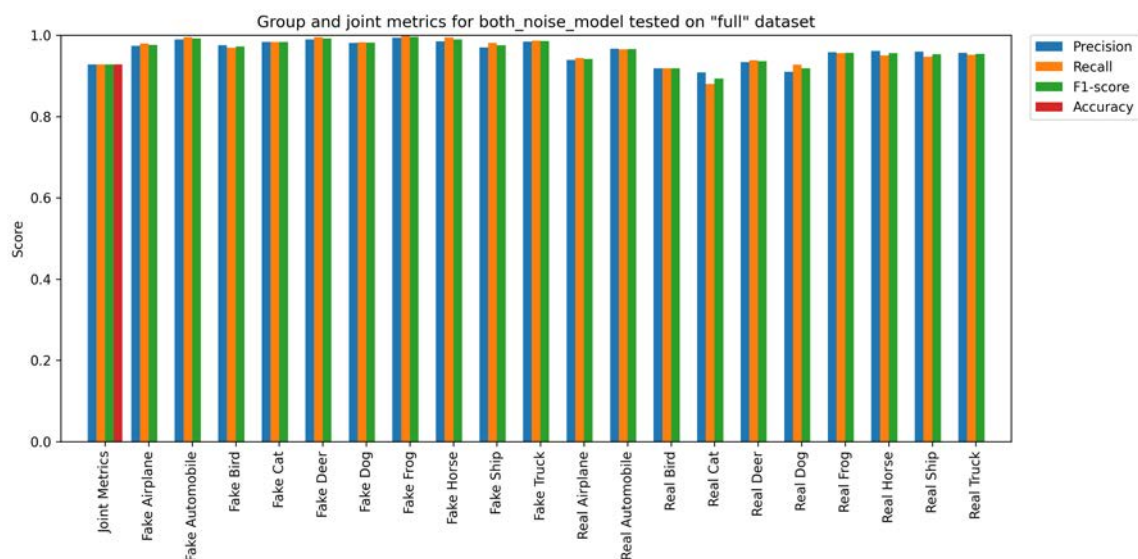
Το μοντέλο *both_noise_model* παρουσιάζει υψηλή αντοχή σε διαφορετικές συνθήκες θορύβου, αφού όπως φαίνεται και από το σχήμα 7.33 οι κοινές μετρικές κυμαίνονται σε μικρό εύρος τιμών, μικρότερο των 2 μονάδων. Το πιο “δύσκολο” σενάριο για το μοντέλο ήταν το “clean” και το πιο “εύκολο” το “gaussian”. Επίσης, όπως φάνηκε από όλα τα διαγράμματα των συνδυασμών κλάσεων, και αυτό δυσκολευόταν περισσότερο στην ταξινόμηση των αληθινών φωτογραφιών. Όλα τα ευρήματα είναι παρόμοια με εκείνα του *full_noise_model*.



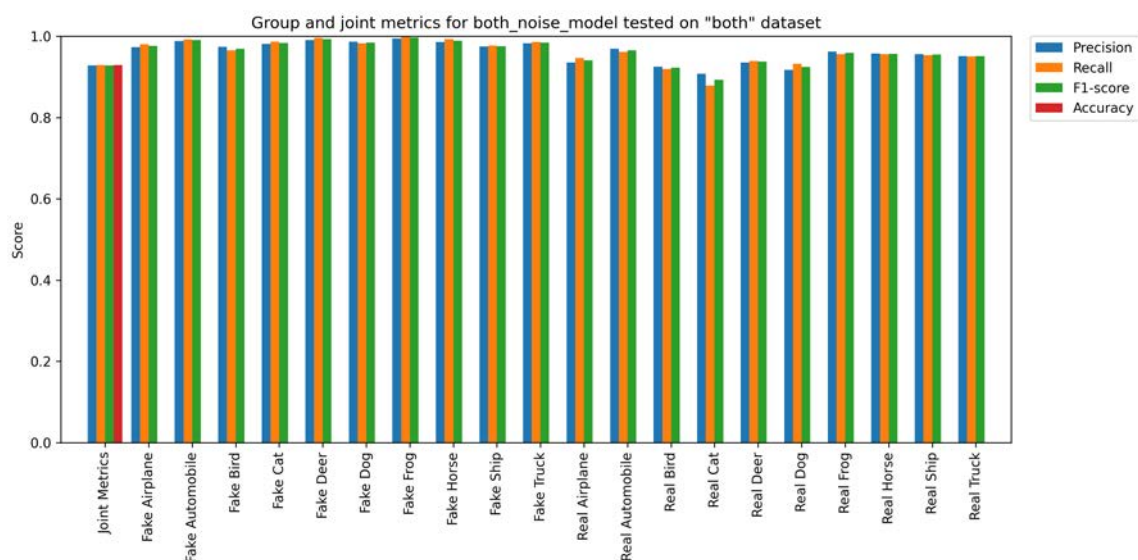
Σχήμα 7.28: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”

7.4.4 Ανάλυσης δοκιμής μοντέλου *jpeg_model*

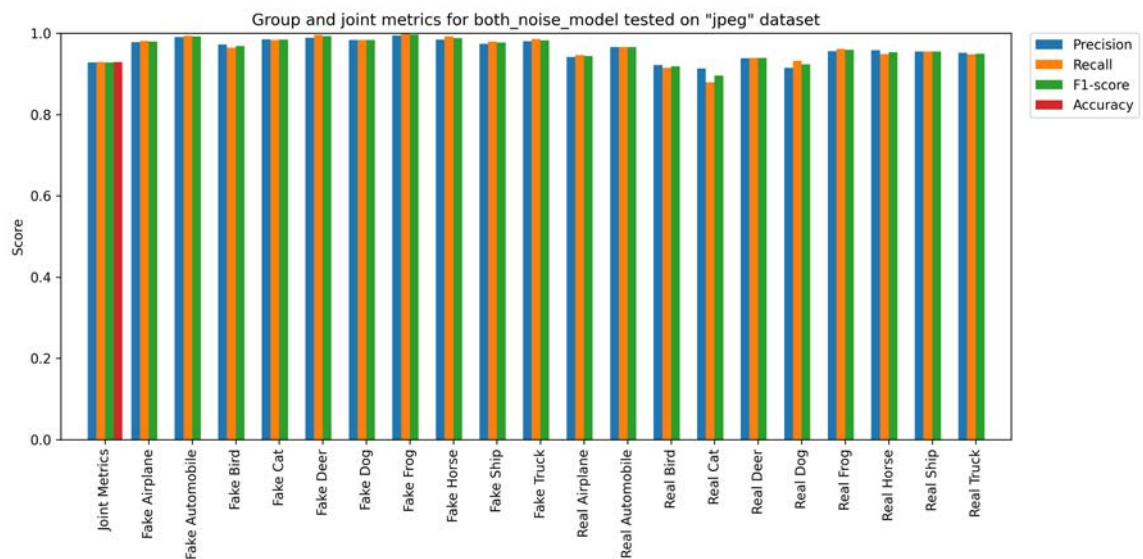
Το μοντέλο *jpeg_model* δοκιμάστηκε και στα 5 σενάρια δοκιμής. Τα αποτελέσματα βρίσκονται στον πίνακα A.4 και τα περισσότερα σχήματα στην ενότητα A.2.4. Ακολουθούν τα



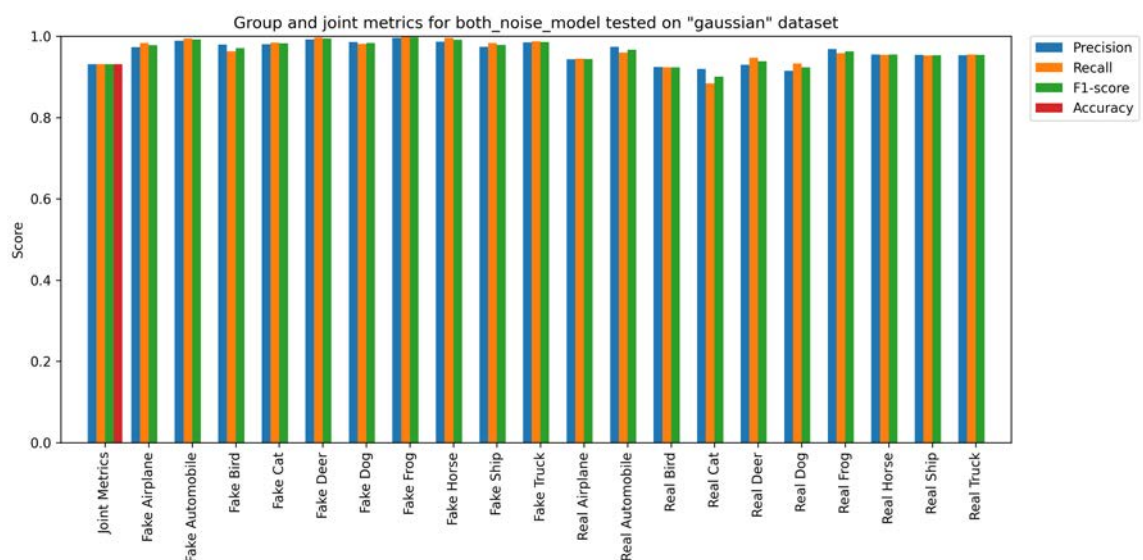
Σχήμα 7.29: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”



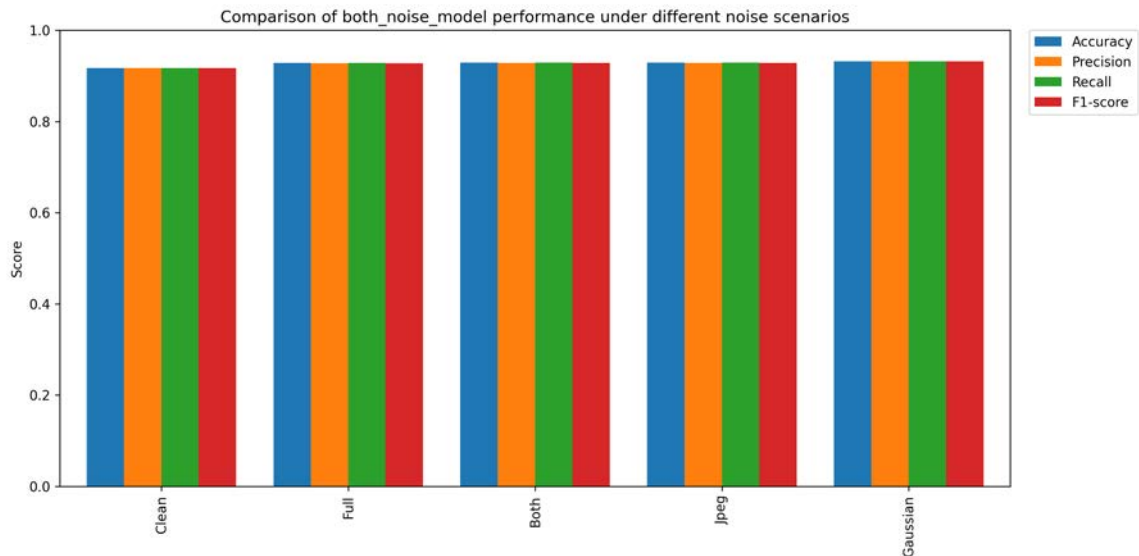
Σχήμα 7.30: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”



Σχήμα 7.31: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”



Σχήμα 7.32: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”



Σχήμα 7.33: Οι κοινές μετρικές για το μοντέλο σε κάθε σενάριο δοκιμής

αποτελέσματα δοκιμής.

Στη ΒΔ “clean” το μοντέλο παρουσιάζει υψηλή ικανότητα ταξινόμησης, αφού στις κοινές μετρικές σημειώνει 92.2%-92.3% (σχήμα 7.34) και στους μέσους όρους των μετρικών των κεφαλών 1 και 2 σημειώνει 97.5% και 94.5% αντίστοιχα.

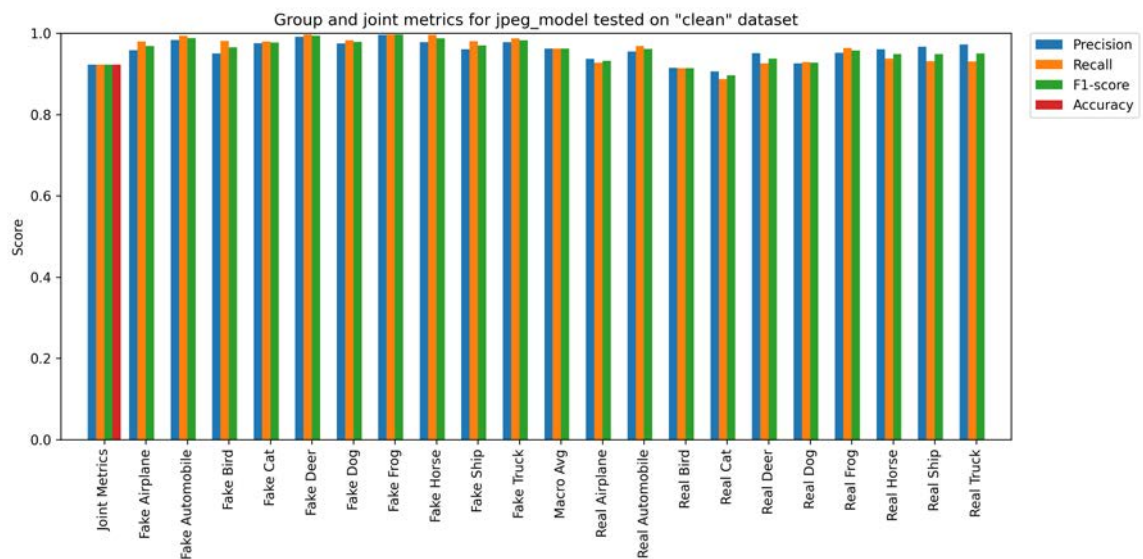
Στη ΒΔ “full” το μοντέλο διατηρεί την υψηλή ικανότητα ταξινόμησης, αφού στις κοινές μετρικές ξεπερνάει το 92.9% (σχήμα 7.35) και στους μέσους όρους των μετρικών των κεφαλών 1 και 2 το 98.17% και 94.59% αντίστοιχα.

Στη ΒΔ “both” το μοντέλο διατηρεί την υψηλή ικανότητα ταξινόμησης, αφού στις κοινές μετρικές ξεπερνάει το 93% (σχήμα 7.36) και στους μέσους όρους των μετρικών των κεφαλών 1 και 2 το 98.31% και 94.56% αντίστοιχα.

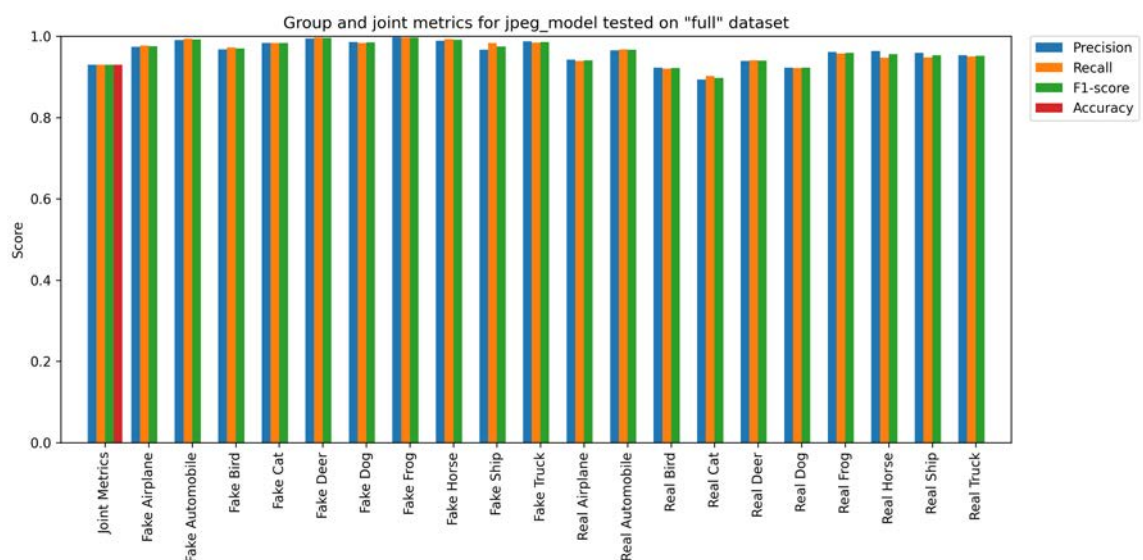
Παρομοίως με τα προηγούμενα σενάρια δοκιμής αυτού του μοντέλου και στο “jpeg” διατηρείται μια σταθερά υψηλή απόδοση. Οι κοινές μετρικές φτάνουν το 93.15% (σχήμα 7.37) ενώ οι μέσοι όροι των κεφαλών 1 και 2 φτάνουν το 98.37% και 94.61% αντίστοιχα.

Στο σενάριο δοκιμής “gaussian” για το μοντέλο δεν διαφοροποιούνται σημαντικά οι τελικές μετρικές. Οι κοινές μετρικές πλησιάζουν το 93.05% (σχήμα 7.38) και οι μέσοι όροι των μετρικών της κεφαλής 1 και κεφαλής 2 το 98.39% και το 94.47% αντίστοιχα.

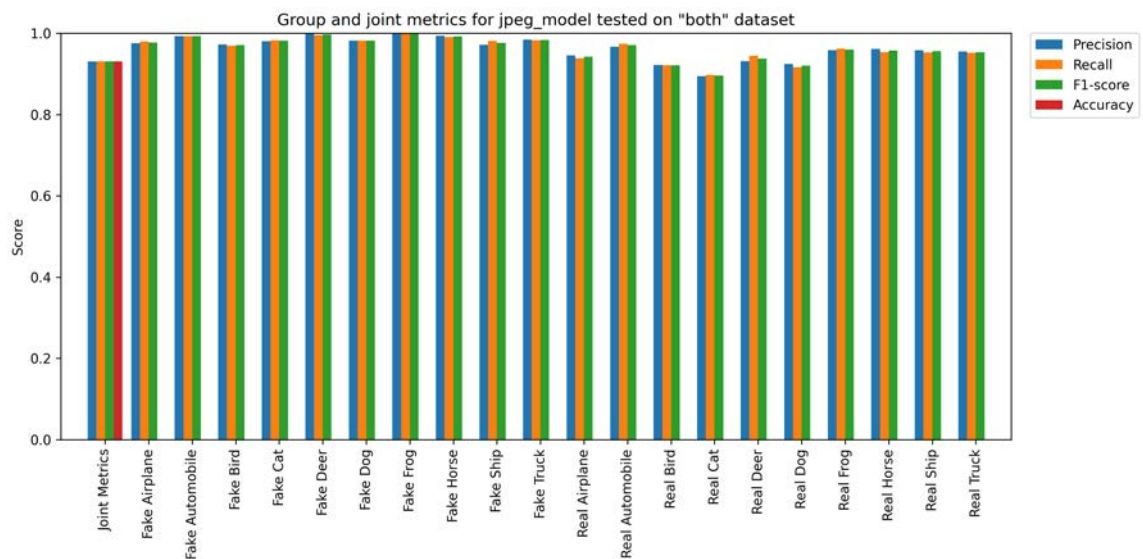
Το μοντέλο γενικά επιδεικνύει πολύ καλή και σταθερή συμπεριφορά σε οποιοδήποτε σενάριο δοκιμής (σχήμα 7.39). Τέλος, και σε αυτό το μοντέλο επιβεβαιώνεται ο μεγαλύτερος βαθμός δυσκολίας ταξινόμησης αληθινών εικόνων.



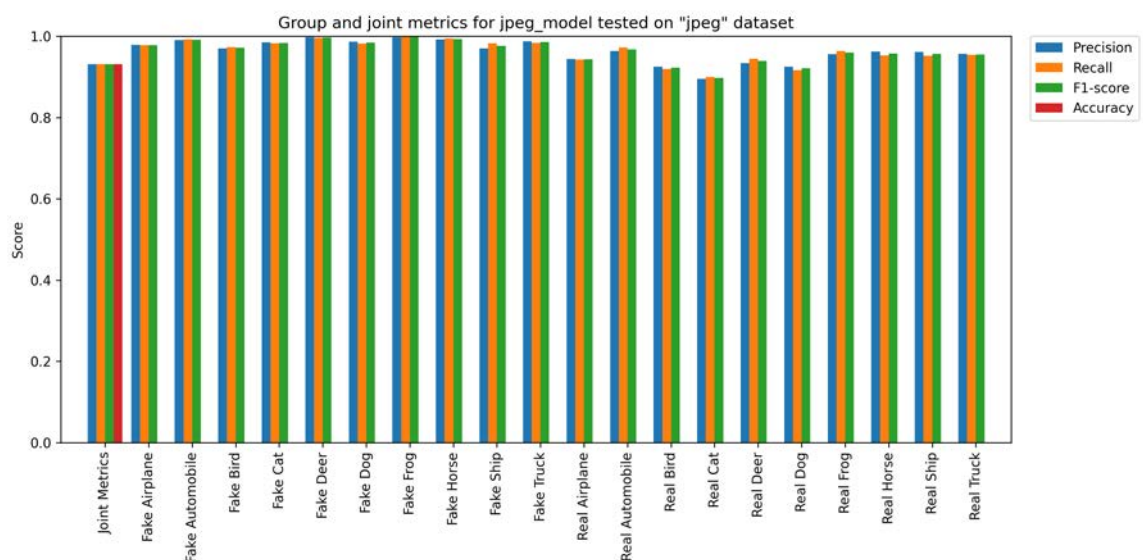
Σχήμα 7.34: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”



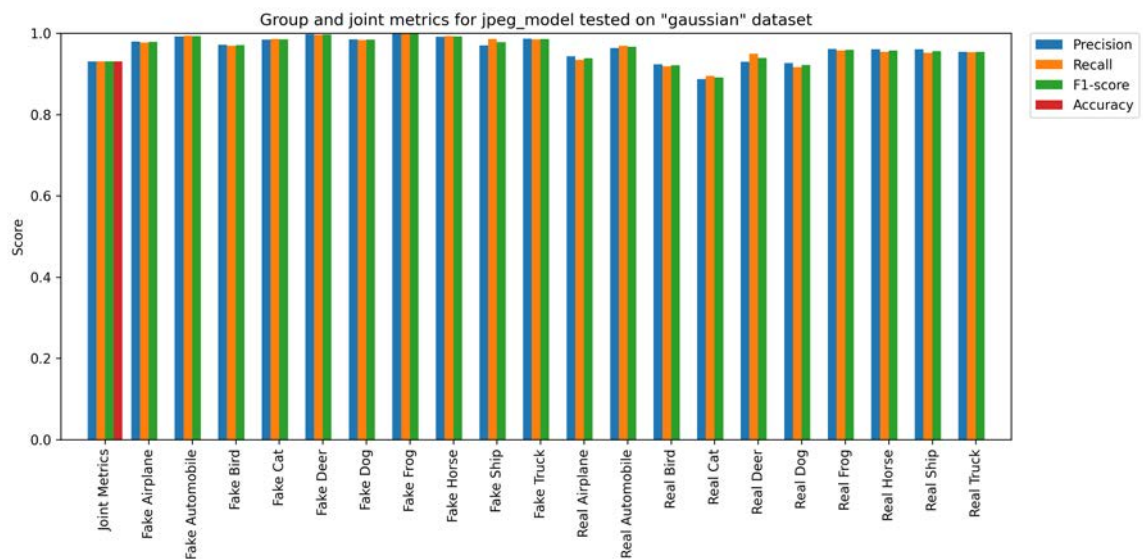
Σχήμα 7.35: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”



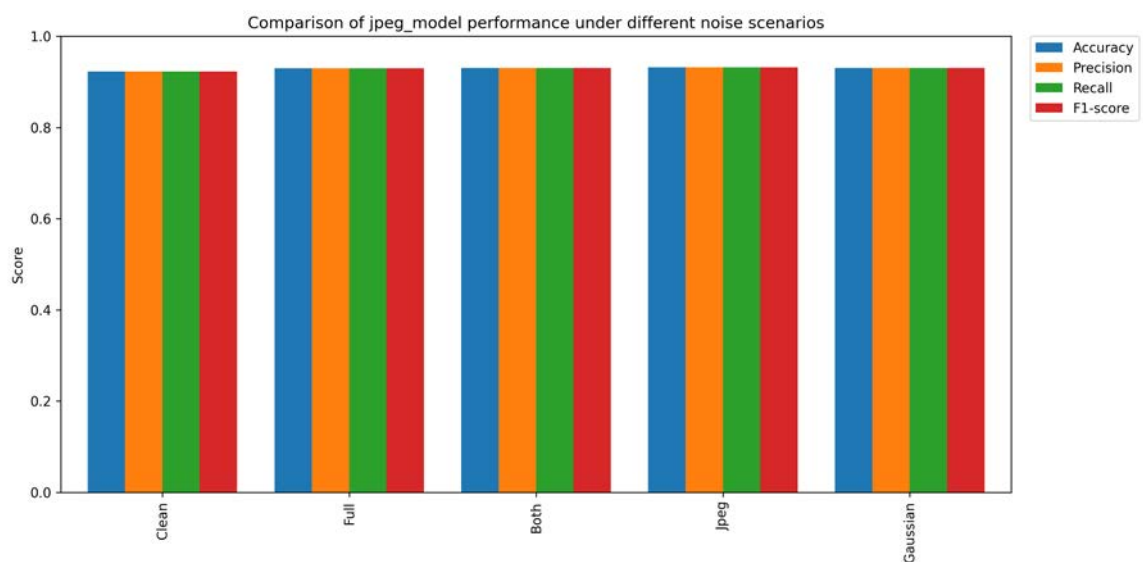
Σχήμα 7.36: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”



Σχήμα 7.37: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”



Σχήμα 7.38: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”



Σχήμα 7.39: Οι κοινές μετρικές του μοντέλου ανά σενάριο δοκιμής

7.4.5 Ανάλυση δοκιμής μοντέλου `gaussian_noise_model`

Τέλος, εξετάζεται το μοντέλο `gaussian_noise_model` και στα 5 σενάρια δοκιμής. Τα αποτελέσματα είναι καταγεγραμμένα στον πίνακα A.5 και τα σχήματα στην ενότητα A.2.5. Ακολουθεί η ανάλυση της επίδοσής του.

Το μοντέλο στο σύνολο δοκιμής με καθαρά δεδομένα δεν εμφανίζει ιδιαίτερη δυνατότητα ταξινόμησης. Η κοινή ακρίβεια και το recall έχουν τιμή 59.73%, εξαιρετικά χαμηλότερο ποσοστό από όλα τα πειράματα που έχουν διεξαχθεί. Το κοινό precision είναι καλύτερο από τις υπόλοιπες κοινές μετρικές με τιμή 71.43%, αλλά εξακολουθεί να είναι πολύ χαμηλό (σχήμα 7.40). Οι μέσοι όροι των μετρικών της κεφαλής 1 και 2 είναι εξίσου χαμηλοί, αφού στην κεφαλή 1 το ποσοστό ανέρχεται στο 76.9% για την ακρίβεια και το recall και 83.3% για το precision, ενώ για την κεφαλή 2 το ποσοστό ακριβείας και recall είναι 76.73% και το ποσοστό precision είναι 81.9%. Το μοντέλο σε αυτό το σενάριο συγχέει πολύ συχνά τις κλάσεις ανιχνεύοντας πολύ μικρό ποσοστό από τις πραγματικές εικόνες μιας κλάσης. Επίσης, σημειώνει παρόμοια ποσοστά και στις δύο κεφαλές, δηλαδή δυσκολεύεται εξίσου και στην δυαδική ταξινόμηση και στην ταξινόμηση δέκα κλάσεων. Μάλιστα, παρατηρώντας τις per-class-combination μετρικές το μοντέλο σε αντίθεση με τις προηγούμενες δοκιμές δυσκολεύεται περισσότερο στην ταξινόμηση συνθετικών εικόνων. Οι μετρικές για τους συνδυασμούς κλάσεων “fake automobile”, “fake bird”, “fake ship”, “fake truck” έχουν πολύ χαμηλά ποσοστά σωστής ταξινόμησης όλων των δειγμάτων και μπορεί να υποτεθεί πως το μοντέλο κάνει σε σημαντικό βαθμό τυχαία επιλογή.

Στο συγκεκριμένο πείραμα με δεδομένα “full”, σε αντίθεση με το προαναφερθέν, το μοντέλο εμφανίζει εντελώς διαφορετική συμπεριφορά. Οι κοινές μετρικές έχουν τιμές από 85.93% - 87.3% (σχήμα 7.41) και οι μέσοι όροι των μετρικών των κεφαλών 1 και 2 έχουν τιμές 93.8% -94.17% και 90.75% -91.24% αντίστοιχα. Η απόδοσή του είναι σαφώς βελτιωμένη αλλά εξακολουθεί να μην συναγωνίζεται τα μοντέλα `full_noise_model`, `both_noise_model` και `jpeg_model`.

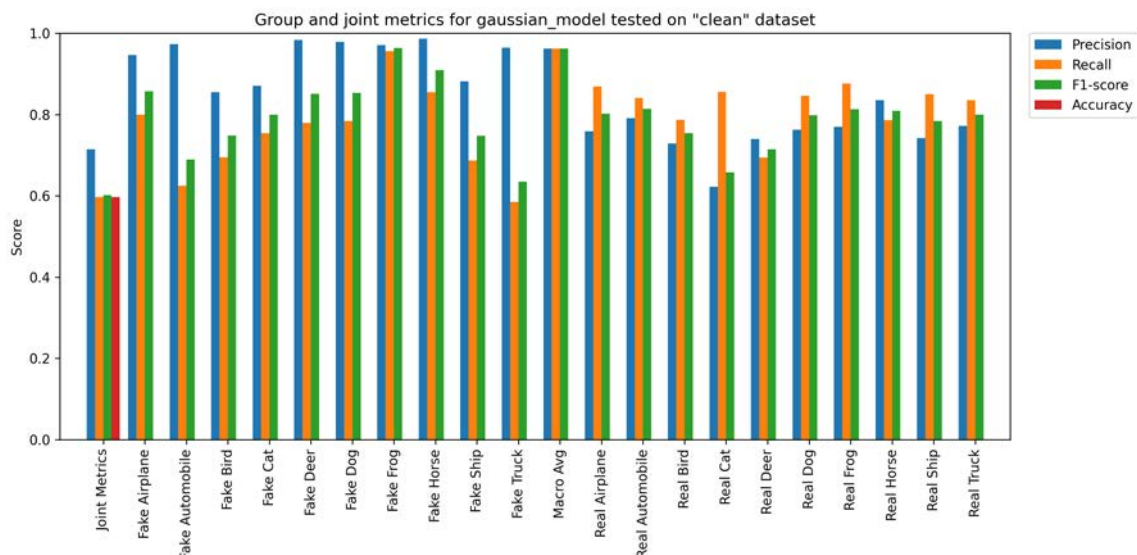
Η απόδοση στο σύνολο “both” είναι βελτιωμένη, αφού πετυχαίνει ποσοστά για τις κοινές μετρικές κοντά στο 90.7% (σχήμα 7.42). Επίσης, και οι δύο κεφαλές έχουν καλύτερη απόδοση, αφού για την κεφαλή 1 τα αποτελέσματα κυμαίνονται στο 96.74% και για την κεφαλή 2 στο 93.5%.

Στο πείραμα με δεδομένα “jpeg” μειώνεται και πάλι η απόδοση του μοντέλου. Οι κοινές μετρικές κυμαίνονται μεταξύ 87.39% και 88.26% (σχήμα 7.43), ενώ για την κεφαλή 1 μεταξύ

95.32% και 95.47% και για την κεφαλή 2 μεταξύ 91.38% και 91.76%.

Το μοντέλο εμφανίζει την καλύτερή του επίδοση στο dataset “gaussian”. Στις κοινές μετρικές σημειώνει 93.6% (σχήμα 7.44) και στις μετρικές για την κεφαλή 1 και την κεφαλή 2 σημειώνει 98.43% και 95.1% αντίστοιχα.

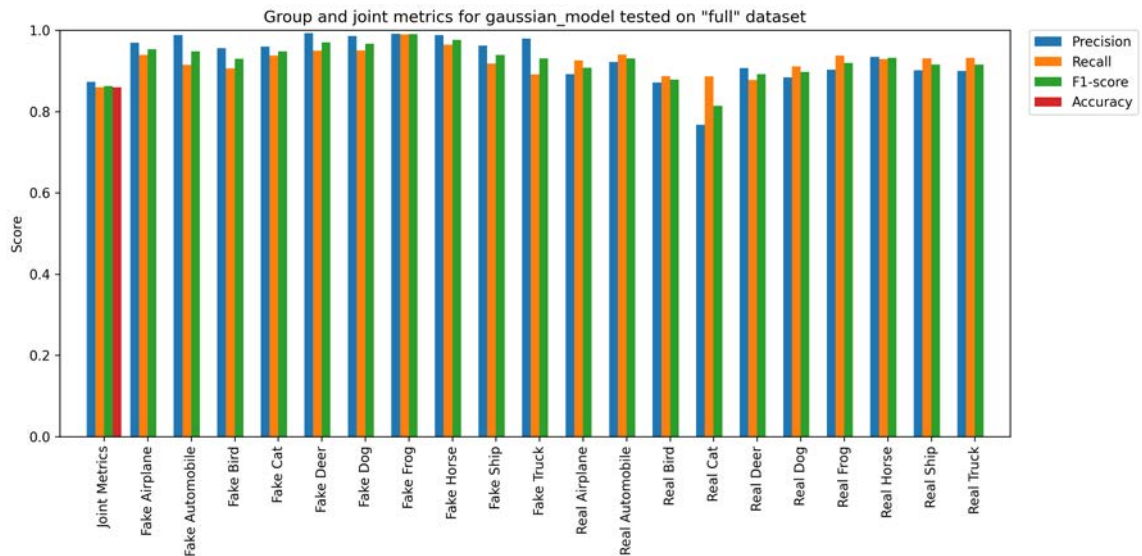
Είναι φανερό πως το μοντέλο δεν έχει καθόλου σταθερή συμπεριφορά και απόδοση (σχήμα 7.45). Όταν εκπαιδεύεται μόνο από καθαρές εικόνες και εικόνες με κανονικό θόρυβο, η απόδοσή του εξαρτάται απόλυτα από τις εικόνες στις οποίες δοκιμάζεται. Εάν οι εικόνες δοκιμής έχουν και εκείνες κανονικό θόρυβο, σημειώνει υψηλά ποσοστά ακριβείας. Όταν όμως δεν αντιμετωπίζει εικόνες με αυτό τον θόρυβο, φαίνεται ότι τις ταξινομεί με πολύ μικρή ακρίβεια.



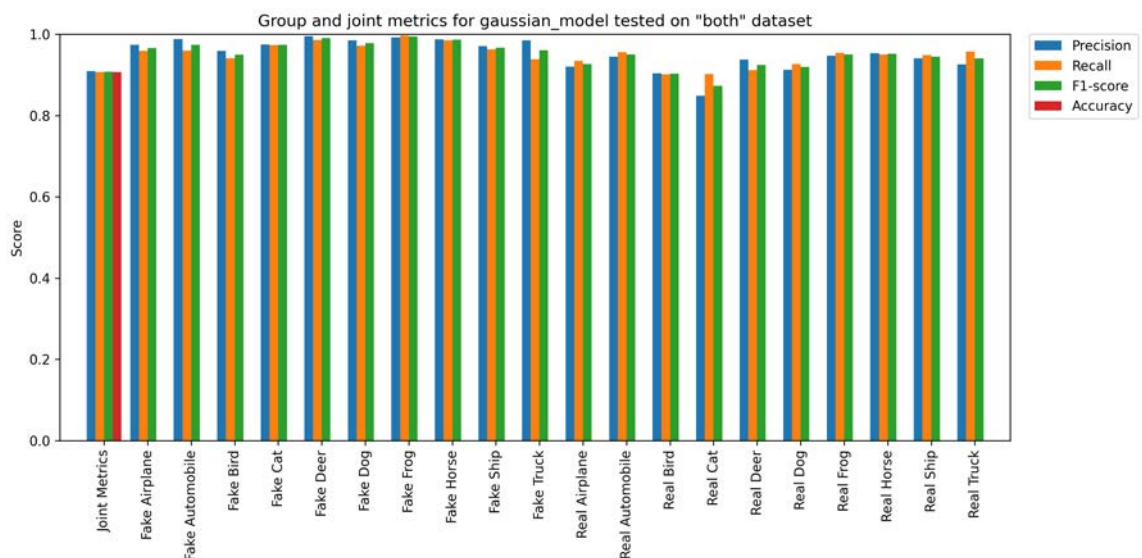
Σχήμα 7.40: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “clean”

7.5 Σύγκριση απόδοσης μοντέλων ανά συνθήκη θορύβου

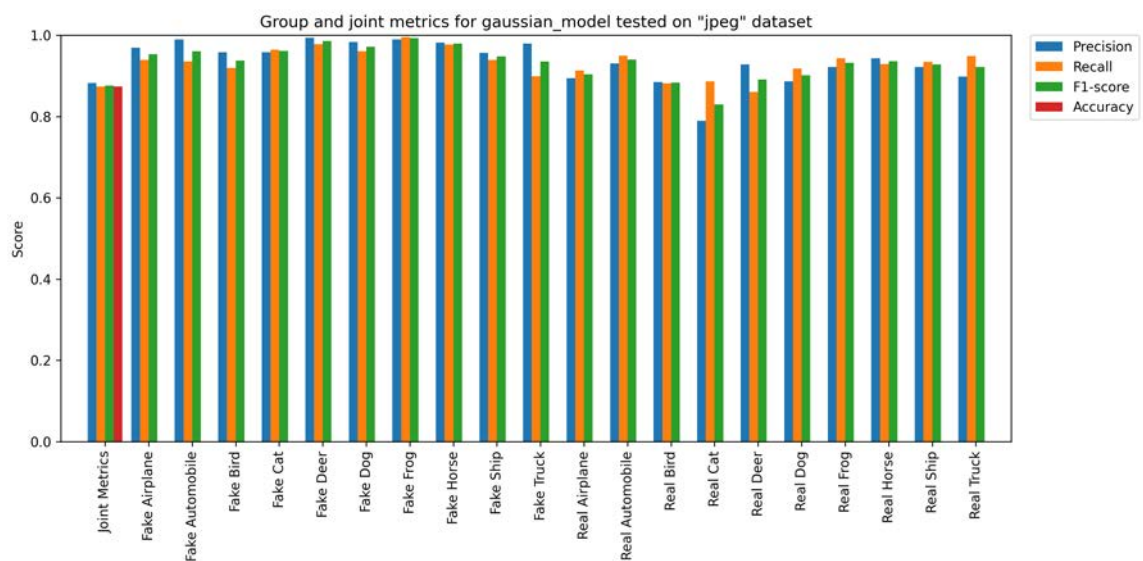
Στην συγκεκριμένη ενότητα πραγματοποιείται σύγκριση των κοινών μετρικών μεταξύ των 5 μοντέλων για κάθε σύνολο δοκιμής. Όπως παρατηρείται από τα σχήματα 7.46, 7.47, 7.48, 7.49 και 7.50, το clean_model και το gaussian_model έχουν την πιο ασταθή συμπεριφορά, σημειώνοντας κάποιες από τις χαμηλότερες τιμές ακριβείας για σενάριο δοκιμής στα οποία δεν εκπαιδεύτηκαν, και κάποιες από τις καλύτερες τιμές ακριβείας για το σενάριο δοκιμής που αντιστοιχεί στο σενάριο εκπαίδευσής τους. Σε αντίθεση με αυτά τα μοντέλα τα υπόλοιπα τρία έχουν σταθερά υψηλή απόδοση, όμως το καλύτερο φαίνεται ότι είναι το μοντέλο που



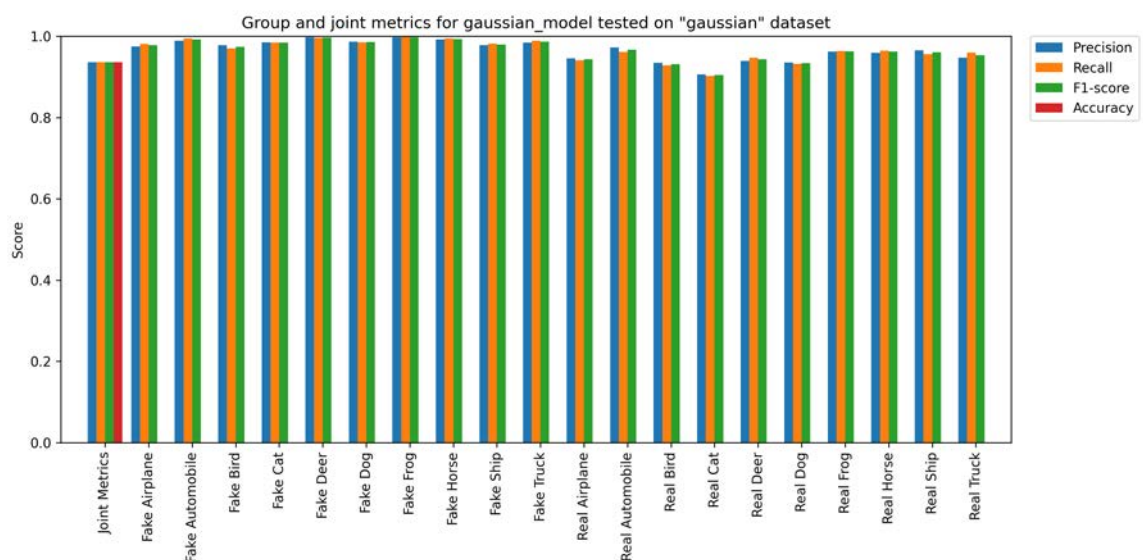
Σχήμα 7.41: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “full”



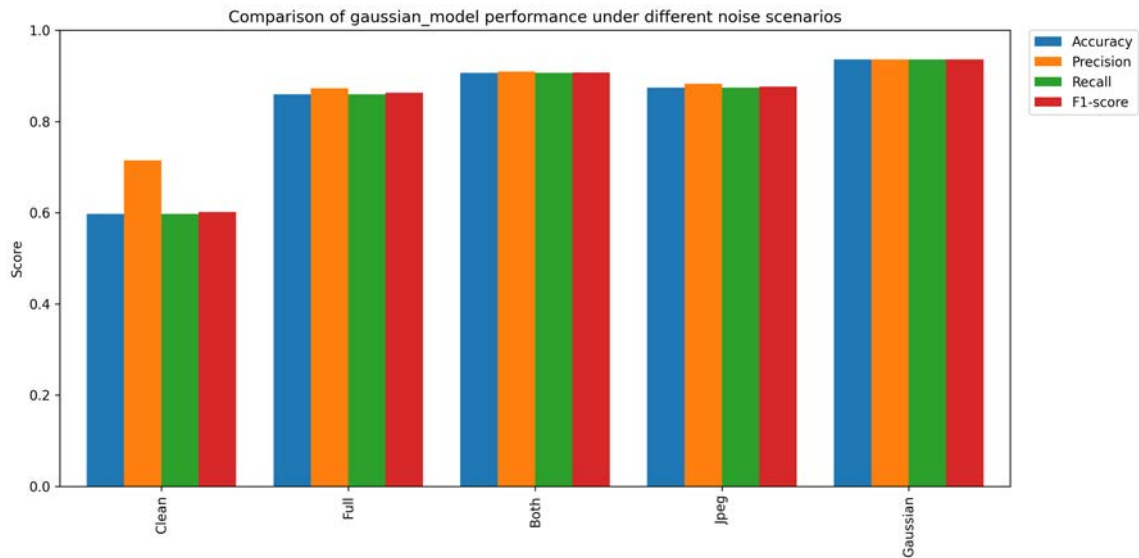
Σχήμα 7.42: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “both”



Σχήμα 7.43: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “jpeg”

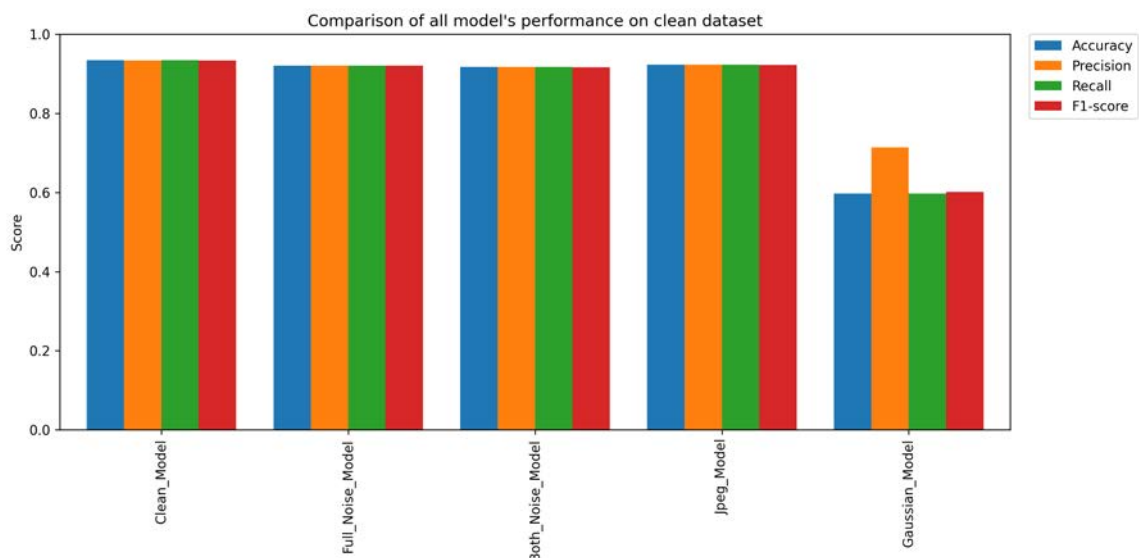


Σχήμα 7.44: Οι κοινές μετρικές και οι μετρικές συνδυασμών κλάσεων για dataset “gaussian”

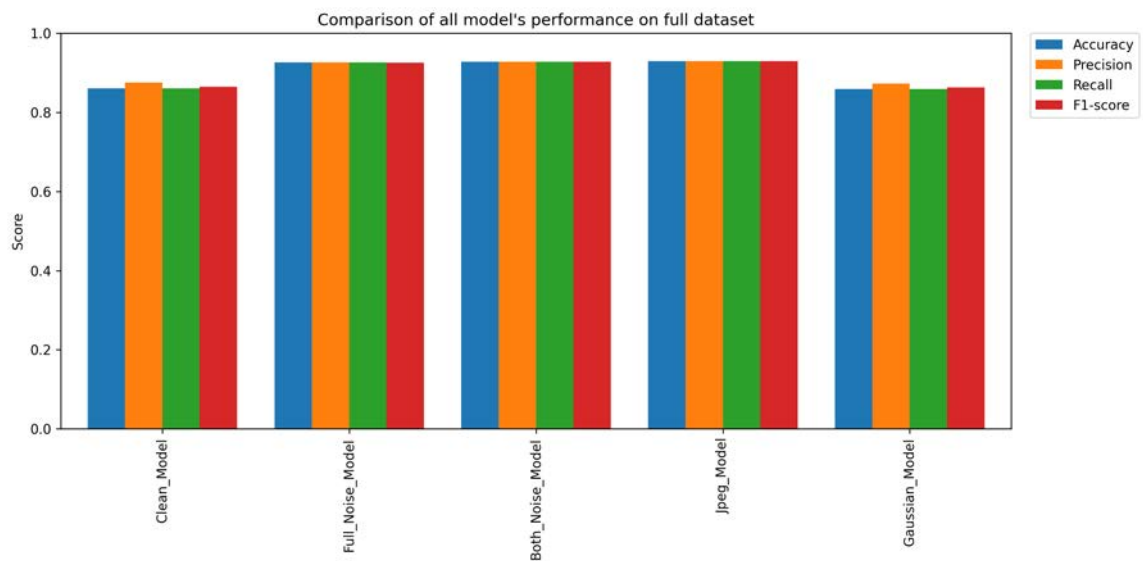


Σχήμα 7.45: Οι κοινές μετρικές ανά σενάριο δοκιμής

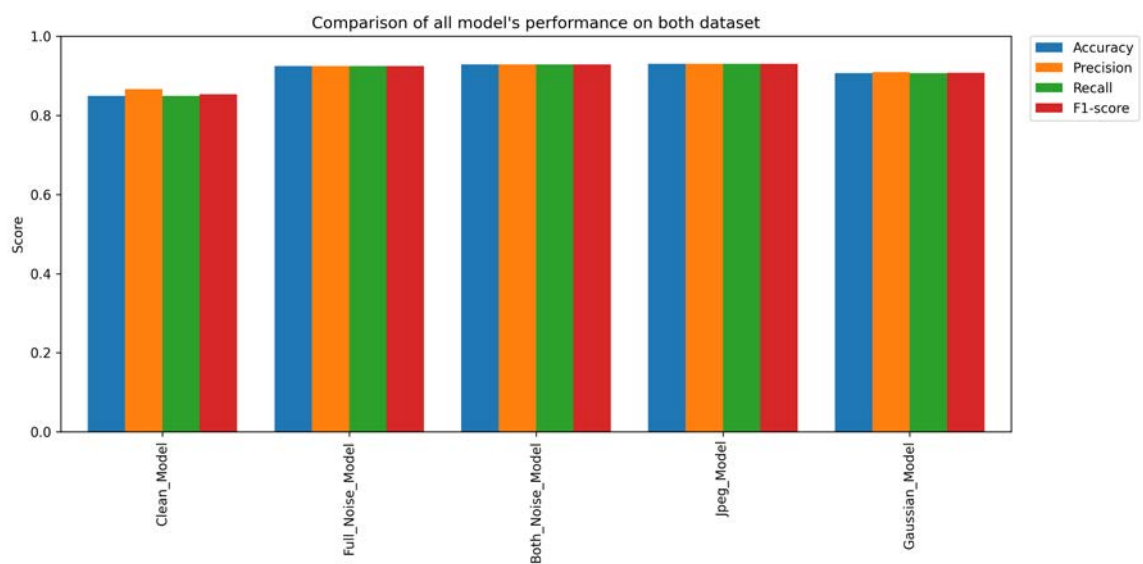
εκπαιδεύτηκε στο both dataset. Από όλα αυτά συμπεραίνεται πως το μοντέλο μπορεί να διατηρήσει μεγάλη ανθεκτικότητα απέναντι σε διαφορετικές συνθήκες θορύβου, αρκεί να έχει εκπαιδευτεί σε κατάλληλο dataset. Η εκπαίδευση σε καθαρά δεδομένα δεν επαρκεί απαραίτητα, ειδικά για εικόνες που έχουν μόνο Γκαουσιανό θόρυβο, ενώ η εκπαίδευση με εικόνες με Γκαουσιανό θόρυβο οδηγεί σε απρόβλεπτη συμπεριφορά και αποτελέσματα.



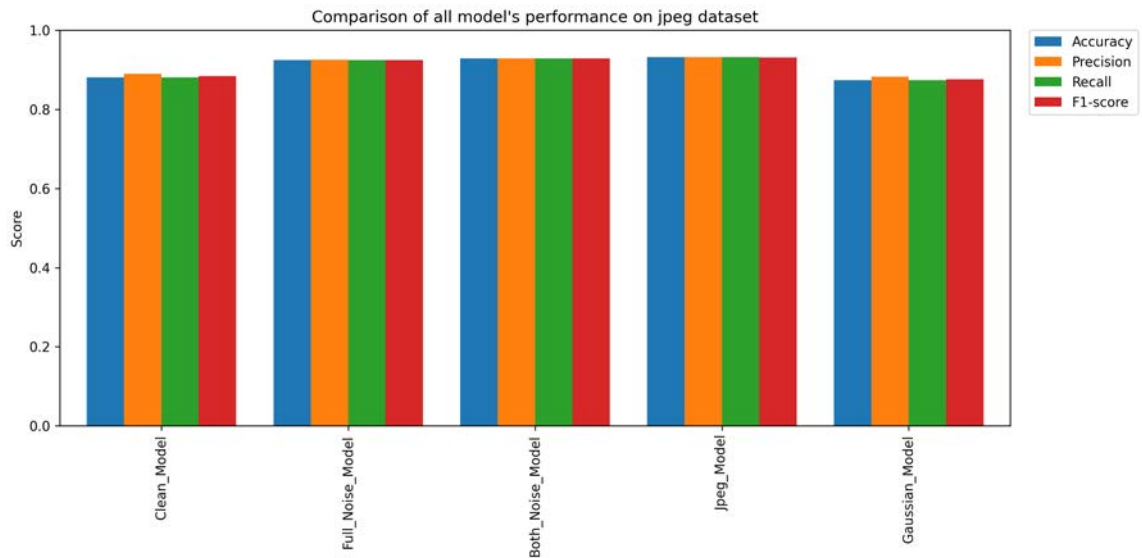
Σχήμα 7.46: Η απόδοση μοντέλων για το dataset “clean”



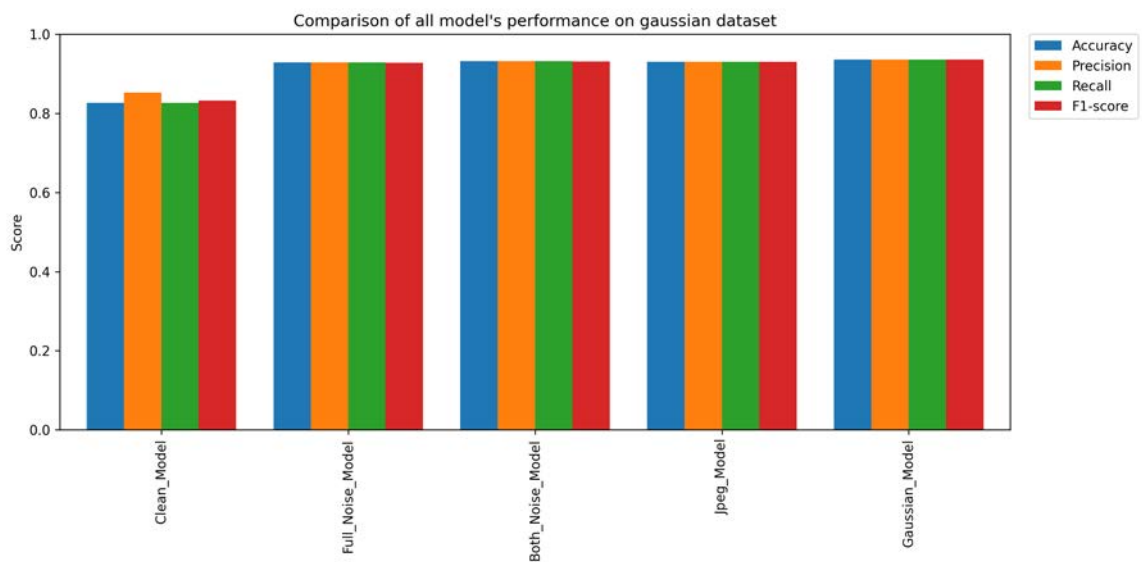
Σχήμα 7.47: Η απόδοση μοντέλων για το dataset “full”



Σχήμα 7.48: Η απόδοση μοντέλων για το dataset “both”



Σχήμα 7.49: Η απόδοση μοντέλων για το dataset “jpeg”



Σχήμα 7.50: Η απόδοση μοντέλων για το dataset “gaussian”

Κεφάλαιο 8

Έρευνα απόδοσης ResNet50Classifier σε σχέση με παραδοσιακές τεχνικές μηχανικής μάθησης

Στο κεφάλαιο αυτό αναλύεται το τρίτο πείραμα και εξετάζονται τα αποτελέσματά του. Το πείραμα της έρευνας αυτής χωρίζεται σε δύο διακριτά μέρη με διαφορετικούς στόχους το κάθε ένα. Στο πρώτο μέρος αναπτύσσονται μέσα μηχανικής μάθησης που εργάζονται στον πίνακα χαρακτηριστικών των εικόνων για να τις ταξινομήσουν. Σε αυτό το στάδιο μελετάται η απόδοση αυτών των συστημάτων και συγκρίνονται με το ResNet50Classifier, για να διαπιστωθεί εάν υπερτερούν του νευρωνικού δικτύου που αναπτύχθηκε στο πρώτο πείραμα. Στο δεύτερο στάδιο αναπτύσσονται τα ίδια μοντέλα μηχανικής μάθησης, αλλά αυτή τη φορά εκπαιδεύονται στο διάνυμα χαρακτηριστικών του μοντέλου ResNet50Classifier. Στόχος είναι να δημιουργηθεί ένα υβριδικό μοντέλο ταξινόμησης με το ResNet50Classifier ως μέθοδο εξαγωγής χαρακτηριστικών και ένα μοντέλο ML ως ταξινομητή αυτών των χαρακτηριστικών.

8.1 Ανάπτυξη μοντέλων ML και εκπαίδευσή τους απευθείας στις εικόνες

Σε αυτό το στάδιο του πειράματος αναπτύσσονται έξι μοντέλα μηχανικής μάθησης, τα οποία εκπαιδεύονται στις 80000 εικόνες του σετ εκπαίδευσης του CIFAKE [5, 6] χωρίς θόρυβο και ελέγχονται στις 20000 εικόνες του συνόλου δοκιμής χωρίς θόρυβο. Συγκεκριμένα, αναπτύσσονται δύο μοντέλα linear SVM, δύο μοντέλα Random Forest και δύο μοντέλα XG-

Boost. Σε κάθε κατηγορία ML μοντέλων το ένα από τα δύο είναι υπεύθυνο για την δυαδική ταξινόμηση ανάμεσα σε αληθινές και συνθετικές εικόνες και το άλλο είναι υπεύθυνο για την ταξινόμηση στις κλάσεις του CIFAR-10 [6].

8.1.1 Εξαγωγή δεδομένων από εικόνες και κανονικοποίηση διανύσματος δεδομένων

Πρώτο βήμα για την ανάπτυξη των μοντέλων αποτελεί η κατάλληλη επεξεργασία των δεδομένων. Για να εξαχθούν τα χαρακτηριστικά κάθε εικόνας ενός συνόλου, ορίζονται αρχικά δύο dataloaders (σχήμα B.21) για μοντέλα ML, από τα οποία το ένα χρησιμοποιείται για δεδομένα εκπαίδευσης και το δεύτερο για δεδομένα δοκιμής. Τα dataloaders φορτώνουν το καθένα το αντίστοιχο dataset που ορίζεται από τον διαχειριστή δεδομένων CIFAKE-DatasetHandler (ενότητα 4.2). Στα datasets αυτά ο μοναδικός μετασχηματισμός που εφαρμόζεται είναι η μετατροπή της εικόνας σε τανυστή με κανονικοποιημένες τιμές, όπως φαίνεται και στο σχήμα B.22. Δεν γίνεται resize, αφού οι εικόνες είναι μεγέθους 32×32 και δεν χρειάζεται να περάσουν από το μοντέλο βαθιάς μάθησης, για να είναι αναγκαίο να αυξηθούν οι διαστάσεις. Επίσης, δεν γίνεται κάποιος άλλος μετασχηματισμός, για να μην υπάρξουν κίνδυνοι αλλοίωσης των προτύπων. Στη συνέχεια, οι εικόνες των dataloaders εισέρχονται σε μια συνάρτηση, την `flatten_loader`, η οποία επιπεδοποιεί τον πίνακα χαρακτηριστικών της κάθε εικόνας και τον μετατρέπει σε διάνυσμα με κατάλληλη αλλαγή των διαστάσεων. Η υλοποίηση της συνάρτησης `flatten_loader` παρουσιάζεται στο σχήμα B.23 και η χρήση της για την εξαγωγή χαρακτηριστικών στο σχήμα B.24.

Τέλος, κανονικοποιούνται τα διανύσματα χαρακτηριστικών και εφαρμόζεται PCA (Principal Component Analysis), μία μέθοδος μείωσης διαστάσεων, προκειμένου οι διαστάσεις των διανυσμάτων να μειωθούν σε 512. Η μείωση αυτή γίνεται για δύο λόγους. Από την μία με την μείωση των διαστάσεων γίνεται και μείωση των υπολογισμών και επιταχύνεται δραστικά η εκπαίδευση των μοντέλων. Γενικά, λόγω περιορισμένων υπολογιστικών πόρων δεν είναι δυνατόν τα μοντέλα να εκπαιδευτούν στα αρχικά διανύσματα χαρακτηριστικών και έτσι επιλέγεται η μείωση διαστάσεων, η οποία ταυτόχρονα δεν οδηγεί σε σημαντική απώλεια πληροφοριών. Ο δεύτερος λόγος που πραγματοποιείται αυτή η μείωση και επιλέγεται το 512 ως ο νέος αριθμός διαστάσεων είναι για να υπάρξει στη συνέχεια δίκαιη σύγκριση μεταξύ των μοντέλων που εκπαιδεύτηκαν απευθείας στις εικόνες και των μοντέλων που εκπαιδεύτηκαν στον χώρο των χαρακτηριστικών του ResNet50Classifier, ο οποίος, όπως θα αναφερθεί

και αργότερα, έχει μέγεθος 512.

Ακολουθεί η μαθηματική ανάλυση της διαδικασίας δημιουργίας του χώρου χαρακτηριστικών.

Αρχικά, έστω εικόνα $\mathbf{I} \in \mathbb{R}^{W \times H \times C}$, με:

- W : πλάτος εικόνας,
- H : ύψος εικόνας,
- C : αριθμός καναλιών εικόνας

Η εικόνα μετατρέπεται σε διάνυσμα στην `flatten_loader` ως εξής:

$$\mathbf{x} = \text{vec}(\mathbf{I} \in \mathbb{R}^d, \quad \text{με } d = W \cdot H \cdot C) \quad (8.1)$$

Το σύνολο εκπαίδευσης είναι:

$$\mathbf{X}_{\text{raw}} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \dots \\ \mathbf{x}^{(n)} \end{bmatrix} \in \mathbb{R}^{n \times d} \quad (8.2)$$

Για κάθε χαρακτηριστικό $j \in \{1, \dots, d\}$ ισχύει:

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij} \quad (8.3)$$

$$\sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2} \quad (8.4)$$

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \quad (8.5)$$

Έτσι, ο πίνακας κανονικοποιημένων δεδομένων είναι:

$$\mathbf{Z} \in \mathbb{R}^{n \times d}, \quad \text{με } Z_{ij} = z_{ij} \quad (8.6)$$

Ακολουθεί η μείωση διαστασιμότητας και προβολή στον νέο χώρο. Ο πίνακας συνδιακύμανσης είναι:

$$\mathbf{C} = \frac{1}{n} \mathbf{Z}^\top \mathbf{Z} \quad (8.7)$$

Έπειτα, υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα:

$$\mathbf{C} \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \dots, d \quad (8.8)$$

Επιλέγονται οι k κύριες συνιστώσες:

$$\mathbf{W}_k = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k] \in \mathbb{R}^{d \times k} \quad (8.9)$$

Τέλος, υπολογίζονται οι προβολές στον νέο μειωμένο χώρο:

$$\mathbf{X}_{\text{pca}} = \mathbf{Z} \cdot \mathbf{W}_k \in \mathbb{R}^{n \times k} \quad (8.10)$$

Στην συγκεκριμένη περίπτωση έχουμε:

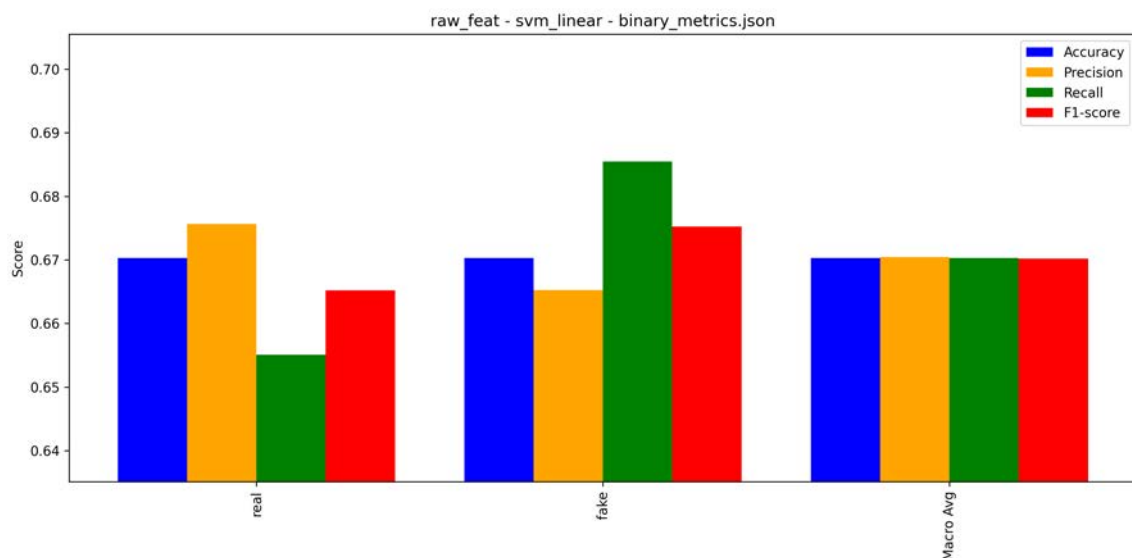
- $W = H = 32$ (μέγεθος των διαστάσεων πλάτους, ύψους της εικόνας),
- $C = 3$,
- $n = 80000$,
- $k = 512$.

8.1.2 Χρήση linear SVM για το πρώτο στάδιο του πειράματος

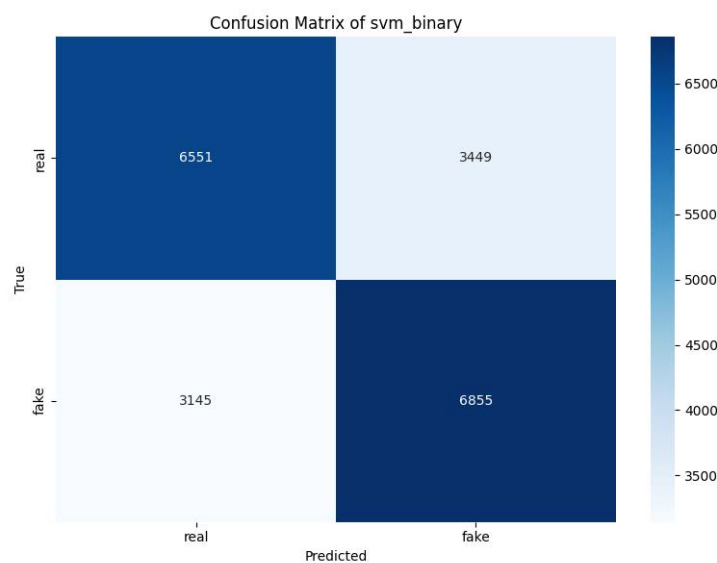
Στο πρώτο στάδιο, όπως προαναφέρθηκε, χρησιμοποιείται δυαδικός ταξινομητής και ταξινομητής 10 κλάσεων linear SVM. Για την υλοποίηση των μηχανών αυτών χρησιμοποιείται η βιβλιοθήκη sklearn. Οι ταξινομητές έχουν 10000 μέγιστες επαναλήψεις για την εκπαίδευσή τους. Τα μοντέλα, αφού εκπαιδεύτηκαν στα δεδομένα εκπαίδευσης, δοκιμάστηκαν στα δεδομένα δοκιμής. Επειδή το κάθε μοντέλο στην ουσία υλοποιεί την αντίστοιχη λειτουργία κάθε κεφαλής του ResNet50Classifier, καταγράφηκαν απλώς οι μετρικές ανά κλάση και οι μέσοι όροι των μετρικών κάθε μοντέλου. Επίσης, καταγράφηκαν τα confusion matrices.

Παρατηρώντας και τα δύο σχήματα (8.1 και 8.3) είναι φανερό πως και τα δύο SVMs είναι υποδεέστερα του νευρωνικού δικτύου στην ταξινόμηση εικόνων, καθώς οι μετρικές τους είναι αρκετά υποδεέστερες από τις μετρικές των αντίστοιχων κεφαλών. Ενώ το νευρωνικό σημειώνει πάνω από 98% ακρίβεια στην δυαδική ταξινόμηση, το SVM σημειώνει ακρίβεια μόλις 67.03%. Οι υπόλοιπες μετρικές (μέσο recall, μέσο precision, μέσο f1 score) είναι εξίσου χαμηλές και κυμαίνονται περίπου στο 67.03%. Η διαφορά επιπέδου απόδοσης μεταξύ νευρωνικού και SVM είναι ακόμη πιο εμφανής στην ταξινόμηση σε 10 κλάσεις, όπου οι μετρικές έχουν σχεδόν διαφορά 49 μονάδων. Η ακρίβεια και το μέσο recall για το SVM είναι σχεδόν στο 46.51%, το μέσο precision 45.39% και το μέσο f1 score 44.89%. Παρατηρώντας, επίσης, τα confusion matrices (σχήματα 8.2 και 8.4) διαπιστώνεται μεγάλη δυσκολία στην

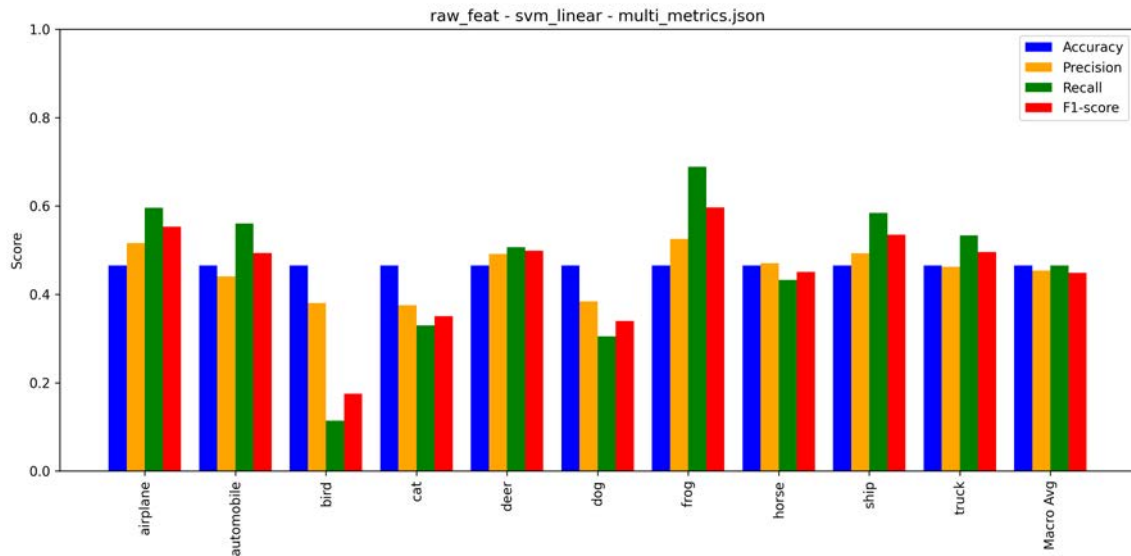
αναγνώριση της κλάσης bird. Μάλιστα, το μοντέλο αναγνώρισε σωστά μόλις τις 228 από τις 1000 εικόνες. Επίσης, αρνητικά αποτελέσματα λαμβάνονται και από το δυαδικό ταξινομητή όπου παρατηρείται μεγαλύτερη δυσκολία στον εντοπισμό αληθινών εικόνων.



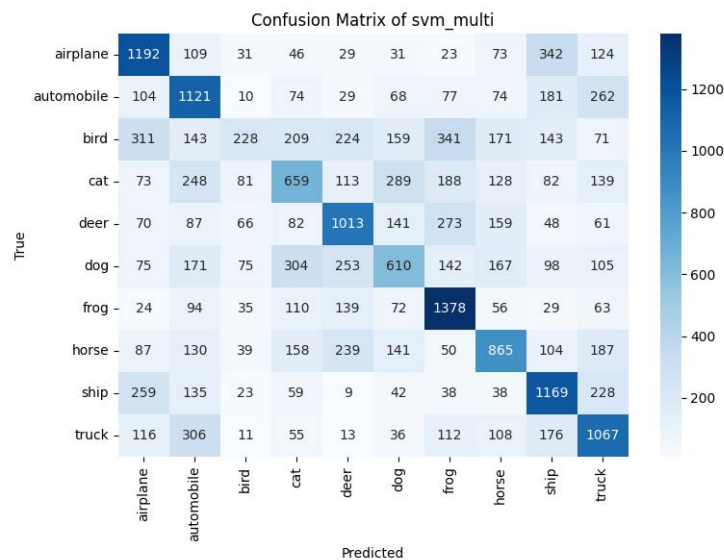
Σχήμα 8.1: Οι μετρικές για το binary linear SVM εκπαιδευμένο απευθείας στον χώρο των εικόνων.



Σχήμα 8.2: Το confusion matrix για το binary linear SVM.



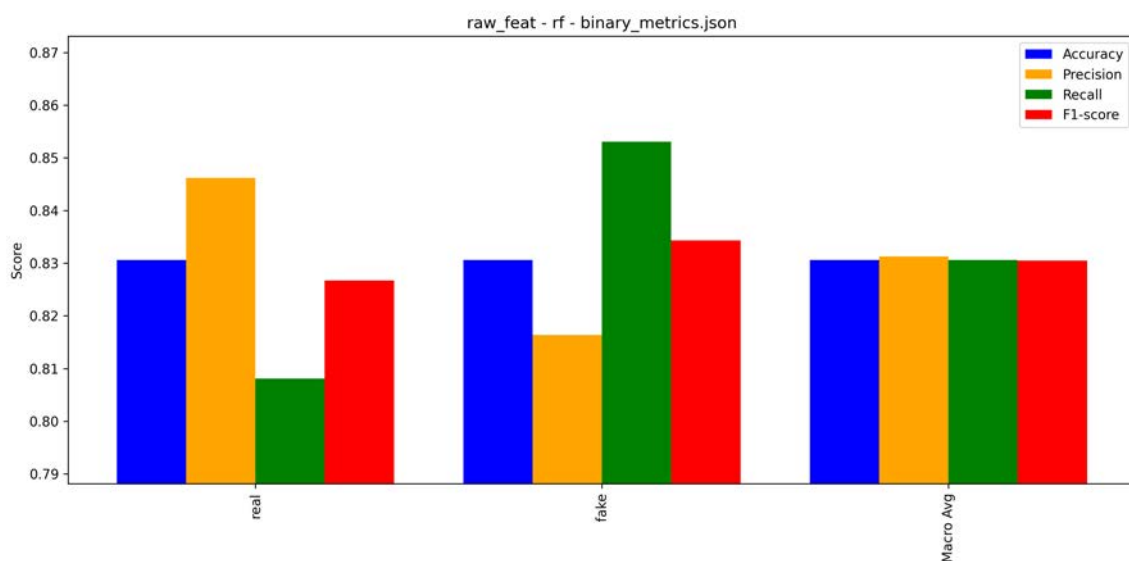
Σχήμα 8.3: Οι μετρικές για το multiclass linear SVM εκπαιδευμένο απευθείας στον χώρο των εικόνων.



Σχήμα 8.4: Το confusion matrix για το multiclass linear SVM.

8.1.3 Χρήση Random Forest για το πρώτο στάδιο του πειράματος

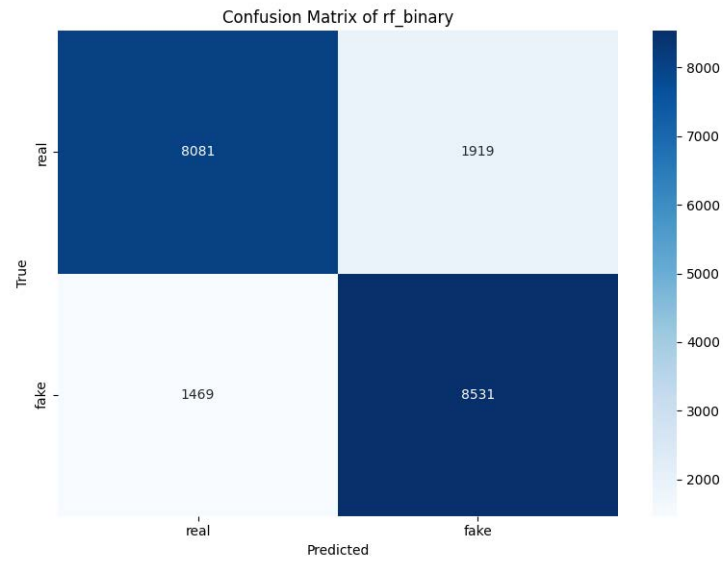
Για την ταξινόμηση χρησιμοποιούνται 2 ταξινομητές Random Forest όπως και στην ενότητα 8.1.2. Από τα σχήματα 8.5 και 8.7 διαπιστώνεται ότι και οι δύο ταξινομητές είναι υποδεέστεροι από το μοντέλο ResNet50Classifier, βέβαια η απόδοσή τους σε σχέση με τους linear SVM ταξινομητές είναι σαφώς ανώτερη. Ο δυαδικός ταξινομητής σημειώνει 83.06% ακρίβεια και μέσο recall, 83.05% μέσο f1 score και 83.13% μέσο precision, ενώ από το confusion matrix του (σχήμα 8.6) είναι φανερό πως και αυτός συναντά περισσότερες δυσκολίες στον εντοπισμό των αληθινών εικόνων. Ο άλλος ταξινομητής έχει αρκετά χαμηλότερη απόδοση επιτυγχάνοντας ποσοστό 55.69% στην ακρίβεια και μέσο recall, περίπου 55.7% μέσο precision και περίπου 55.61% μέσο f1 score. Επίσης, με βάση το σχήμα 8.8 παρατηρείται πως δυσκολεύεται κυρίως στις κλάσεις cat, dog, horse. Επομένως, είναι προτιμότερη λύση από το linear SVM αλλά κατώτερη από το νευρωνικό δίκτυο που αναπτύχθηκε.



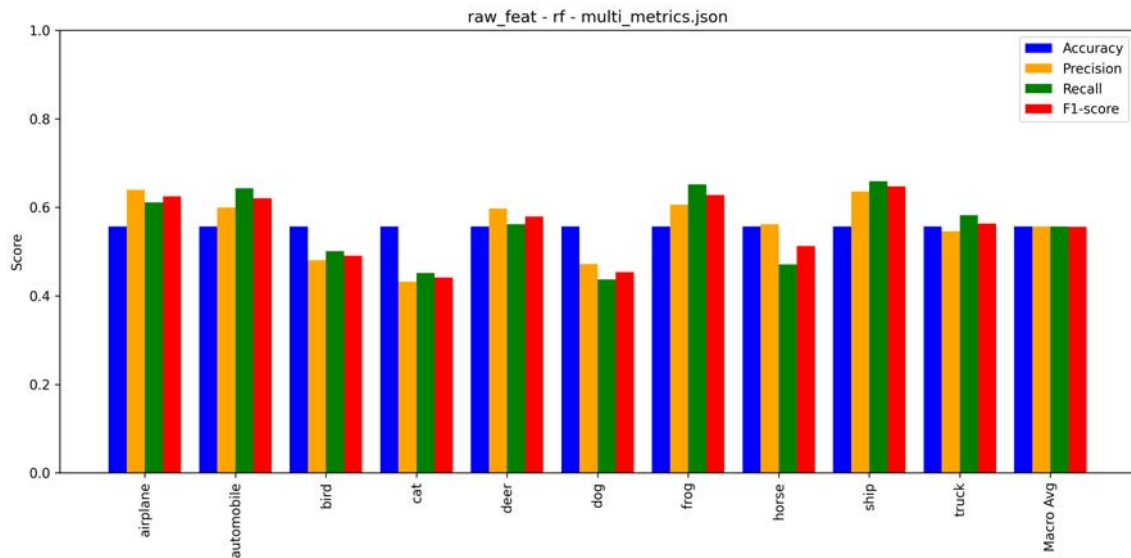
Σχήμα 8.5: Οι μετρικές για το binary Random Forest εκπαιδευμένο απευθείας στον χώρο των εικόνων.

8.1.4 Χρήση XGBoost για το πρώτο στάδιο του πειράματος

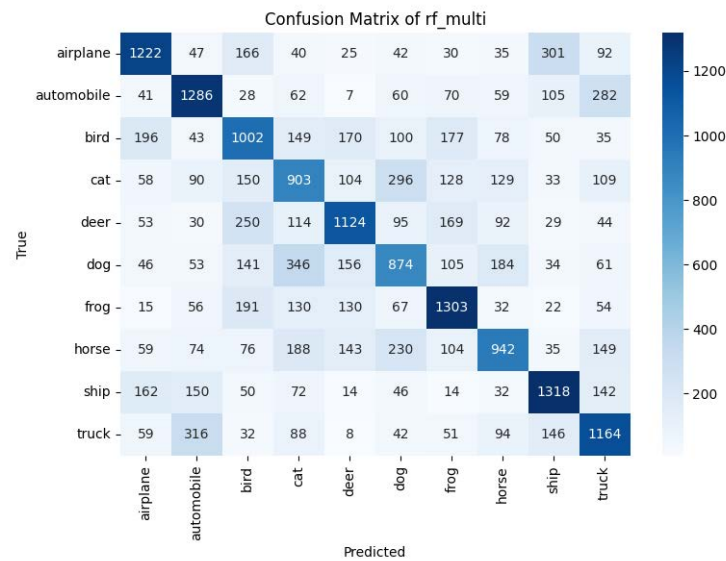
Για τις ανάγκες ταξινόμησης χρησιμοποιούνται 2 μοντέλα XGBoost, όπως και στις προηγούμενες περιπτώσεις. Από τα σχήματα 8.9 και 8.11 επιβεβαιώνεται και πάλι η ανώτερη ικανότητα ταξινόμησης του μοντέλου ResNet50Classifier σε σύγκριση με τις παραδοσιακές τεχνικές ML. Ο δυαδικός ταξινομητής σημείωσε ποσοστά 81.69% στις μετρικές ακρίβεια,



Σχήμα 8.6: Το confusion matrix για το binary Random Forest.

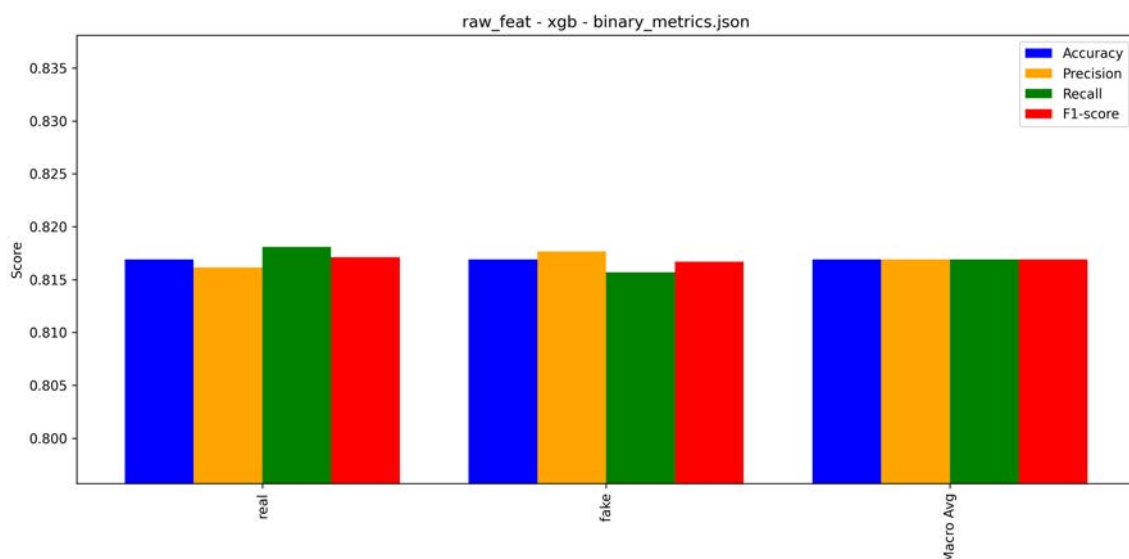


Σχήμα 8.7: Οι μετρικές για το multiclass Random Forest εκπαιδευμένο απευθείας στον χώρο των εικόνων.

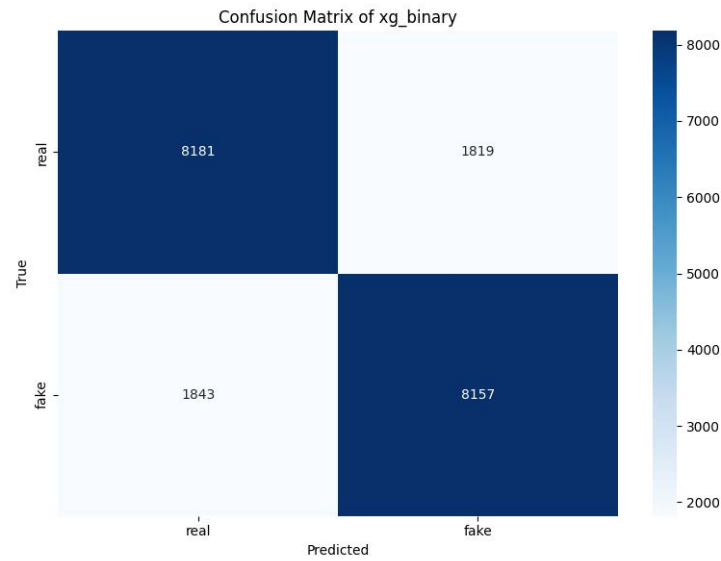


Σχήμα 8.8: Το confusion matrix για το multiclass Random Forest.

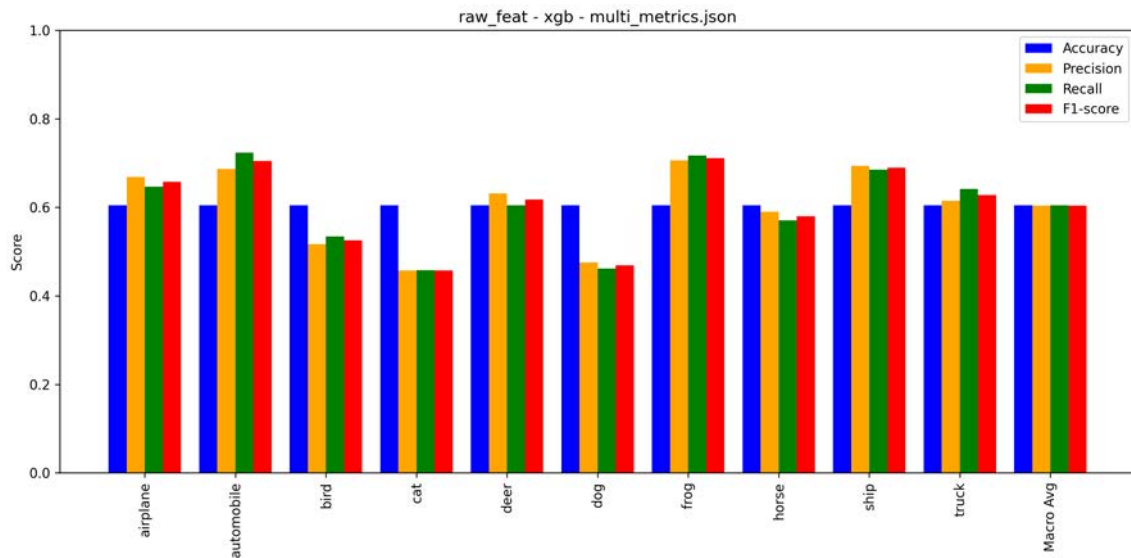
μέσο recall, μέσο precision και μέσο f1 score, ενώ ο άλλος ταξινομητής σημείωσε ποσοστά 60.4% στις αντίστοιχες μετρικές. Επιπρόσθετα, ο δυαδικός ταξινομητής σε αντίθεση με τον Linear SVM και τον Random Forest ουσιαστικά δεν αντιμετώπισε μεγαλύτερη δυσκολία σε καμία κάποια κλάση (σχήμα 8.10). Αντιθέτως, ο ταξινομητής 10 κλάσεων δυσκολεύτηκε στις κλάσεις cat και dog (σχήμα 8.12).



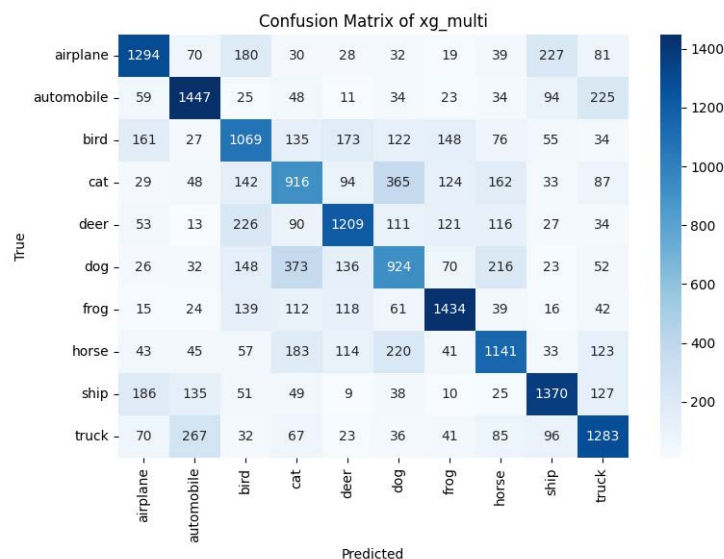
Σχήμα 8.9: Οι μετρικές για το binary XGBoost εκπαιδευμένο απευθείας στον χώρο των ει-
κόνων.



Σχήμα 8.10: To confusion matrix για το binary XGBoost.



Σχήμα 8.11: Οι μετρικές για το multiclass XGBoost εκπαιδευμένο απευθείας στον χώρο των εικόνων.



Σχήμα 8.12: Το confusion matrix για το multiclass XGBoost.

8.2 Χρήση χαρακτηριστικών απευθείας από το CNN για εκπαίδευση ML μοντέλων

Σε αυτή την ενότητα θα αναπτυχθούν τα υπόλοιπα οκτώ μοντέλα μηχανικής μάθησης. Αναπτύσσονται έξι μοντέλα αντίστοιχα με την προηγούμενη ενότητα 8.1, ένας δυαδικός ταξινομητής SVM με πυρήνα RBF και ένας ταξινομητής SVM δέκα κλάσεων με πυρήνα RBF. Η μόνη διαφορά με το προηγούμενο στάδιο είναι πως τα μοντέλα εκπαιδεύτηκαν σε χαρακτηριστικά που εξάχθηκαν από το μοντέλο ResNet50Classifier.

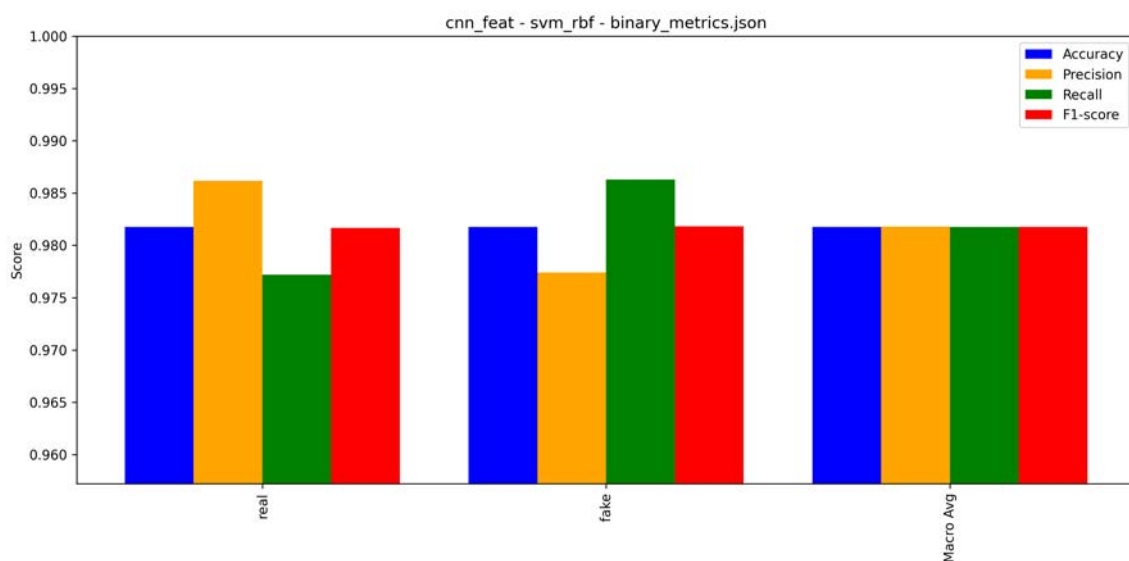
8.2.1 Εξαγωγή χαρακτηριστικών

Αρχικά, επιλέγεται να εξαχθούν τα διανύσματα χαρακτηριστικών από το μοντέλο έτσι όπως διαμορφώνονται μετά την έξοδό τους από το `final_layer`. Επιλέχθηκε το συγκεκριμένο στρώμα, διότι είναι το τελευταίο πριν περάσουν τα χαρακτηριστικά στις κεφαλές και υπολογιστούν τα τελικά αποτελέσματα. Η διαδικασία εξαγωγής χαρακτηριστικών γίνεται στη συνάρτηση `feature_extraction_ML`, η οποία δέχεται ως ορίσματα το μοντέλο και το `dataloader`. Στη συνάρτηση αυτή πραγματοποιείται μια μικρή εκδοχή της ρουτίνας δοκιμής του μοντέλου, αφού το μοντέλο τίθεται σε κατάσταση αξιολόγησης, απενεργοποιείται η οπισθοδιδόση και εκτελείται η προώθηση της εισόδου έως και το τέλος του `final_layer`. Έπειτα, διανυσματοποιείται ο πίνακας χαρακτηριστικών και αυτό το διάνυσμα μήκους 512 είναι το τελικό

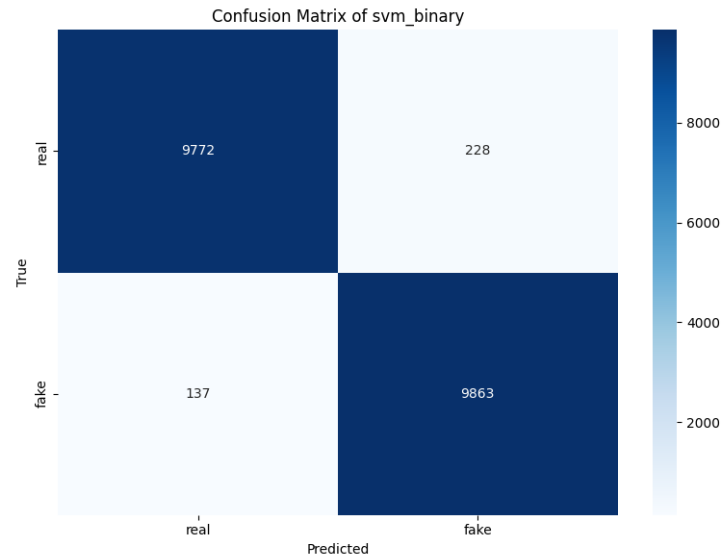
διάνυσμα χαρακτηριστικών για μια εικόνα. Ταυτόχρονα, αποσπώνται και οι δύο ετικέτες της εικόνας. Η συνάρτηση `feature_extraction_ML` καλείται και για τα δεδομένα εκπαίδευσης και για τα δεδομένα δοκιμής. Η υλοποίηση της συνάρτησης `feature_extraction_ML` παρουσιάζεται στο σχήμα B.25. Έπειτα, γίνεται κανονικοποίηση διανυσμάτων, αλλά δεν εφαρμόζεται μείωση διαστασιμότητας, καθώς τα διανύσματα έχουν ήδη επιθυμητό μέγεθος. Στη συνέχεια, ακολουθεί η εκπαίδευση και ο έλεγχος των ML μοντέλων. Η διαδικασία εκπαίδευσης και δοκιμής για τα ML μοντέλα έχει περιγραφεί στην ενότητα 8.1 και το μόνο που αλλάζει είναι τα διανύσματα χαρακτηριστικών.

8.2.2 RBF kernel SVM

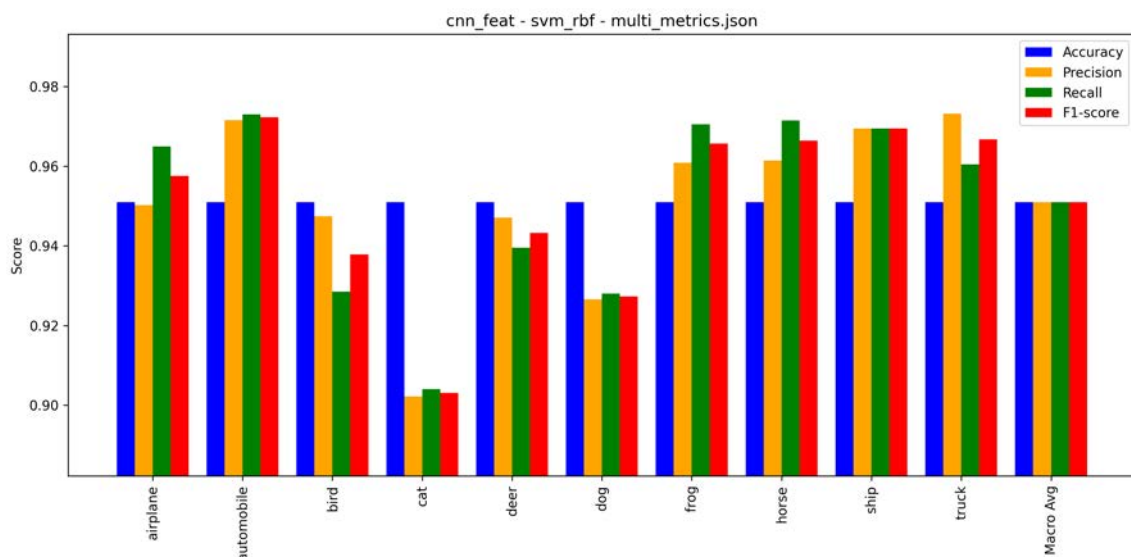
Σε αυτή την ενότητα εξετάζεται η απόδοση του SVM με συνάρτηση πυρήνα RBF που εκπαιδεύτηκε στον χώρο χαρακτηριστικών του μοντέλου. Οι ταξινομητές RBF SVM σημειώνουν πολύ υψηλά ποσοστά σωστής ταξινόμησης και είναι παρόμοιοι στην απόδοση με το CNN. Ο δυαδικός ταξινομητής καταγράφει ποσοστά 98.18% στην ακρίβεια και στο μέσο precision και 97.17% στο μέσο recall και μέσο f1 score (σχήμα 8.13), ενώ ο άλλος ταξινομητής επιτυγχάνει 95.1% σε όλες τις αντίστοιχες μετρικές. Επίσης, δεν δυσκολεύεται ιδιαίτερα να διακρίνει τις 10 κλάσεις μεταξύ τους, καθώς σε όλες σημειώνει ποσοστά σωστής ταξινόμησης άνω του 90% (σχήμα 8.15). Τέλος, στα σχήματα 8.14 και 8.16 παρουσιάζονται αναλυτικά οι πίνακες σύγχυσης του μοντέλου.



Σχήμα 8.13: Οι μετρικές για το binary RBF SVM εκπαιδευμένο σε ResNet50Classifier features.



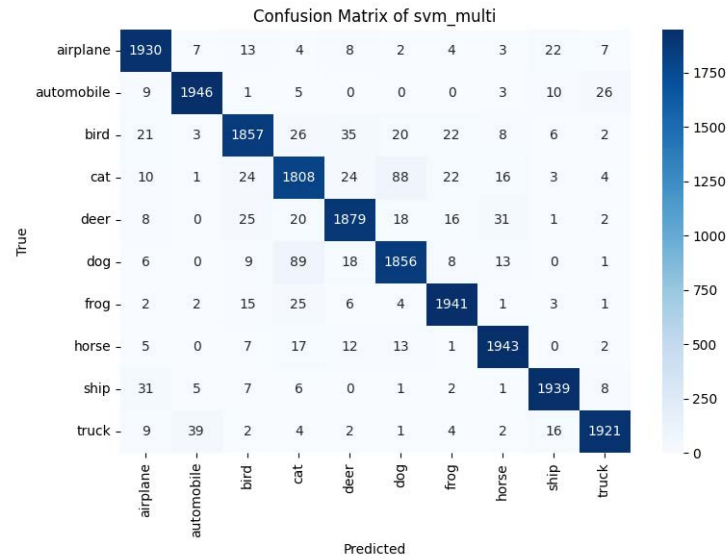
Σχήμα 8.14: Το confusion matrix για το binary RBF SVM.



Σχήμα 8.15: Οι μετρικές για το multiclass RBF SVM εκπαιδευμένο σε ResNet50Classifier features.

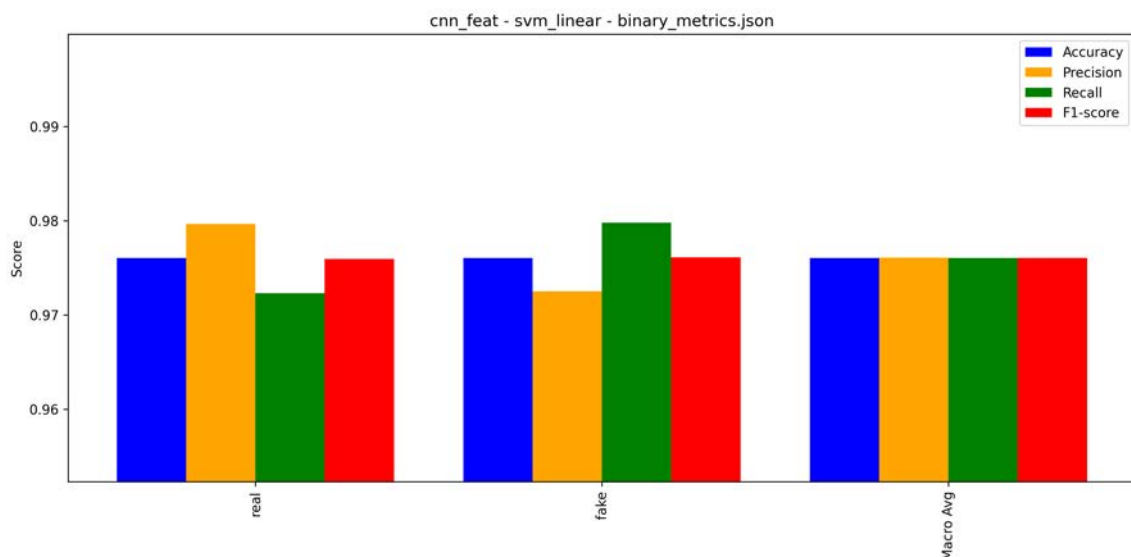
8.2.3 Linear SVM

Σε αυτή την ενότητα εξετάζεται η απόδοση του linear SVM που εκπαιδεύτηκε στον χώρο χαρακτηριστικών του μοντέλου. Οι ταξινομητές Linear SVM που εκπαιδεύτηκαν σε χαρακτηριστικά εικόνων που εξάχθηκαν από το CNN έχουν αποτελέσματα συγκρίσιμα με αυτά του CNN, ελάχιστα όμως χειρότερα. Ο δυαδικός ταξινομητής καταγράφει περίπου 97.61%

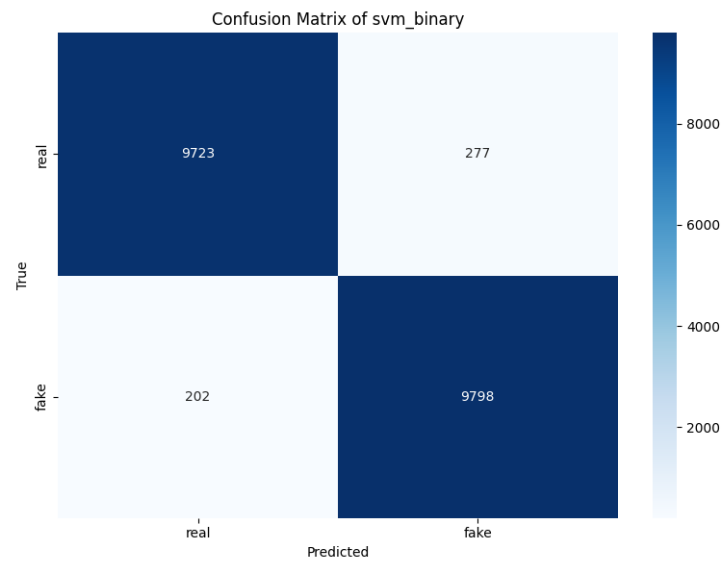


Σχήμα 8.16: Το confusion matrix για το RBF SVM.

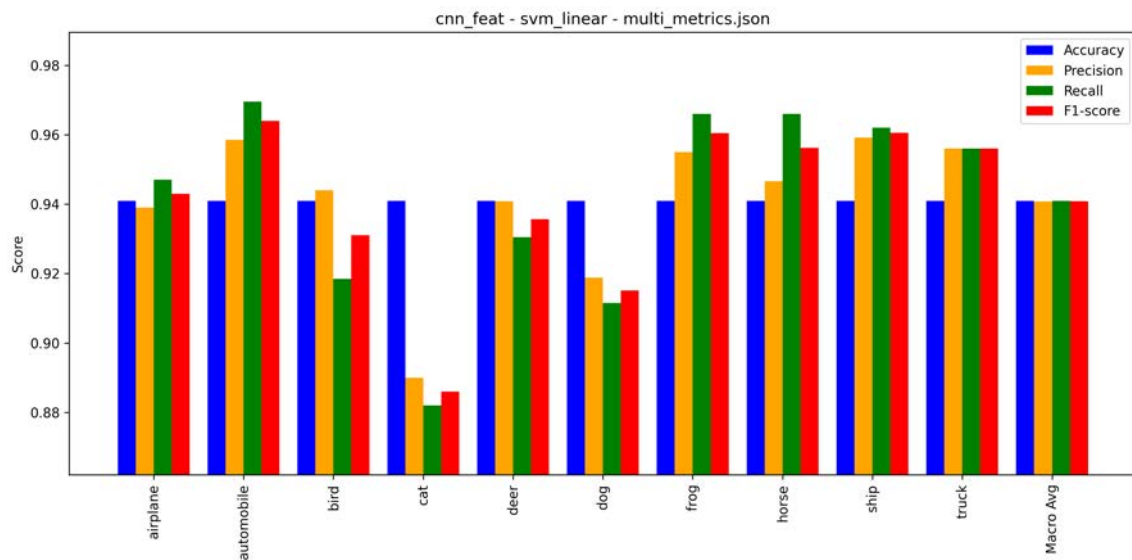
ακρίβεια, μέσο recall και μέσο precision και περίπου 97.60% μέσο f1 score (σχήμα 8.17), ενώ ο άλλος ταξινομητής καταγράφει 94.09% ακρίβεια, 94.08% μέσο precision, μέσο recall, μέσο f1 score (σχήμα 8.19). Τέλος, στα σχήματα 8.18 και 8.20 παρουσιάζονται αναλυτικά οι πίνακες σύγχυσης του μοντέλου.



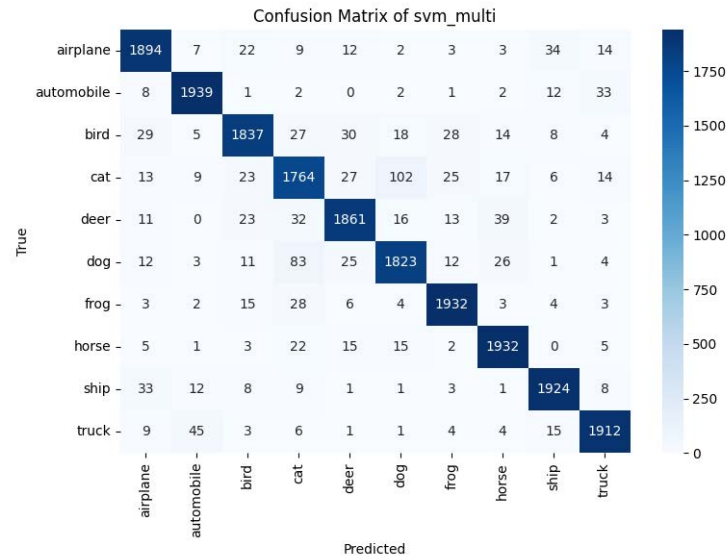
Σχήμα 8.17: Οι μετρικές για το binary Linear SVM εκπαιδευμένο σε ResNet50Classifier features.



Σχήμα 8.18: Το confusion matrix για το binary Linear SVM.



Σχήμα 8.19: Οι μετρικές για το multiclass Linear SVM εκπαιδευμένο σε ResNet50Classifier features.



Σχήμα 8.20: Το confusion matrix για το Linear SVM.

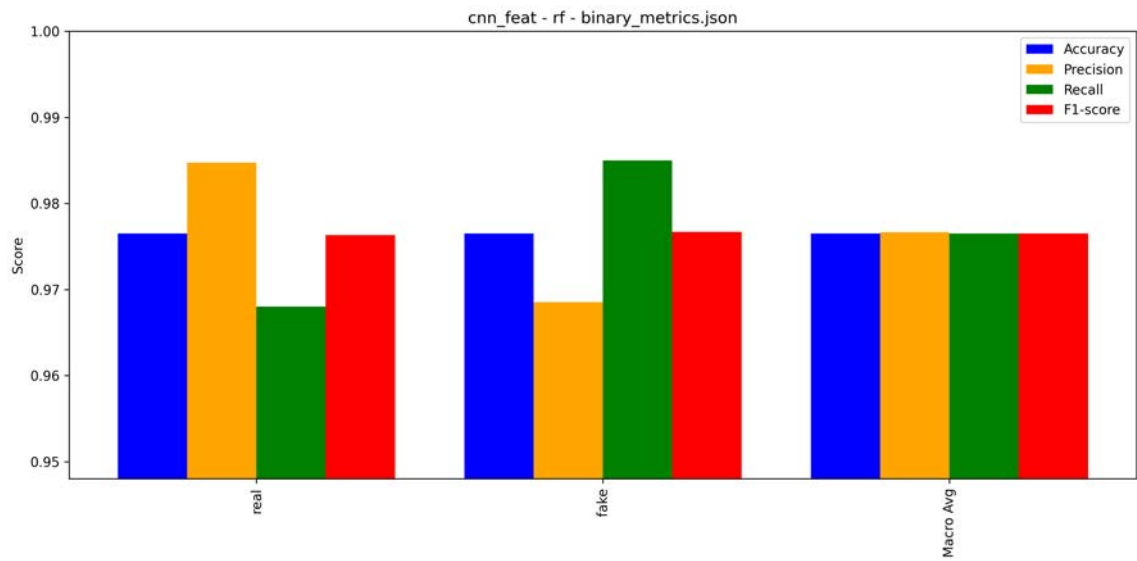
8.2.4 Random Forest Classifier

Σε αυτή την ενότητα εξετάζεται η απόδοση του Random Forest ταξινομητή που εκπαιδεύτηκε στον χώρο χαρακτηριστικών του μοντέλου. Οι ταξινομητές Random Forest σημειώνουν επίσης υψηλά ποσοστά στις μετρικές, παρόμοια με αυτά του SVM με γραμμικό πυρήνα. Ο ταξινομητής 10 κλάσεων έχει ακρίβεια, μέσο recall, μέσο f1 score 94.45% και μέσο precision 94.46% (σχήμα 8.23). Ο ταξινομητής 2 κλάσεων έχει 97.65% ακρίβεια, μέσο recall, μέσο f1 score, 97.66% μέσο precision (σχήμα 8.21). Τέλος, στα σχήματα 8.22 και 8.24 παρουσιάζονται αναλυτικά οι πίνακες σύγχυσης του μοντέλου.

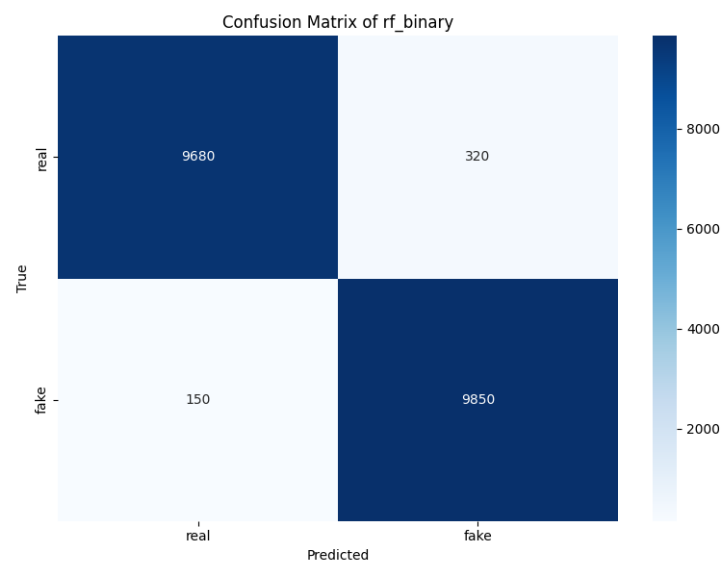
8.2.5 XGBoost Classifier

Σε αυτή την ενότητα εξετάζεται η απόδοση του XGBoost ταξινομητή που εκπαιδεύτηκε στον χώρο χαρακτηριστικών του μοντέλου. Οι ταξινομητές XGBoost καταγράφουν, επίσης, υψηλές επιδόσεις. Ο δυαδικός ταξινομητής σημειώνει ποσοστά 97.94% στην ακρίβεια, μέσο precision, μέσο recall, μέσο f1 score (σχήμα 8.25) και ο άλλος ταξινομητής επιτυγχάνει 94.51% ακρίβεια, μέσο recall, μέσο f1 score και 94.52% μέσο precision (σχήμα 8.27). Τέλος, στα σχήματα 8.26 και 8.28 παρουσιάζονται αναλυτικά οι πίνακες σύγχυσης του μοντέλου.

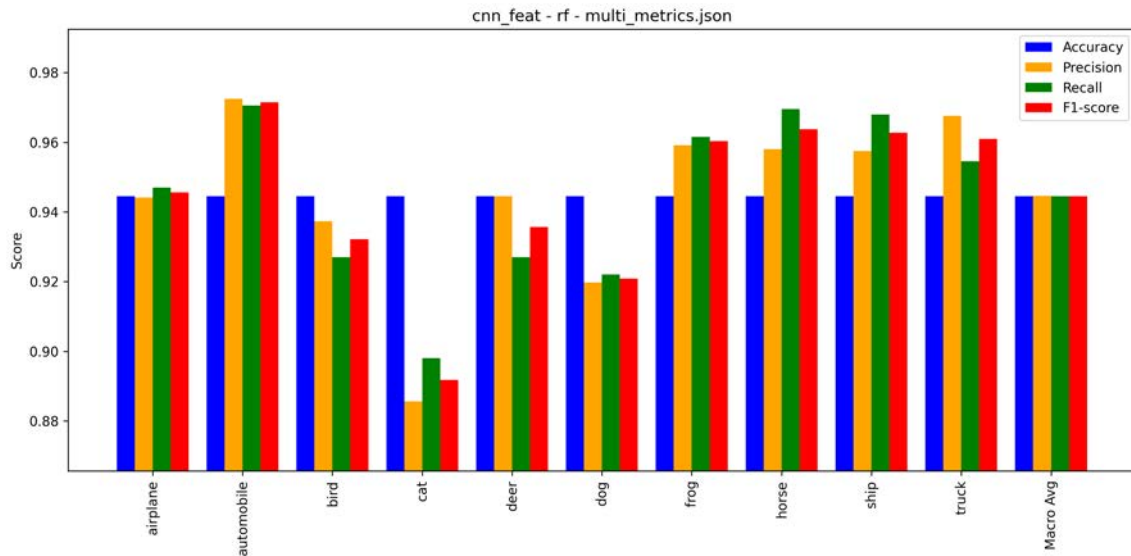
Συνολικά, παρατηρείται πως τα μοντέλα που εκπαιδεύτηκαν σε χαρακτηριστικά προερχόμενα από το νευρωνικό δίκτυο είναι πολύ πιο αποδοτικά από τα αντίστοιχα μοντέλα ML, που εκπαιδεύτηκαν απευθείας στα διανύσματα χαρακτηριστικών των εικόνων. Αυτό είναι



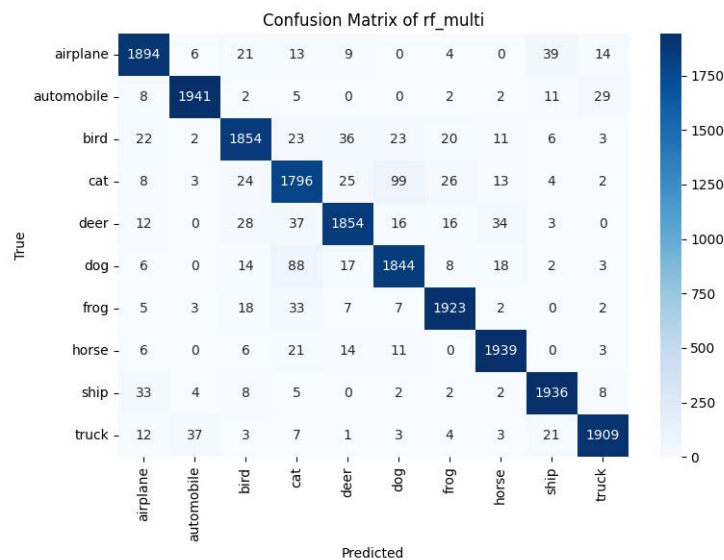
Σχήμα 8.21: Οι μετρικές για το binary Random Forest εκπαιδευμένο σε ResNet50Classifier features.



Σχήμα 8.22: Το confusion matrix για το binary Random Forest.

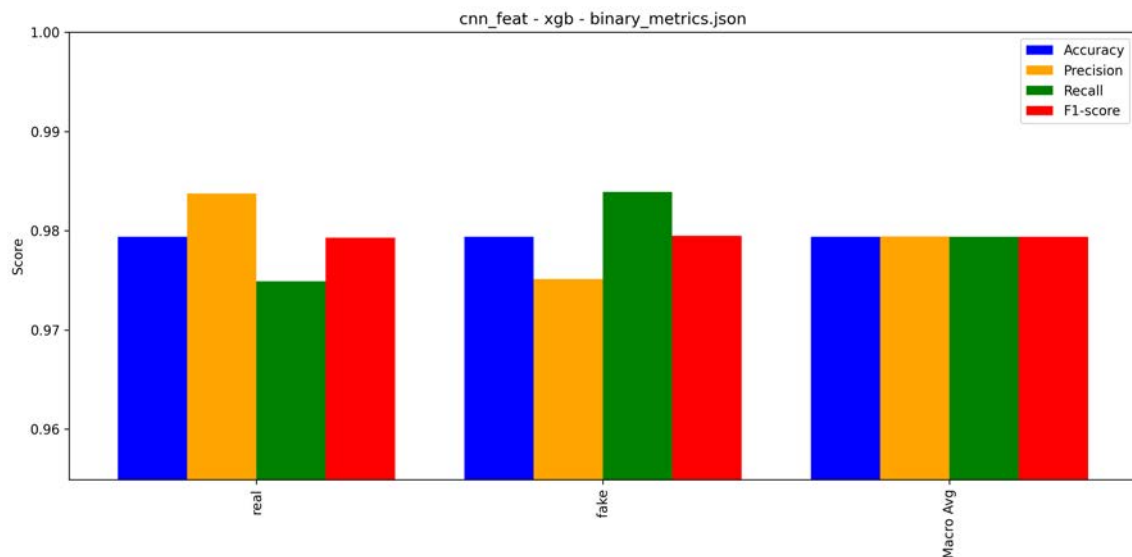


Σχήμα 8.23: Οι μετρικές για το multiclass Random Forest εκπαιδευμένο σε ResNet50Classifier features.

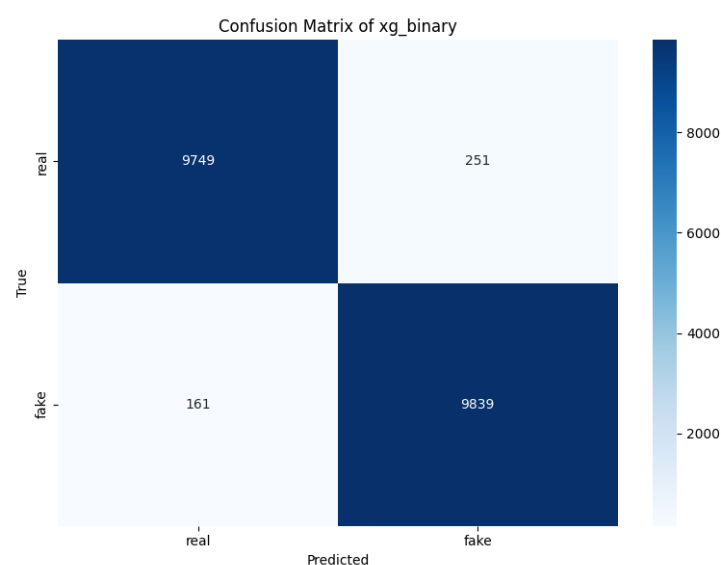


Σχήμα 8.24: To confusion matrix για το Random Forest.

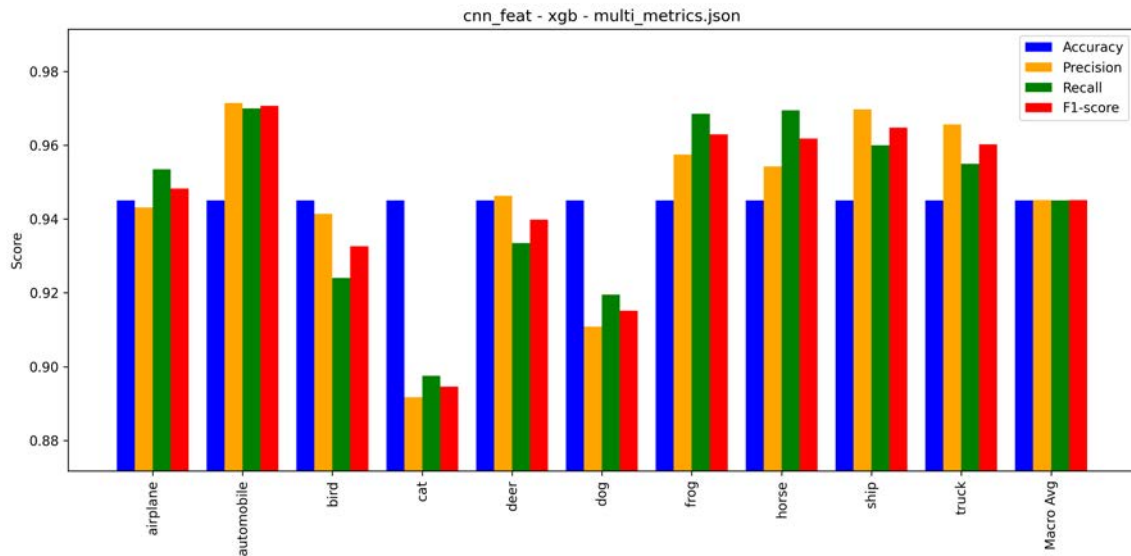
αναμενόμενο, αφού στο στρώμα `final_layer` του μοντέλου έχουν ήδη σχηματιστεί σε μεγάλο βαθμό οι τελικές προβλέψεις. Επίσης, παρατηρείται πως τα ML μοντέλα επωφελούνται από την ύπαρξη ενός νευρωνικού δικτύου, χωρίς βέβαια να υπερέχουν σε ικανότητα ταξινόμησης. Τα μοντέλα αυτά, λοιπόν, θα μπορούσαν να διαδραματίζουν βοηθητικό ρόλο σε εφαρμογές που η ταχύτητα έχει μεγάλη σημασία, καθώς είναι πιο γρήγορα στην ταξινόμηση από το νευρωνικό δίκτυο που αναπτύχθηκε.



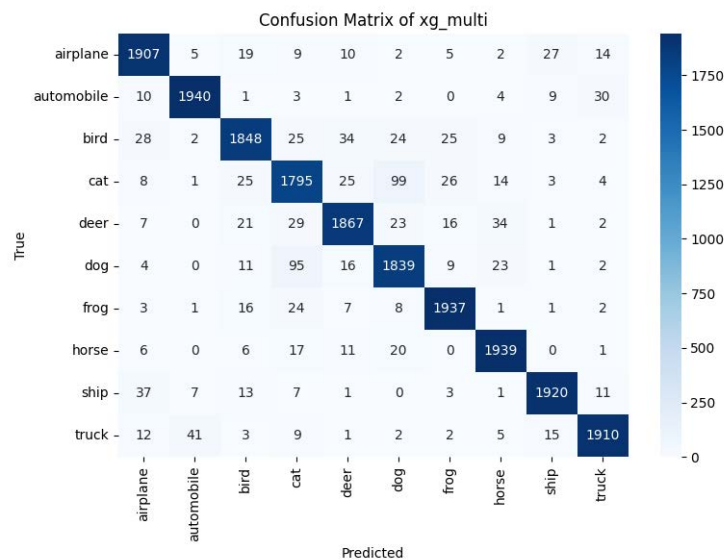
Σχήμα 8.25: Οι μετρικές για το binary XGBoost εκπαιδευμένο σε ResNet50Classifier features.



Σχήμα 8.26: Το confusion matrix για το binary XGBoost.



Σχήμα 8.27: Οι μετρικές για το multiclass XGBoost εκπαιδευμένο σε ResNet50Classifier features.



Σχήμα 8.28: To confusion matrix για το XGBoost.

Κεφάλαιο 9

Συμπεράσματα

9.1 Σύνοψη και συμπεράσματα

Αυτή η διπλωματική διερεύνησε την ικανότητα των συνελκτικών νευρωνικών δικτύων να ταξινομούν εικόνες με βάση το μέσο προέλευσής τους, κάμερα ή γεννήτρια τεχνητής νοημοσύνης, και το θέμα απεικόνισης. Πέτυχε την ανάπτυξη ενός CNN συστήματος ταξινόμησης δύο κεφαλών που βασίζεται στο μοντέλο ResNet-50 [3] σημειώνοντας 93.45% κοινή ακρίβεια, 98.19% ακρίβεια στο πρόβλημα δυαδικής ταξινόμησης σε αληθινές ή ψεύτικες εικόνες και 95.06% ακρίβεια στο πρόβλημα ταξινόμησης στις δέκα κλάσεις θέματος απεικόνισης. Απέδειξε πως το σύστημα ταξινόμησης μπορεί να είναι ανθεκτικό σε παραμορφώσεις εικόνων, εάν εκπαιδευτεί σε κατάλληλα δεδομένα, μέσα από την εκπαίδευση και δοκιμή του μοντέλου σε διάφορες συνθήκες θορύβου. Τέλος, απέδειξε πως το ResNet50Classifier είναι ανώτερο από κλασσικές μεθόδους μηχανικής μάθησης και ότι μπορεί να χρησιμοποιηθεί σε συνεργασία με αυτές σε υβριδικά συστήματα, ως μέσο εξαγωγής χαρακτηριστικών αυξάνοντας την ταχύτητα προβλέψεων και διαφυλάσσοντας την ακρίβεια του συστήματος.

9.2 Μελλοντικές επεκτάσεις

Υπάρχει περιθώριο για αρκετή έρευνα στο μέλλον με σκοπό την βελτίωση των συστημάτων ανίχνευσης εικόνων τεχνητής νοημοσύνης και ταξινόμησης περιεχομένου των εικόνων. Η δοκιμή σε πολλαπλές βάσεις δεδομένων με εικόνες από διαφορετικές αρχιτεκτονικές Gen-AI και αληθινές εικόνες περισσότερων κλάσεων, μπορεί να διερευνήσει την δυνατότητα δημιουργίας ενός καθολικού ανιχνευτή που είναι ικανός να αναγνωρίσει την προέλευση της

εικόνας ανεξαρτήτως της αρχιτεκτονικής, καθώς και το περιεχόμενο της απεικόνισης. Επίσης, η ανάπτυξη ενός υβριδικού συστήματος με τεχνικές βαθιάς και μηχανικής μάθησης μπορεί να αποτελέσει την βάση μιας εφαρμογής κυβερνοασφάλειας που έχει ως στόχο τον γρήγορο και αξιόπιστο εντοπισμό εικόνων Gen-AI, που κακόβουλα διαδίδουν ψευδείς πληροφορίες και χρησιμοποιούνται σε επιθέσεις ηλεκτρονικού ψαρέματος (phishing) με στόχο την εξαπάτηση του θύματος και την απόσπαση προσωπικών δεδομένων.

Βιβλιογραφία

- [1] Galamo Monkam, Weifeng Xu, and Jie Yan. A GAN-based approach to detect AI-generated images. In *2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)*, pages 229–232, 2023.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NeurIPS)*, 2017.
- [3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [4] Eri Matsuyama. A deep learning interpretable model for novel coronavirus disease (COVID-19) screening with chest CT images. *Journal of Biomedical Science and Engineering*, 13:140–152, 01 2020.
- [5] Jordan Bird and Ahmad Lotfi. CIFAKE: Image classification and explainable identification of AI-generated synthetic images. *IEEE Access*, PP:1–1, 01 2024.
- [6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 05 2012.
- [7] Weili Fang, Peter E.D. Love, Hanbin Luo, and Lieyun Ding. Computer vision for behaviour-based safety in construction: A review and future directions. *Advanced Engineering Informatics*, 43:100980, 2020.
- [8] Mingjian Zhu, Hanting Chen, Qiangyu Yan, Xudong Huang, Guanyu Lin, Wei Li, Zhi-jun Tu, Hailin Hu, Jie Hu, and Yunhe Wang. GenImage: A million-scale benchmark

- for detecting AI-generated image. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.
- [9] Samah S. Baraheem and Tam V. Nguyen. AI vs. AI: Can AI detect AI-generated images? *Journal of Imaging*, 9(10), 2023.
- [10] Md. Zahid Hossain, Farhad Uz Zaman, and Md. Rakibul Islam. Advancing AI-generated image detection: Enhanced accuracy through CNN and vision transformer models with explainable AI insights. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*, pages 1–6, 2023.
- [11] Shilin Yan, Ouxiang Li, Jiayin Cai, Yanbin Hao, Xiaolong Jiang, Yao Hu, and Weidi Xie. A sanity check for AI-generated image detection. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [12] Maryam Taeb and Hongmei Chi. Comparison of deepfake detection techniques through deep learning. *Journal of Cybersecurity and Privacy*, 2(1):89–106, 2022.
- [13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.
- [14] Adrian Lokner Ladević, Tin Kramberger, Renata Kramberger, and Dino Vlahsek. Detection of AI-generated synthetic images with a lightweight CNN. *AI*, 5(3):1575–1593, 2024.
- [15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
- [16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.
- [17] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

- [18] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, September 2023.
- [19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.
- [20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, 2022.
- [21] Yongtao Yuan. Research and application of transformation-based generative AI technology. *Applied and Computational Engineering*, 111:1–10, 11 2024.
- [22] Efstathios Karypidis, Stylianos Mouslech, Kassiani Skoulariki, and Alexandros Gazis. Comparison analysis of traditional machine learning and deep learning techniques for data and image classification. *WSEAS Transactions on Mathematics*, 21:122–130, 04 2022.
- [23] Xiaowu Sun, Lizhen Liu, Hanshi Wang, Wei Song, and Jingli Lu. Image classification via support vector machine. In *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, volume 01, pages 485–489, 2015.
- [24] Rim Koulali, Hajar Zaidani, and Maryeme Zaim. Image classification approach using machine learning and an industrial hadoop based data pipeline. *Big Data Research*, 24:100184, 2021.
- [25] Yuyang Wang, Yizhi Hao, and Amando Xu Cong. Harnessing machine learning for discerning AI-generated synthetic images. *arXiv preprint arXiv:2401.07358*, 2024.
- [26] Hoang Le, Thai Le, Son Tran, Hai Tran, and Nguyen Thuy. Image classification using support vector machine and artificial neural network. *International Journal of Information Technology and Computer Science*, 4, 05 2012.
- [27] Erhui Xi. Image classification and recognition based on deep learning and random forest algorithm. *Wireless Communications and Mobile Computing*, 2022:1–9, 06 2022.

- [28] Wenjiang Jiao, Xingwei Hao, and Chao Qin. The image classification method with CNN-XGBoost model based on adaptive particle swarm optimization. *Information*, 12(4), 2021.
- [29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.
- [30] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.
- [31] Druthik Sai Chinta, Sindhusa Kamineni, Ratna Pujitha Chatragadda, and Sujatha Kamepalli. Analyzing image classification on AI-generated art vs human created art using deep learning models. In *2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pages 1–6, 2024.
- [32] Qing Lv, Suzhen Zhang, and Yuechun Wang. Deep learning model of image classification using machine learning. *Advances in Multimedia*, 2022(1):3351256, 2022.
- [33] Chitra Desai. Image classification using transfer learning and deep learning. *International Journal of Engineering and Computer Science*, 10, 09 2021.
- [34] Jaya H. Dewan, Rik Das, Sudeep D. Thepade, Harsh Jadhav, Nishant Narsale, Ajinkya Mhasawade, and Sinu Nambiar. Image classification by transfer learning using pre-trained CNN models. In *2023 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI)*, pages 1–6, 2023.
- [35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [36] R. Ahmad Fattah Saskoro, Novanto Yudistira, and Tirana Noor Fatyanosa. Detection of AI-generated images from various generators using gated expert convolutional neural network. *IEEE Access*, 12:147772–147783, 2024.

- [37] Jiaqi Lu. Convolutional neural network and vision transformer for image classification. *Applied and Computational Engineering*, 5:104–108, 06 2023.
- [38] Fernando Martin-Rodriguez, Rocio Garcia-Mojon, and Monica Fernandez-Barciela. Detection of AI-created images using pixel-wise feature extraction and convolutional neural networks. *Sensors*, 23(22), 2023.
- [39] Rimsha Rafique, Rahma Gantassi, Rashid Amin, Jaroslav Frnda, Aida Mustapha, and Asma Alshehri. Deep fake detection and classification using error-level analysis and deep learning. *Scientific Reports*, 13, 05 2023.
- [40] Nan Zhong, Yiran Xu, Sheng Li, Zhenxing Qian, and Xinpeng Zhang. Patchcraft: Exploring texture patch for efficient AI-generated image detection. *arXiv preprint arXiv:2311.12397*, 2023.
- [41] Yuming Chen and Maryam Yashtini. Detecting AI generated images through texture and frequency analysis of patches. In *2024 4th International Conference on Artificial Intelligence, Virtual Reality and Visualization*, pages 103–110, 2024.
- [42] Sinisa Mihajlovic, Dragan Ivetic, and Ivana Berković. Image classification using convolutional neural networks. In *Proceedings of the 10th International Conference on Applied Internet and Information Technologies*, Zrenjanin, Serbia, 2020.
- [43] Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. In Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos, editors, *Machine Learning and Its Applications: Advanced Lectures*, pages 249–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- [44] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.
- [45] 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, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Ad-*

vances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

- [46] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 12 2014.
- [47] Ameen Abd Al-salam Selami and Ahmed Fadhil. A study of the effects of gaussian noise on image features. *Kirkuk University Journal / Scientific Studies (1992-0849)*, 11:152 – 169, 04 2016.
- [48] A. Raid, Wael Khedr, Mohamed El-dosuky, and Wesam Ahmed. Jpeg image compression using discrete cosine transform - a survey. *International Journal of Computer Science & Engineering Survey*, 5, 05 2014.

ΠΑΡΑΡΤΗΜΑΤΑ

Παράρτημα Α

Αποτελέσματα έρευνας αντοχής του μοντέλου σε θόρυβο

Στους πίνακες και στα σχήματα του παραρτήματος ισχύουν οι εξής ονοματολογικές συμβάσεις.

Για το όνομα του μοντέλου:

- “clean_model”: Αφορά το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα χωρίς θόρυβο,
- “full_noise_model”: Αφορά το μοντέλο που εκπαιδεύτηκε σε εικόνες με όλα τα είδη θορύβου,
- “both_noise_model”: Αφορά το μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου,
- “jpeg_model”: Αφορά το μοντέλο που εκπαιδεύτηκε σε εικόνες με jpeg compression,
- “gaussian_model”: Αφορά το μοντέλο που εκπαιδεύτηκε σε εικόνες με κανονικό θόρυβο.

Για την ονομασία της βάσης δεδομένων:

- “clean”: Αφορά την βάση με καθαρά δεδομένα χωρίς θόρυβο,
- “full”: Αφορά την βάση με εικόνες με όλα τα είδη θορύβου,
- “both”: Αφορά την βάση με εικόνες με έως ένα είδος θορύβου,

- “jpeg”: Αφορά την βάση με εικόνες με jpeg compression,
- “gaussian”: Αφορά την βάση με εικόνες με κανονικό θόρυβο.

A.1 Πίνακες αποτελεσμάτων

Πίνακας A.1: Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “clean_model”.

Μετρική		Βάση δεδομένων δοκιμής				
		“clean”	“full”	“both”	“jpeg”	“gaussian”
Accuracy	κεφαλή 1	98.19%	95.76%	95.37%	96.56%	94.78%
	κεφαλή 2	95.06%	89.63%	88.69%	90.98%	86.89%
	κοινό	93.41%	86.09%	84.91%	88.11%	82.66%
Precision	κεφαλή 1	98.19%	95.77%	95.38%	96.57%	94.79%
	κεφαλή 2	95.06%	90.67%	89.99%	91.57%	88.92%
	κοινό	93.38%	87.49%	86.67%	89.01%	85.26%
Recall	κεφαλή 1	98.19%	95.76%	95.37%	96.56%	94.78%
	κεφαλή 2	95.06%	89.64%	88.69%	90.99%	86.89%
	κοινό	93.41%	86.09%	84.91%	88.12%	82.66%
F1 score	κεφαλή 1	98.18%	95.76%	95.37%	96.55%	94.77%
	κεφαλή 2	95.05%	89.86%	88.97%	91.13%	87.30%
	κοινό	93.38%	86.45%	85.35%	88.38%	83.25%

Πίνακας A.2: Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “full_noise_model”.

Μετρική		Βάση δεδομένων δοκιμής				
		“clean”	“full”	“both”	“jpeg”	“gaussian”
Accuracy	κεφαλή 1	97.80%	98.12%	98.19%	98.23%	98.30%
	κεφαλή 2	94.05%	94.28%	94.11%	94.06%	94.37%
	κοινό	92.05%	92.60%	92.49%	92.50%	92.85%
Precision	κεφαλή 1	97.80%	98.12%	98.19%	98.23%	98.30%
	κεφαλή 2	94.07%	94.28%	94.11%	94.07%	94.38%
	κοινό	92.06%	92.59%	92.49%	92.51%	92.85%
Recall	κεφαλή 1	97.79%	98.12%	98.19%	98.23%	98.30%
	κεφαλή 2	94.05%	94.28%	94.11%	94.06%	94.37%
	κοινό	92.05%	92.60%	92.49%	92.49%	92.85%
F1 score	κεφαλή 1	97.79%	98.11%	98.19%	98.23%	98.29%
	κεφαλή 2	94.05%	94.27%	94.09%	94.04%	94.36%
	κοινό	92.03%	92.57%	92.46%	92.47%	92.82%

Πίνακας Α.3: Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “both_noise_model”.

Μετρική		Βάση δεδομένων δοκιμής				
		“clean”	“full”	“both”	“jpeg”	“gaussian”
Accuracy	κεφαλή 1	97.54%	98.22%	98.20%	98.16%	98.42%
	κεφαλή 2	93.92%	94.45%	94.48%	94.54%	94.60%
	κοινό	91.73%	92.82%	92.88%	92.88%	93.17%
Precision	κεφαλή 1	97.55%	98.22%	98.20%	98.16%	98.42%
	κεφαλή 2	93.95%	94.45%	94.47%	94.53%	94.60%
	κοινό	91.71%	92.80%	92.85%	92.84%	93.16%
Recall	κεφαλή 1	97.53%	98.22%	98.20%	98.16%	98.42%
	κεφαλή 2	93.92%	94.45%	94.48%	94.54%	94.59%
	κοινό	91.73%	92.82%	92.88%	92.88%	93.17%
F1 score	κεφαλή 1	97.53%	98.21%	98.20%	98.16%	98.41%
	κεφαλή 2	93.91%	94.44%	94.47%	94.53%	94.59%
	κοινό	91.67%	92.80%	92.85%	92.85%	93.15%

Πίνακας A.4: Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “jpeg_model”.

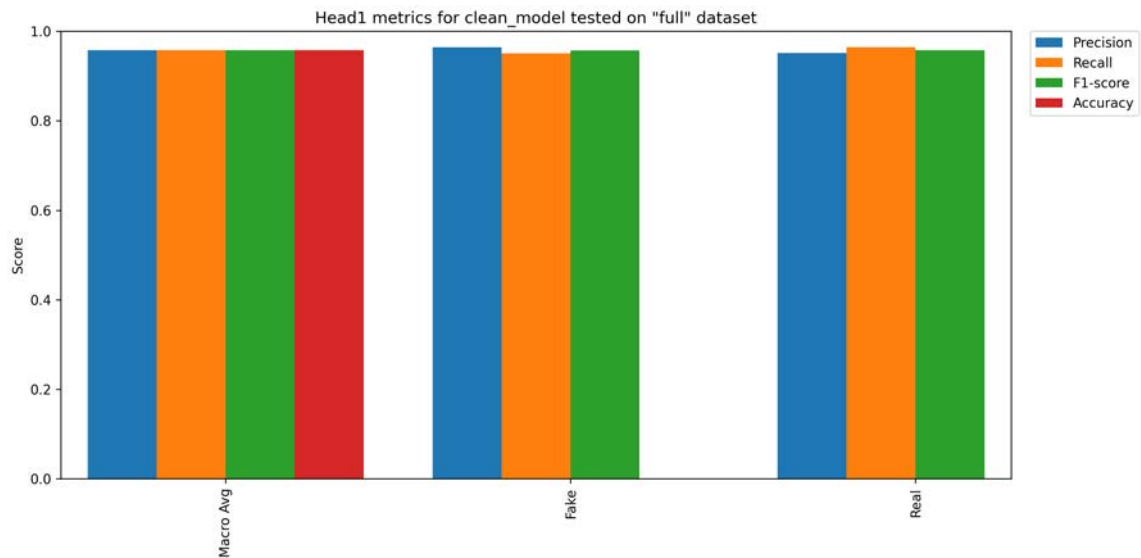
Μετρική		Βάση δεδομένων δοκιμής				
		“clean”	“full”	“both”	“jpeg”	“gaussian”
Accuracy	κεφαλή 1	97.52%	98.18%	98.31%	98.37%	98.39%
	κεφαλή 2	94.53%	94.60%	94.57%	94.61%	94.46%
	κοινό	92.30%	92.97%	93.04%	93.16%	93.05%
Precision	κεφαλή 1	97.56%	98.18%	98.31%	98.37%	98.39%
	κεφαλή 2	94.55%	94.60%	94.57%	94.61%	94.47%
	κοινό	92.28%	92.98%	93.04%	93.16%	93.06%
Recall	κεφαλή 1	97.52%	98.18%	98.31%	98.37%	98.39%
	κεφαλή 2	94.53%	94.59%	94.57%	94.61%	94.46%
	κοινό	92.29%	92.97%	93.04%	93.16%	93.05%
F1 score	κεφαλή 1	97.51%	98.17%	98.31%	98.37%	98.39%
	κεφαλή 2	94.53%	94.60%	94.56%	94.61%	94.46%
	κοινό	92.24%	92.97%	93.03%	93.15%	93.04%

Πίνακας Α.5: Μέσος όρος των κοινών μετρικών και κεφαλών για accuracy, recall, precision, f1 score για όλες τις δοκιμές με τις διαφορετικές βάσεις δεδομένων στο μοντέλο “gaussian_model”.

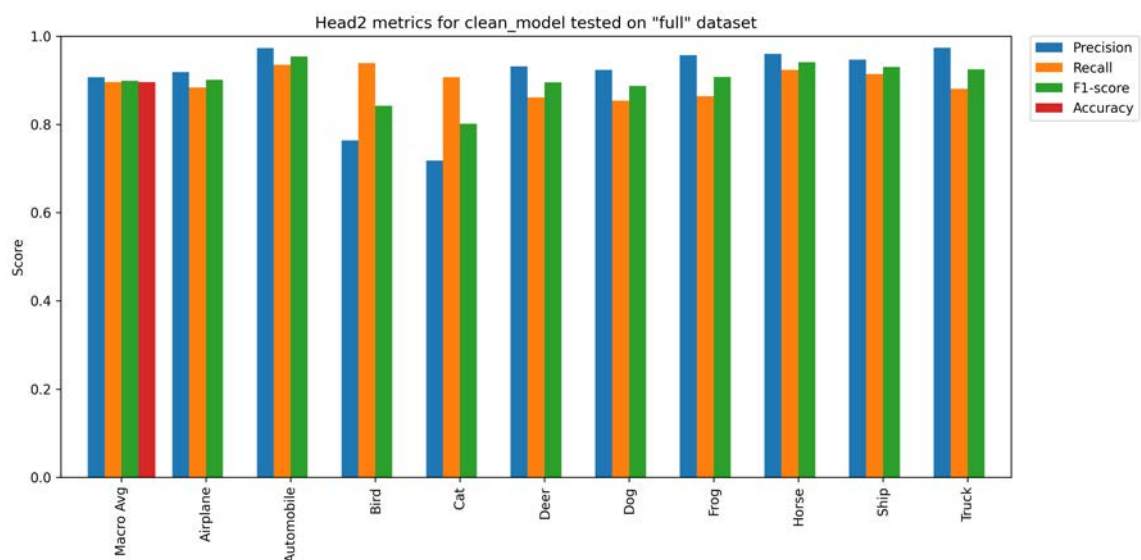
Μετρική		Βάση δεδομένων δοκιμής				
		“clean”	“full”	“both”	“jpeg”	“gaussian”
Accuracy	κεφαλή 1	76.91%	93.81%	96.74%	95.33%	98.43%
	κεφαλή 2	76.73%	90.75%	93.50%	91.38%	95.11%
	κοινό	59.73%	85.94%	90.67%	87.39%	93.64%
Precision	κεφαλή 1	83.35%	94.17%	96.78%	95.47%	98.43%
	κεφαλή 2	81.90%	91.24%	93.59%	91.76%	95.10%
	κοινό	71.43%	87.30%	90.96%	88.26%	93.63%
Recall	κεφαλή 1	76.91%	93.81%	96.74%	95.52%	98.43%
	κεφαλή 2	76.73%	90.75%	93.50%	91.38%	95.10%
	κοινό	59.73%	85.93%	90.67%	87.39%	93.64%
F1 score	κεφαλή 1	75.73%	93.80%	96.74%	95.32%	98.43%
	κεφαλή 2	77.61%	90.87%	93.52%	91.47%	95.10%
	κοινό	60.14%	86.30%	90.75%	87.60%	93.63%

A.2 Γραφήματα έρευνας αντοχής στον θόρυβο

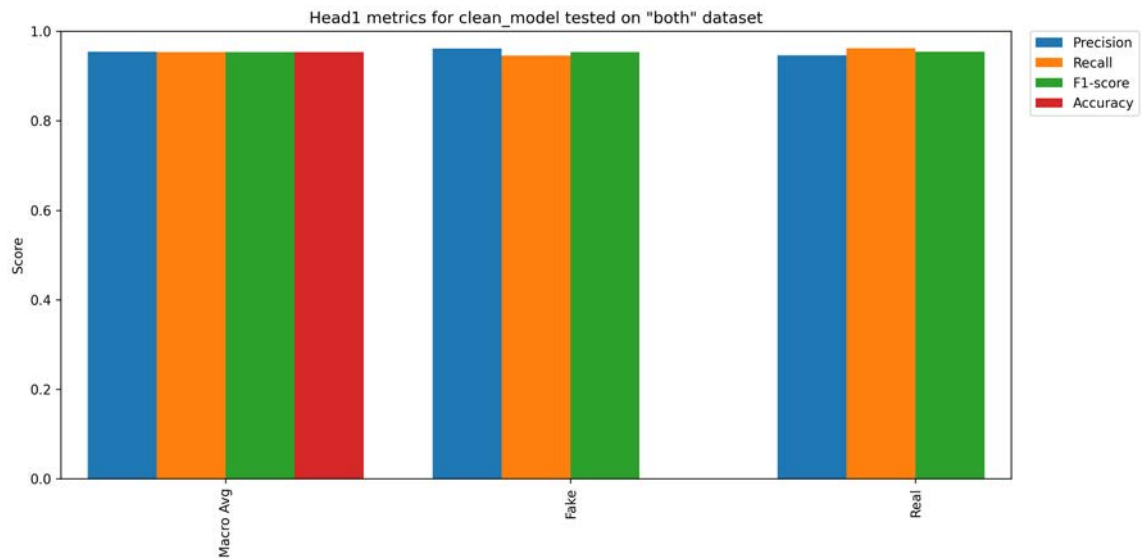
A.2.1 Γραφήματα μοντέλου “clean_model”



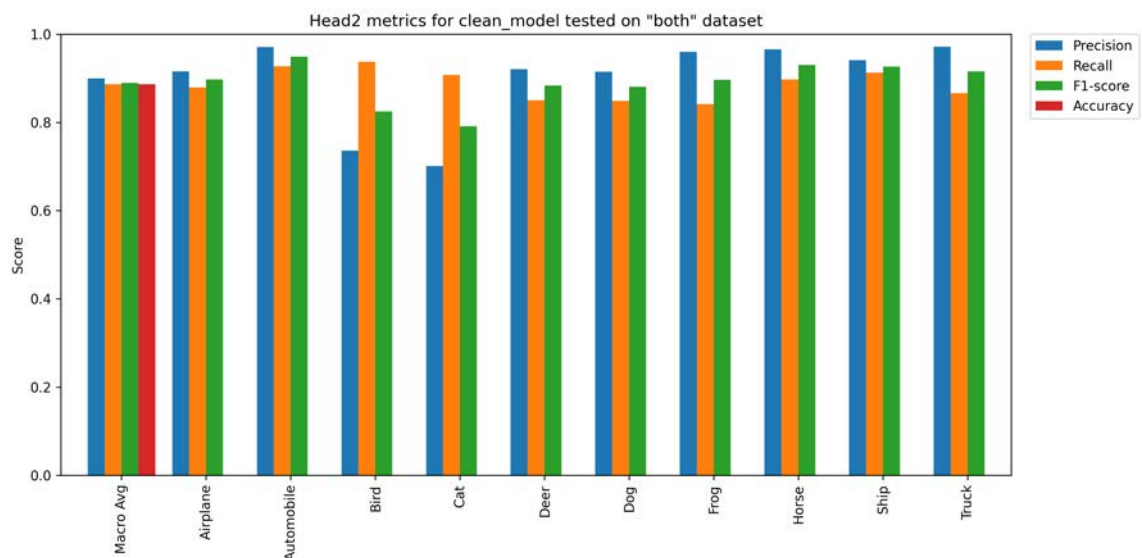
Σχήμα A.1: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου



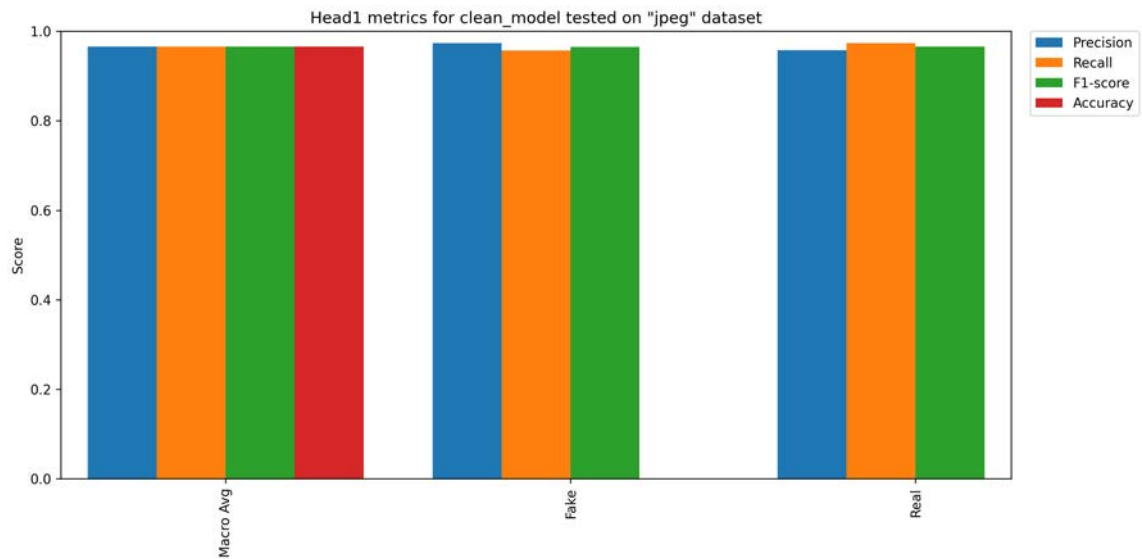
Σχήμα A.2: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου



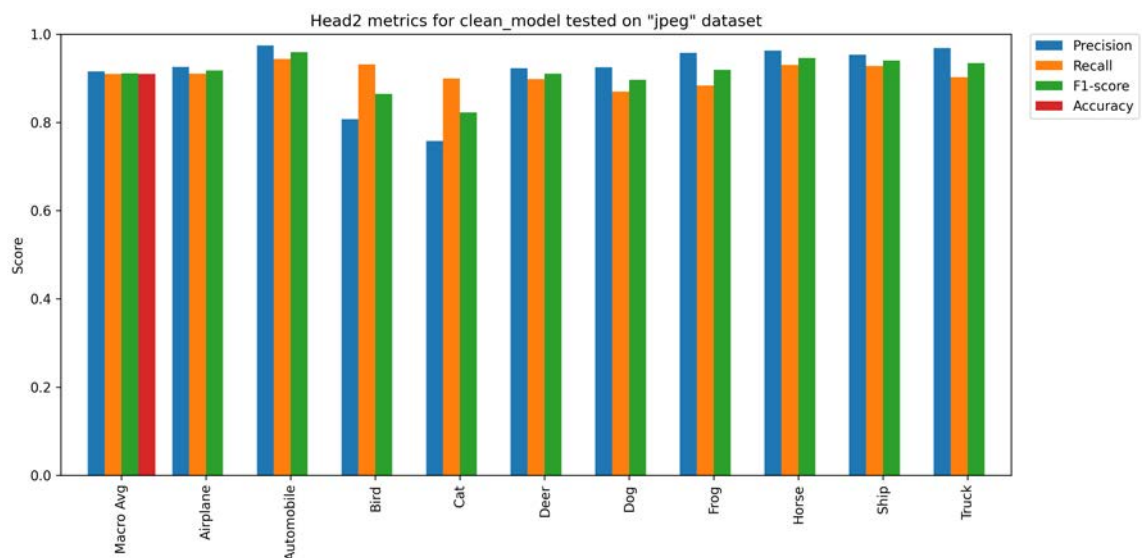
Σχήμα Α.3: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου, χωρίς να συνυπάρχουν σε εικόνες και τα δύο είδη θορύβου



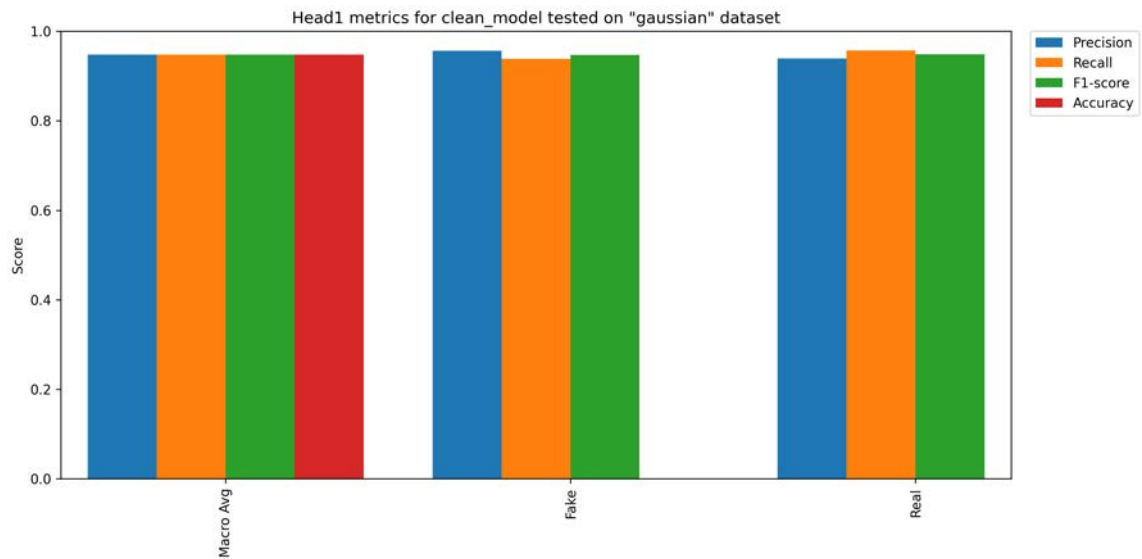
Σχήμα Α.4: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου, χωρίς να συνυπάρχουν σε εικόνες και τα δύο είδη θορύβου



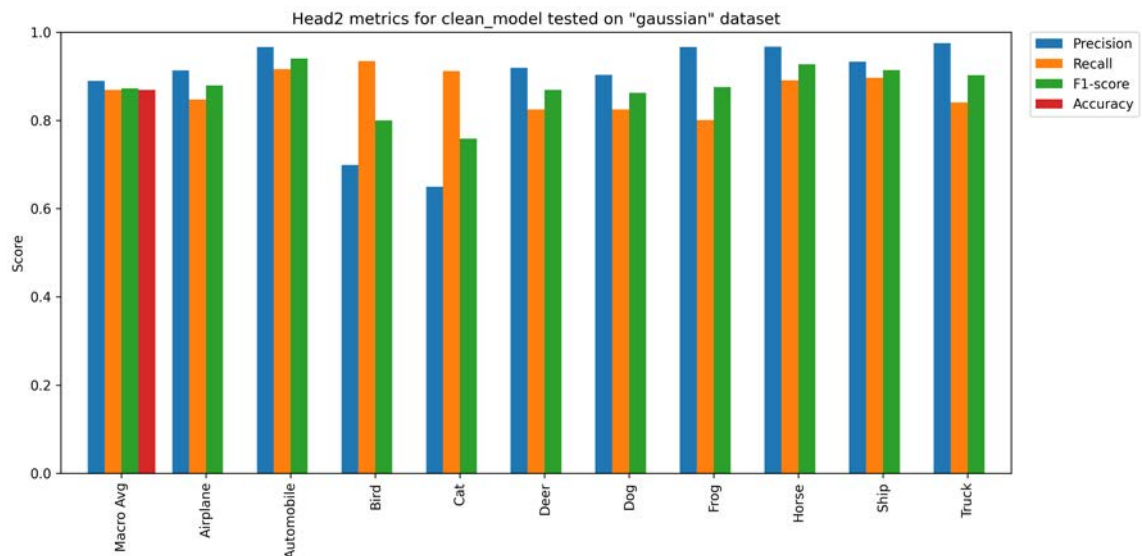
Σχήμα A.5: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.



Σχήμα A.6: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.

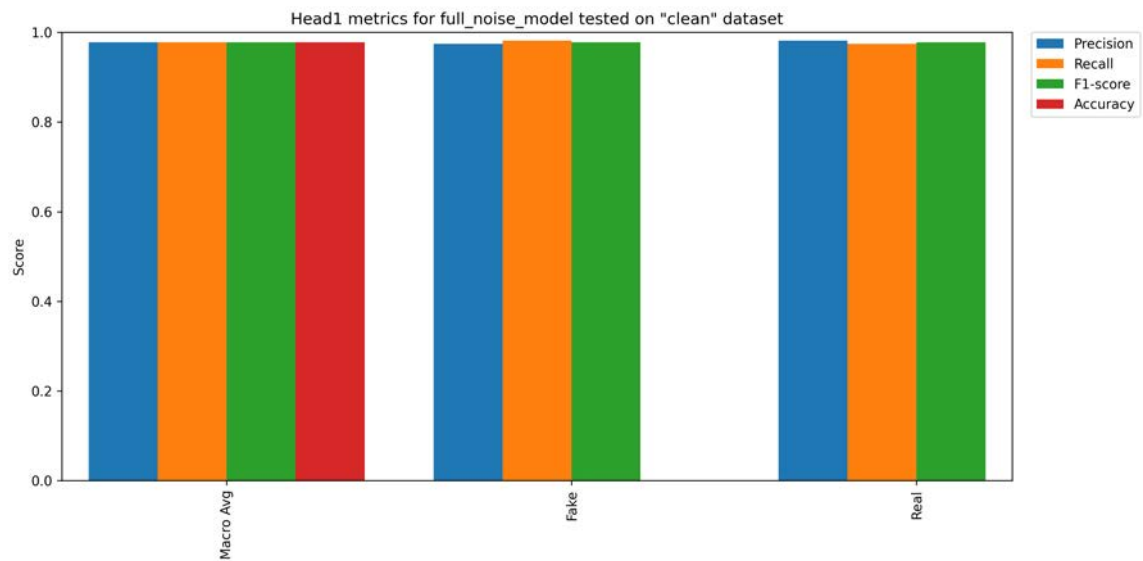


Σχήμα Α.7: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

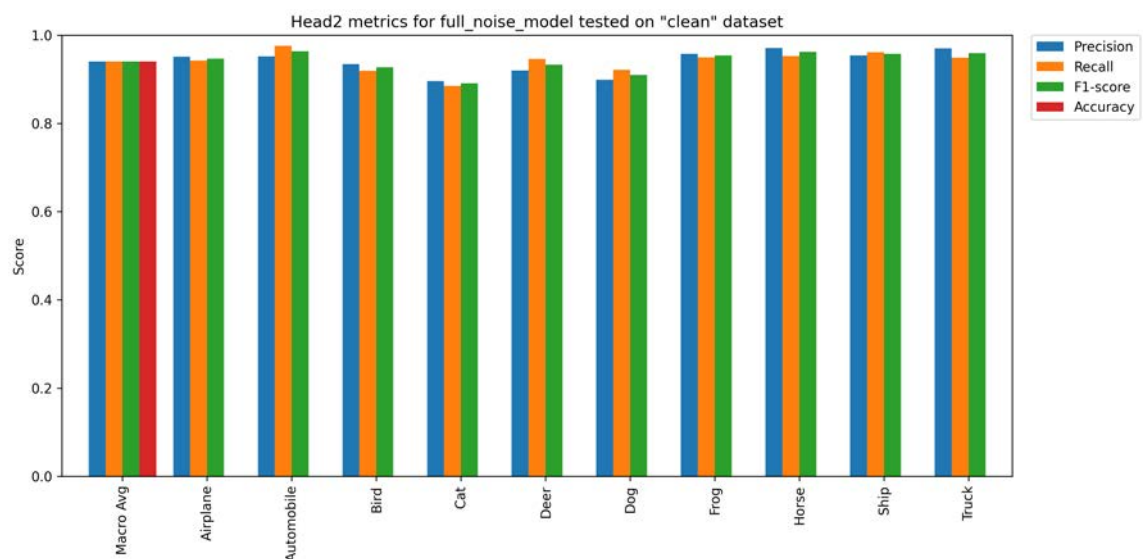


Σχήμα Α.8: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε καθαρά δεδομένα και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

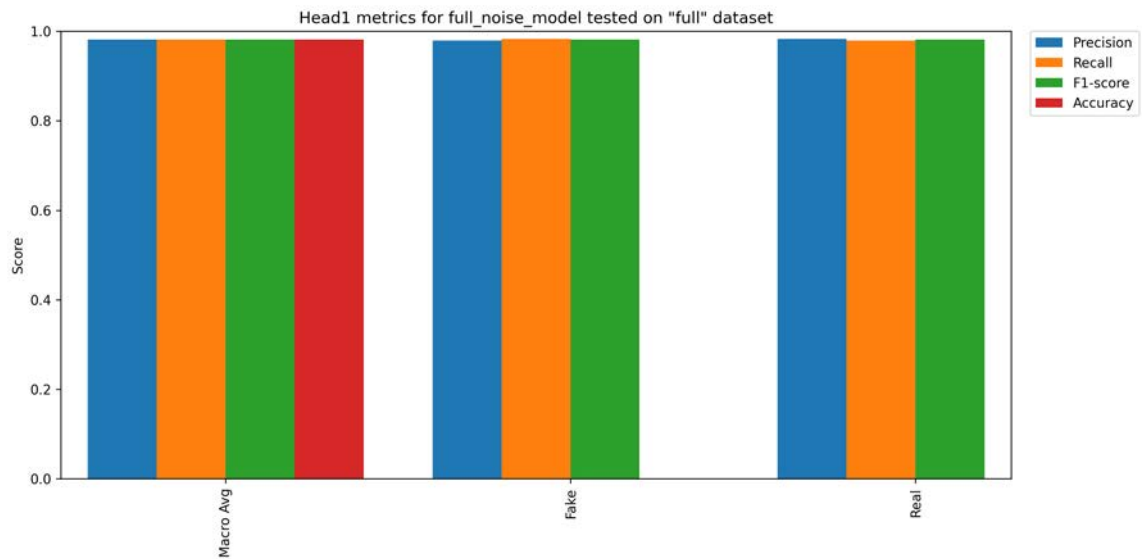
A.2.2 Γραφήματα μοντέλου “full_noise_model”



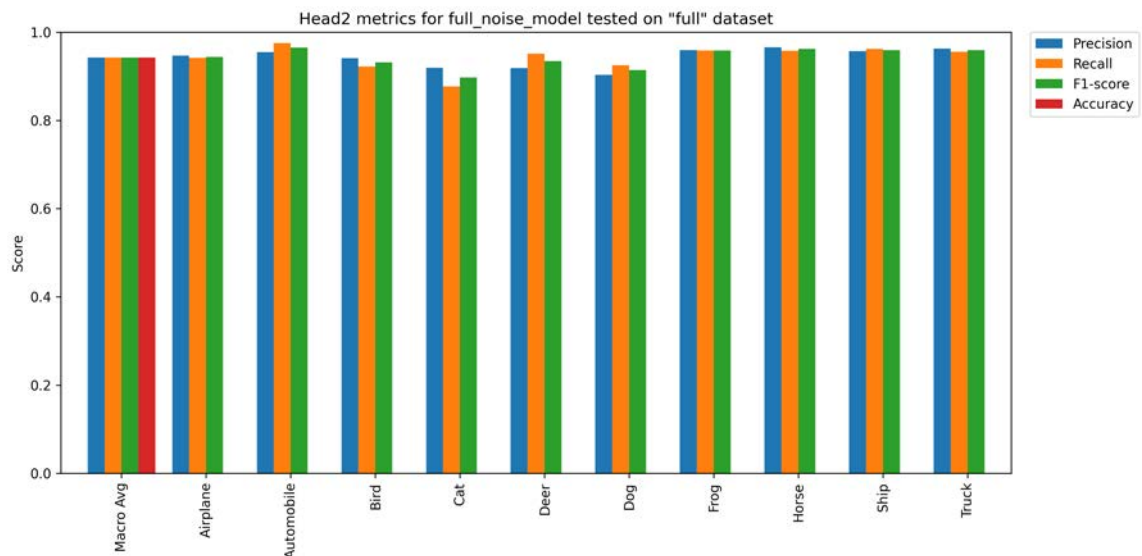
Σχήμα A.9: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε καθαρά δεδομένα.



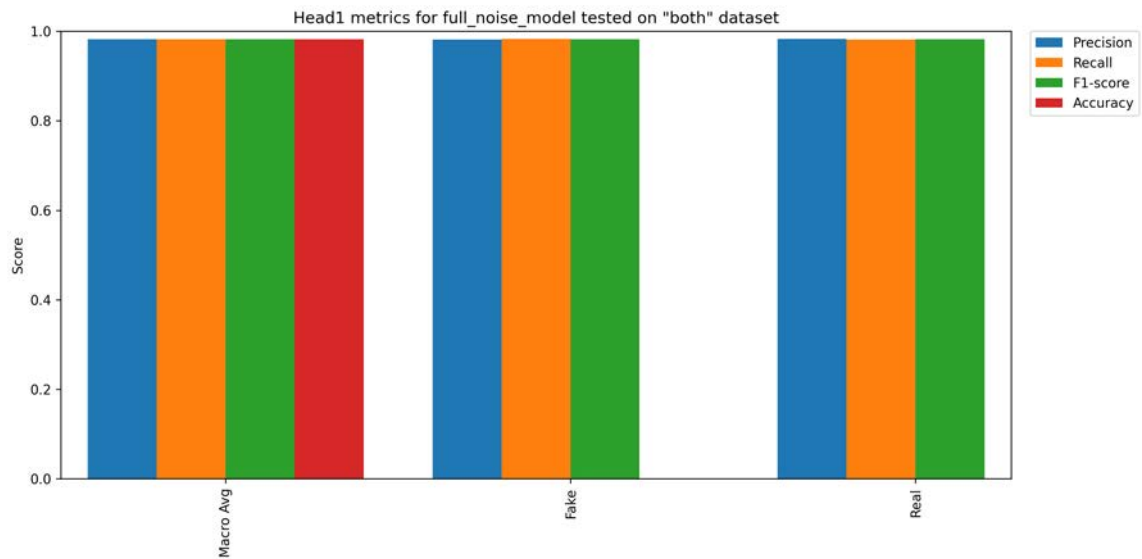
Σχήμα A.10: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε καθαρά δεδομένα.



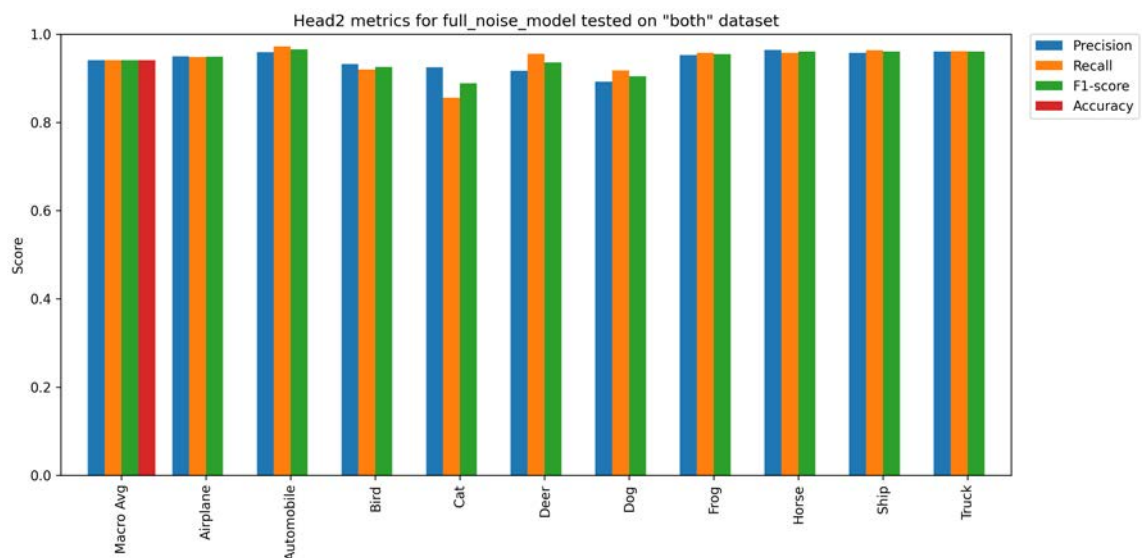
Σχήμα Α.11: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα.



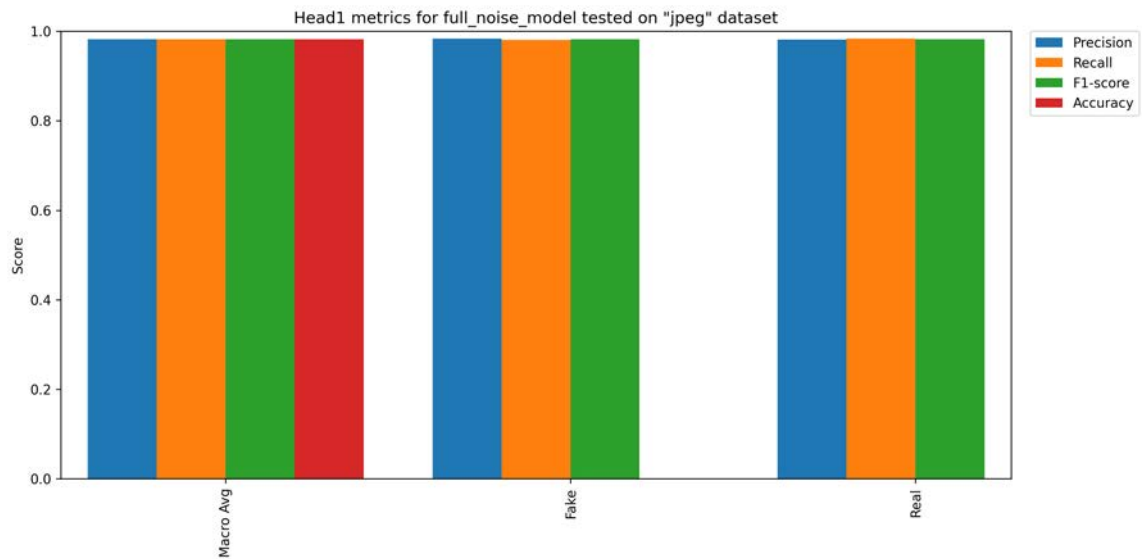
Σχήμα Α.12: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα



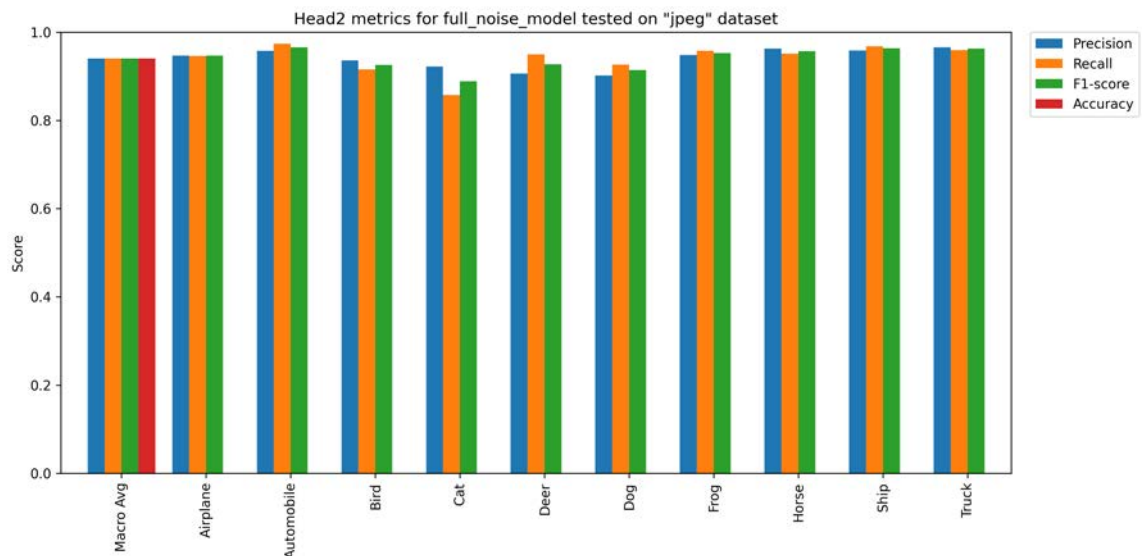
Σχήμα A.13: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες που έχουν έως ένα είδος θορύβου.



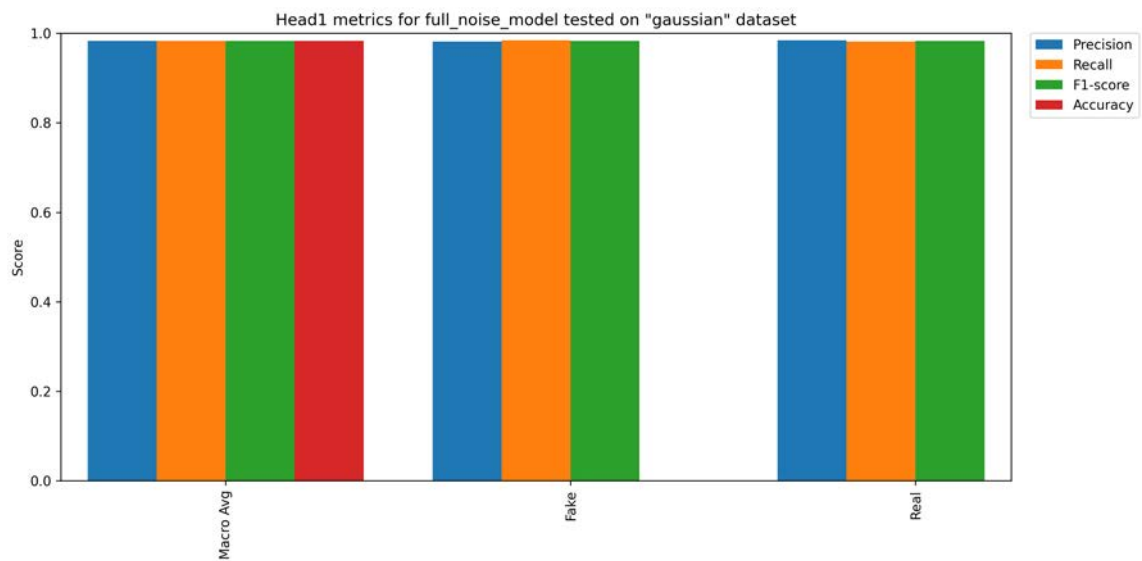
Σχήμα A.14: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες που έχουν έως ένα είδος θορύβου



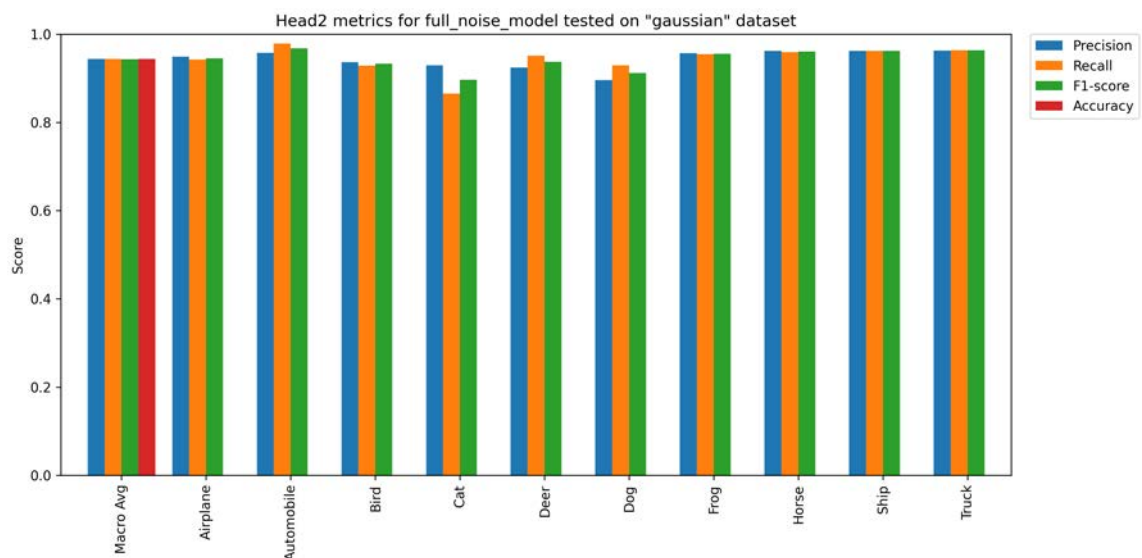
Σχήμα Α.15: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.



Σχήμα Α.16: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.

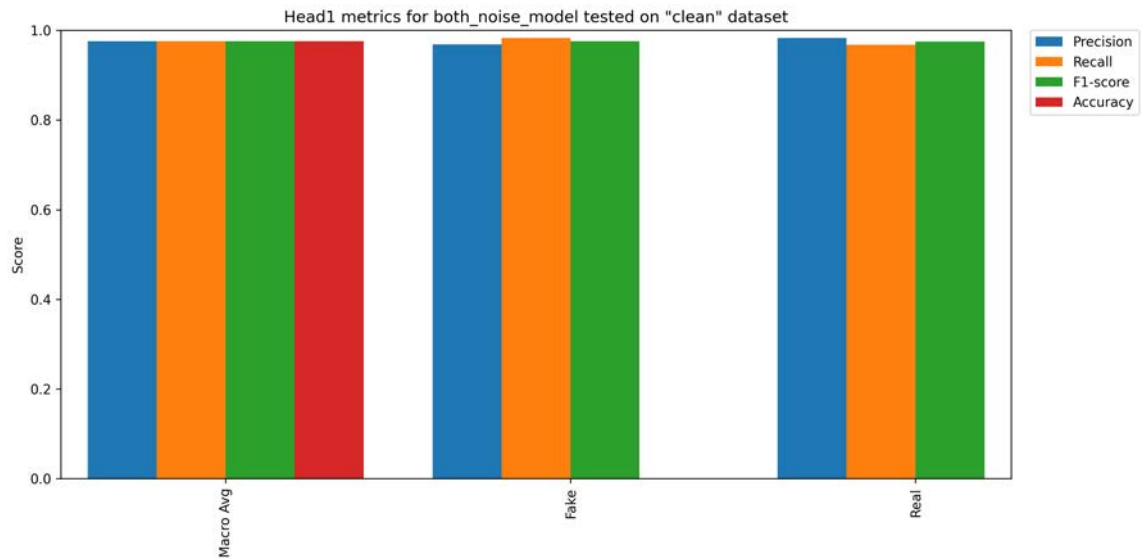


Σχήμα A.17: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με κανονικό θόρυβο.

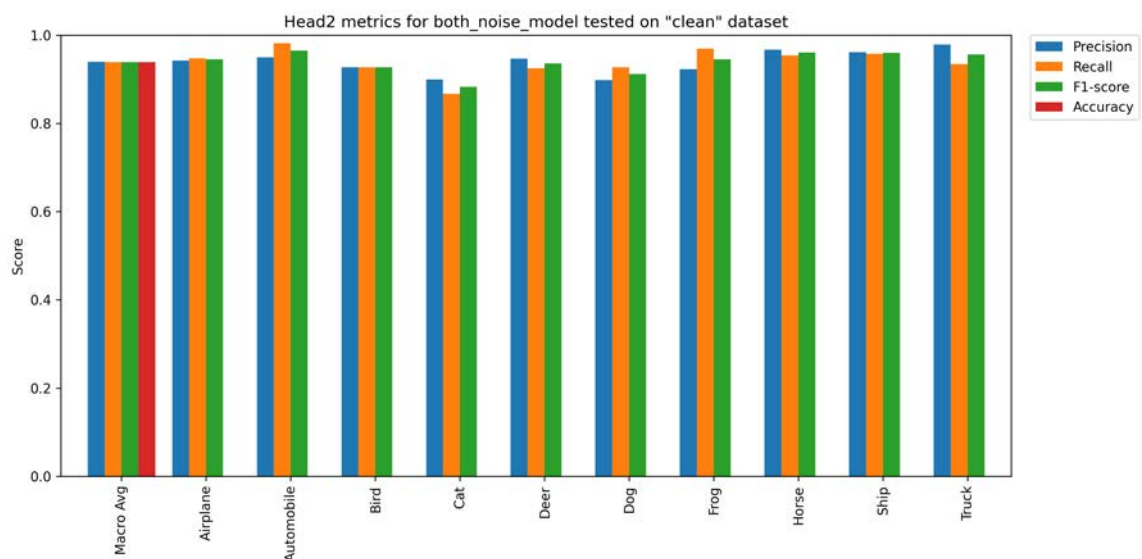


Σχήμα A.18: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε δεδομένα με όλα τα είδη θορύβου και δοκιμάστηκε σε εικόνες με κανονικό θόρυβο.

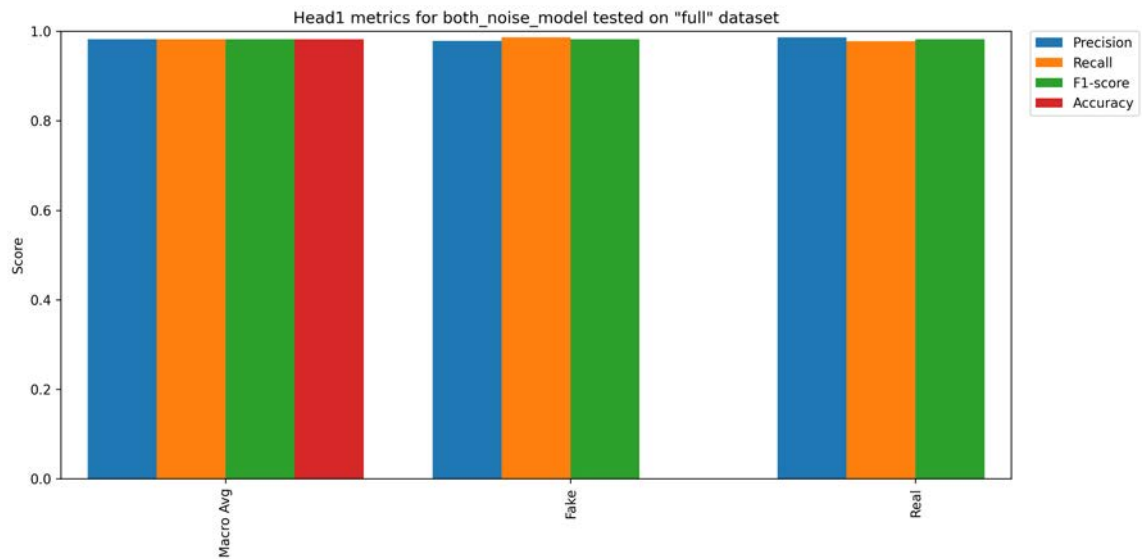
A.2.3 Γραφήματα μοντέλου “both_noise_model”



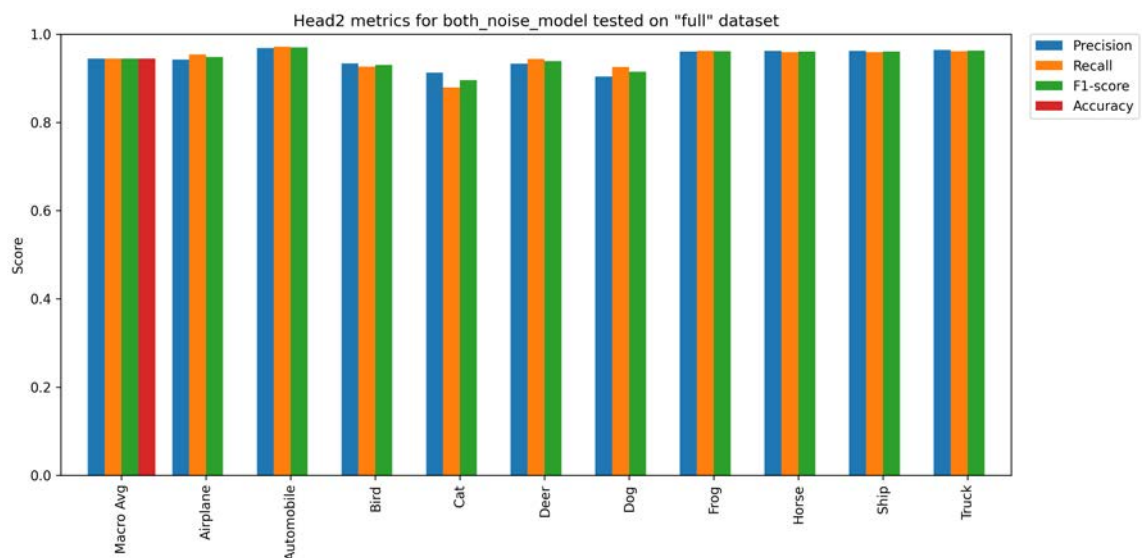
Σχήμα Α.19: Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα χωρίς θόρυβο.



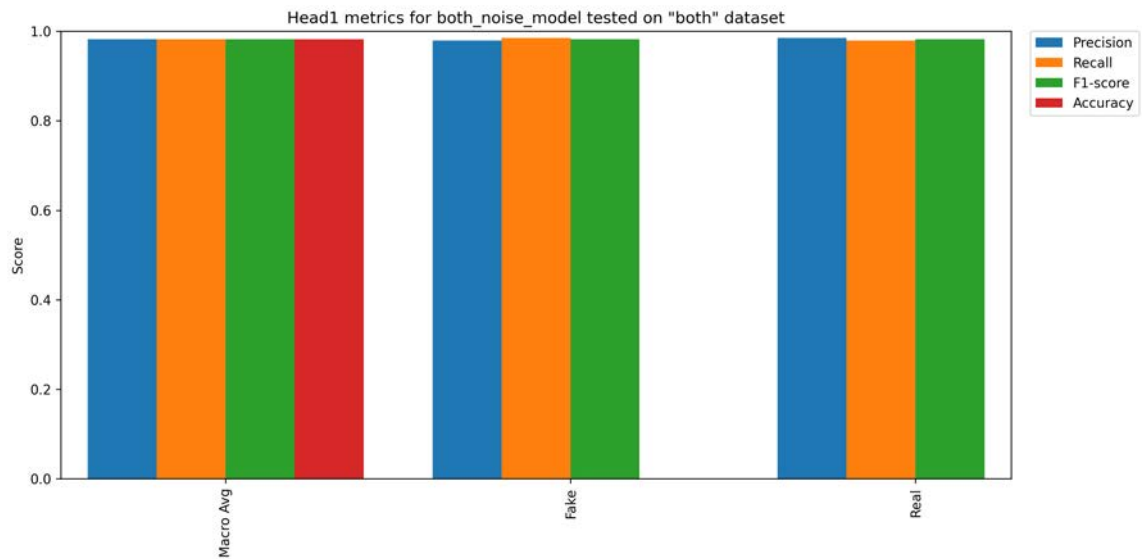
Σχήμα Α.20: Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα χωρίς θόρυβο.



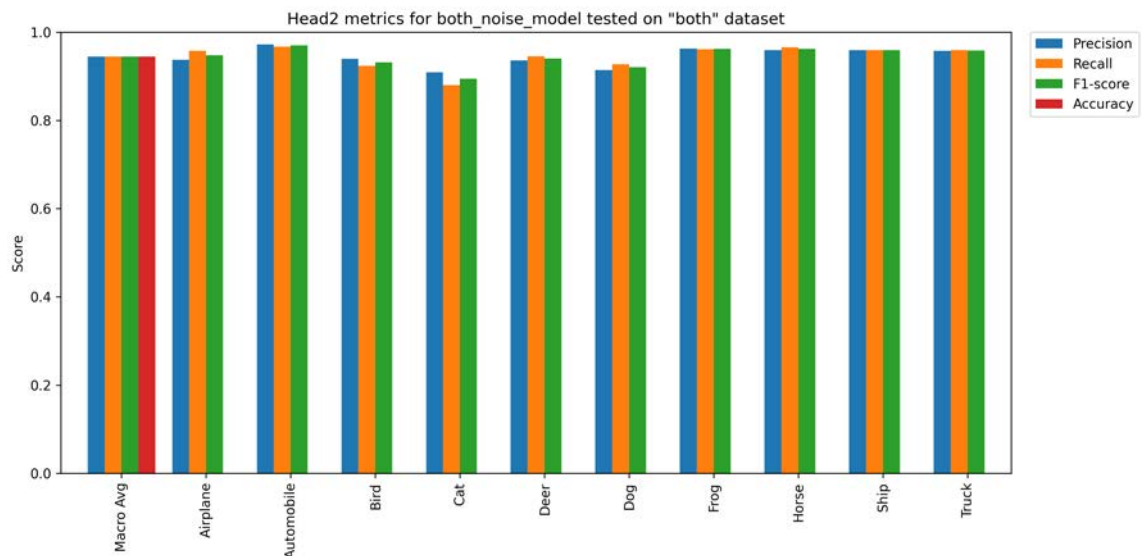
Σχήμα A.21: Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου.



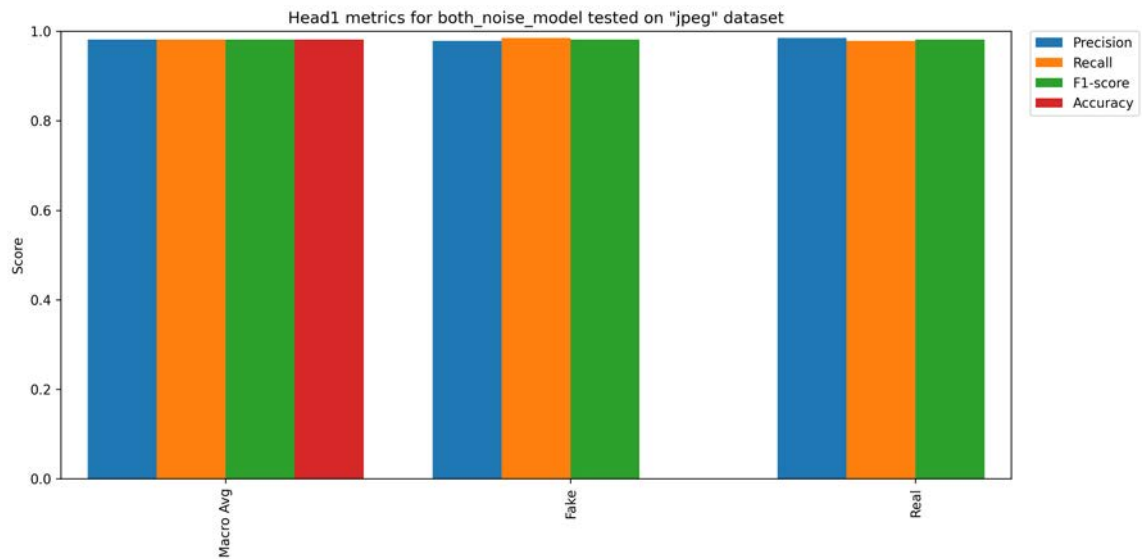
Σχήμα A.22: Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου.



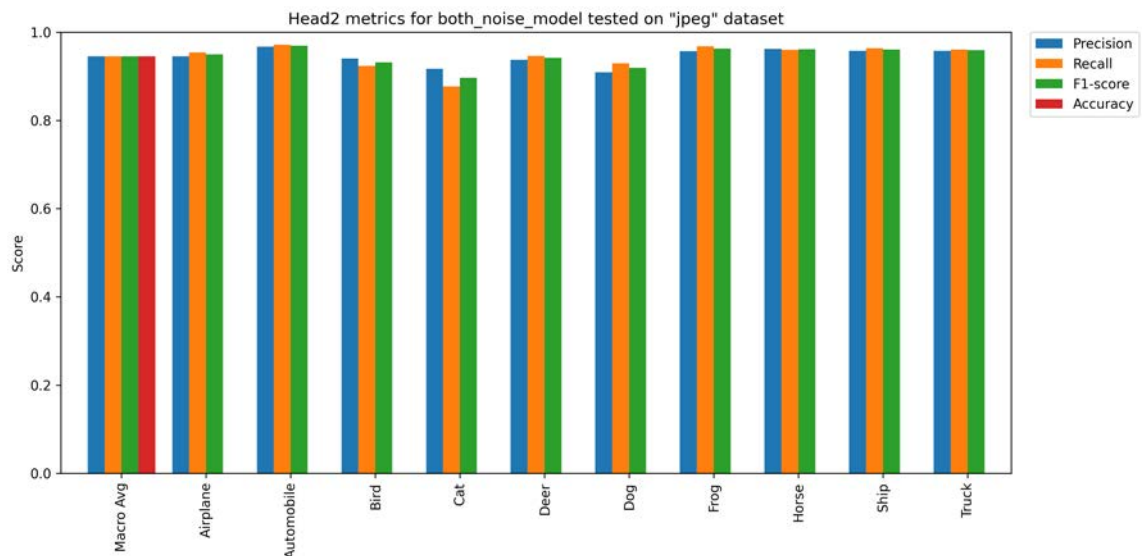
Σχήμα Α.23: Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε στα αντίστοιχα δεδομένα.



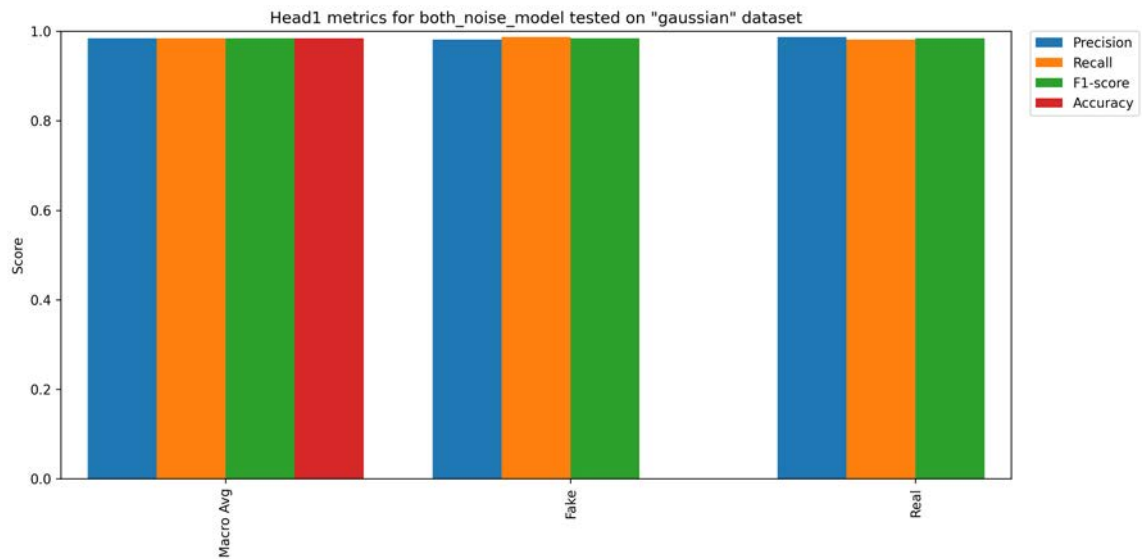
Σχήμα Α.24: Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με όλα τα είδη θορύβου



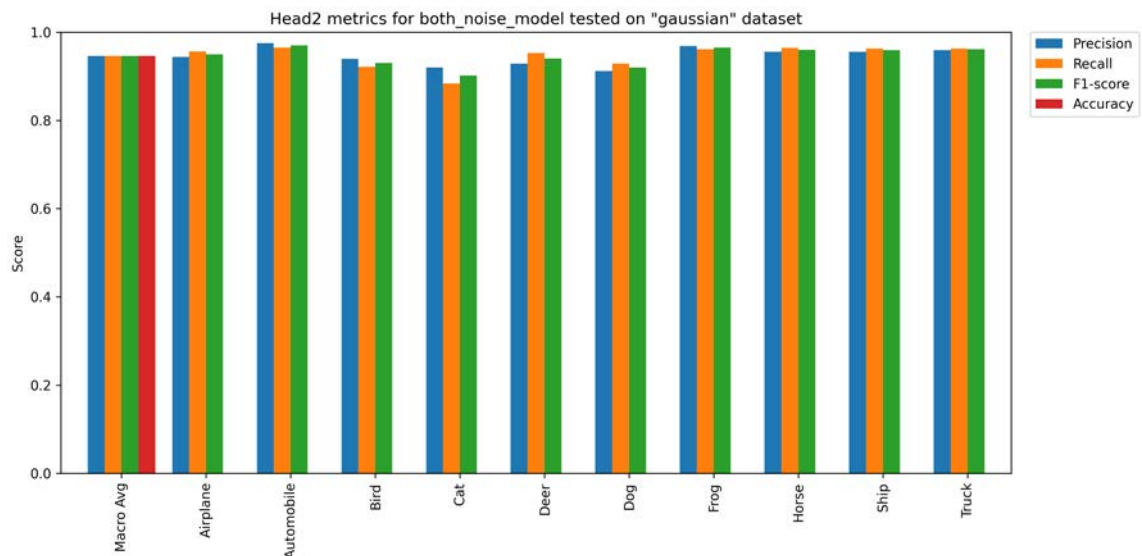
Σχήμα A.25: Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.



Σχήμα A.26: Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με θόρυβο jpeg compression.

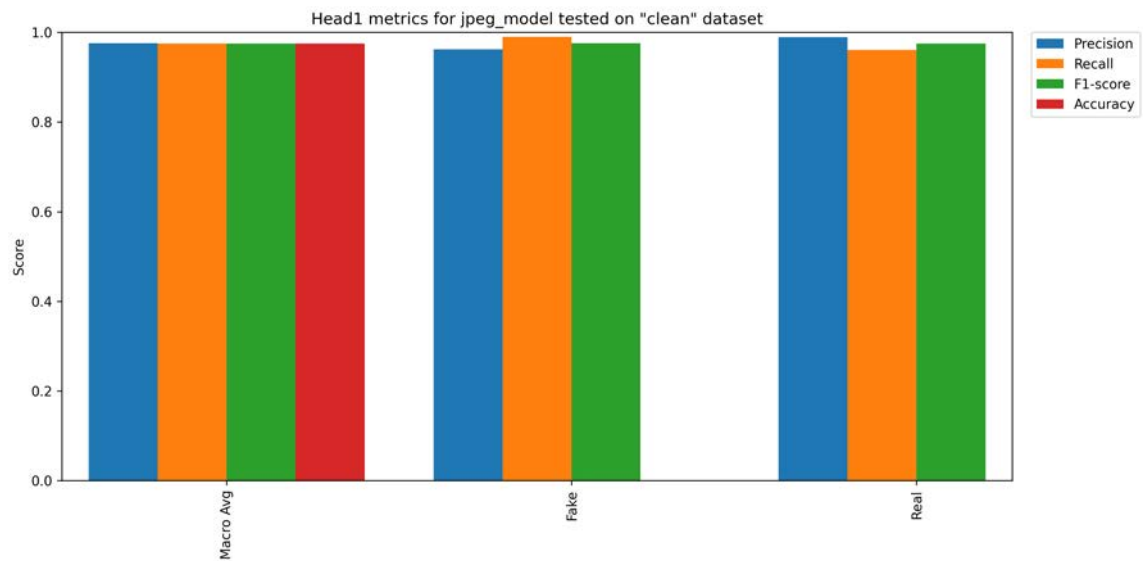


Σχήμα Α.27: Οι μετρικές της πρώτης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

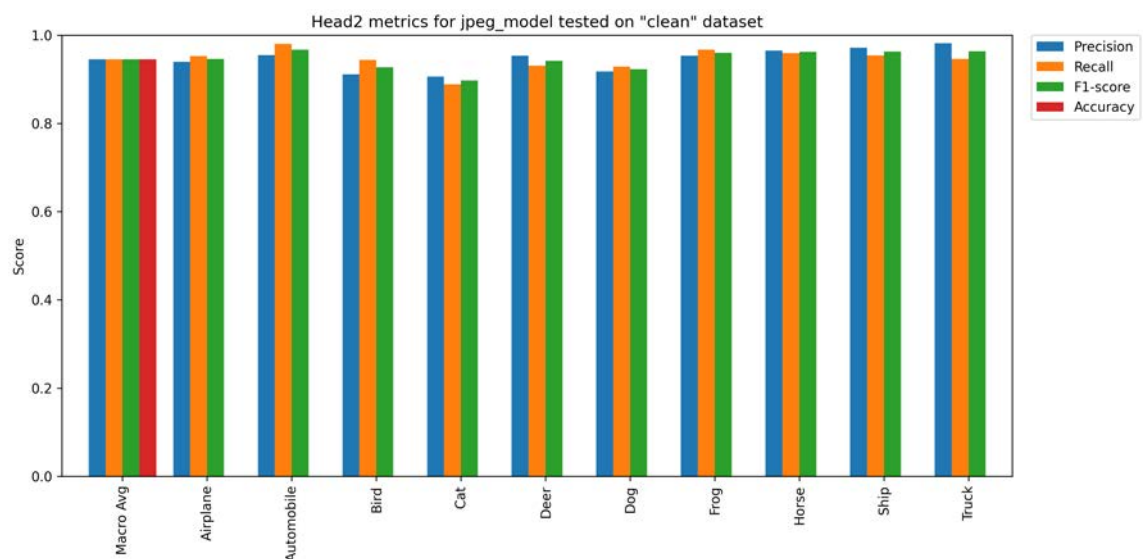


Σχήμα Α.28: Οι μετρικές της δεύτερης κεφαλής για μοντέλο που εκπαιδεύτηκε σε εικόνες με έως ένα είδος θορύβου και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

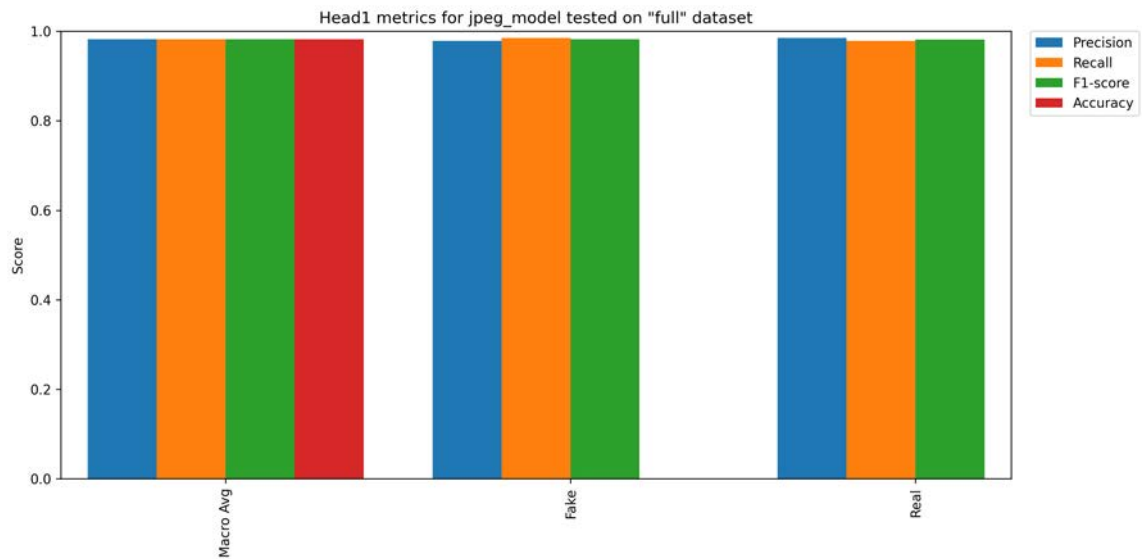
A.2.4 Γραφήματα μοντέλου “jpeg_model”



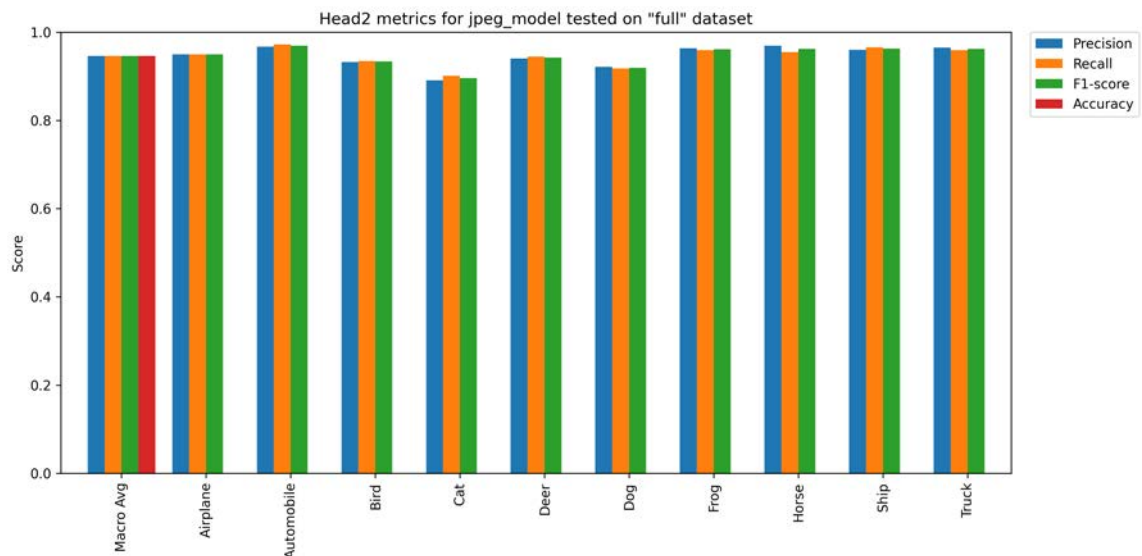
Σχήμα A.29: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.



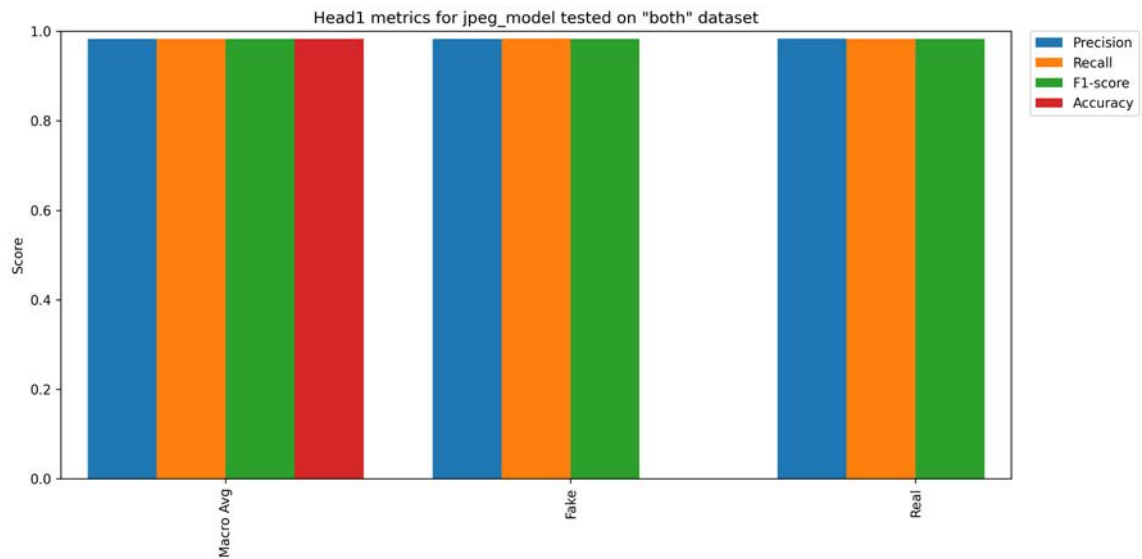
Σχήμα A.30: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.



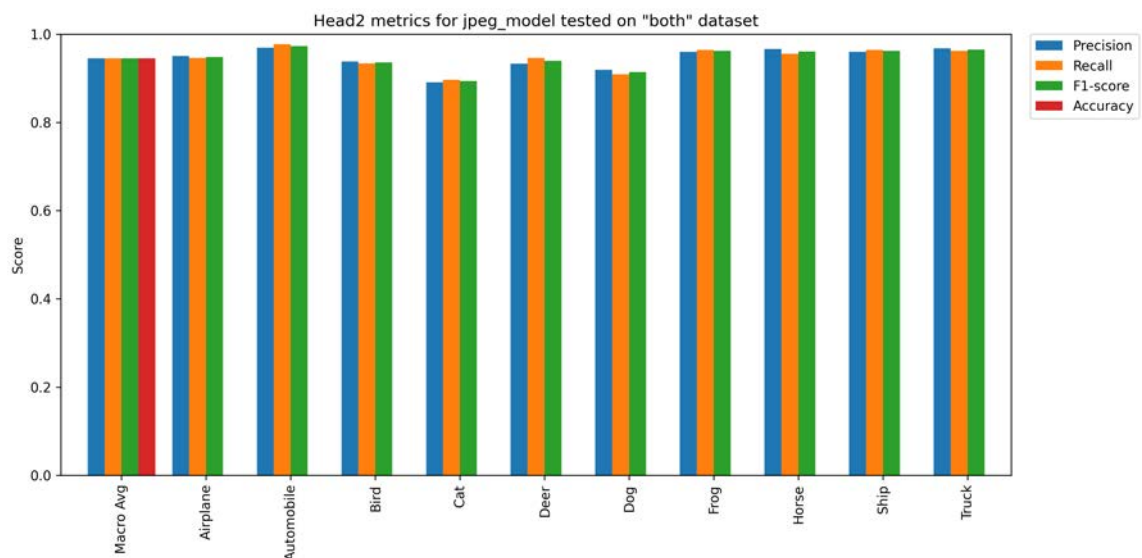
Σχήμα Α.31: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.



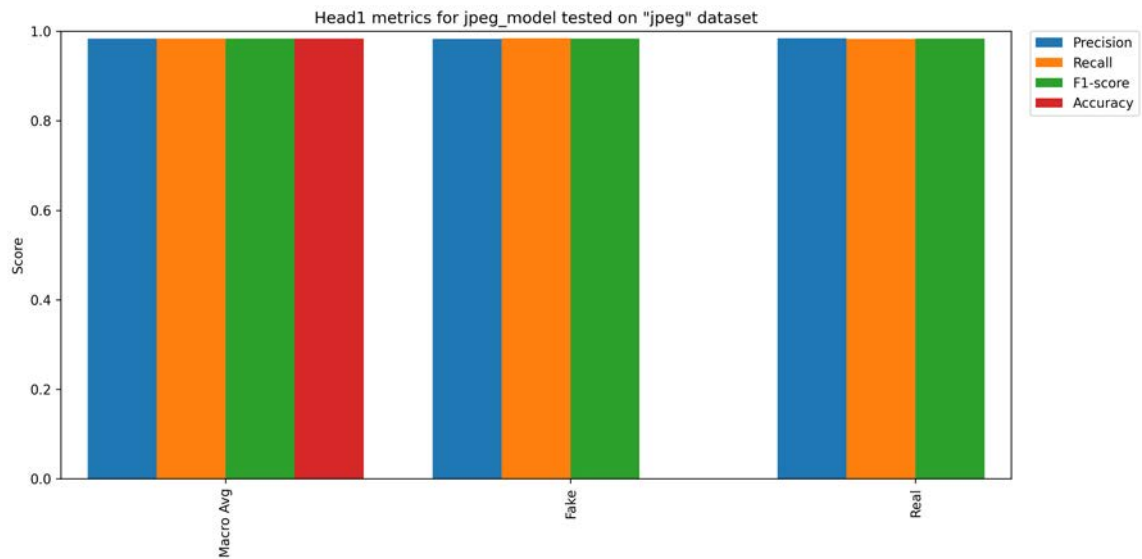
Σχήμα Α.32: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.



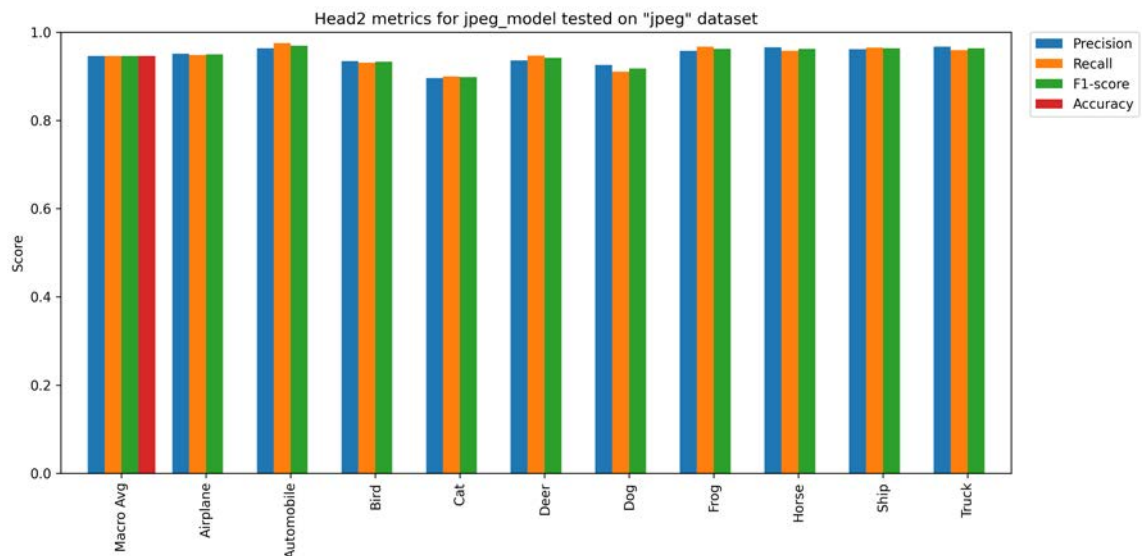
Σχήμα A.33: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.



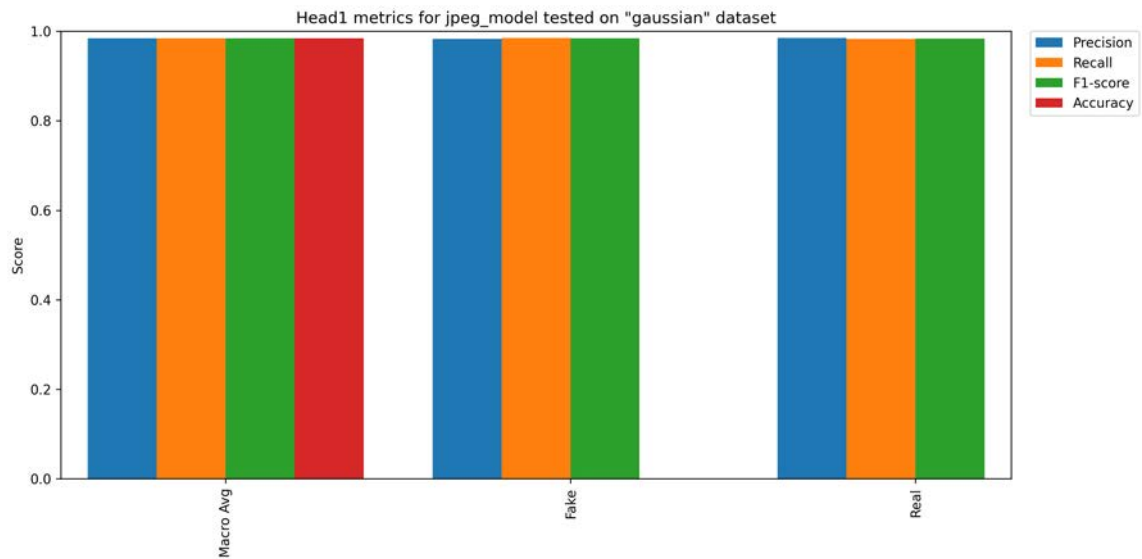
Σχήμα A.34: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.



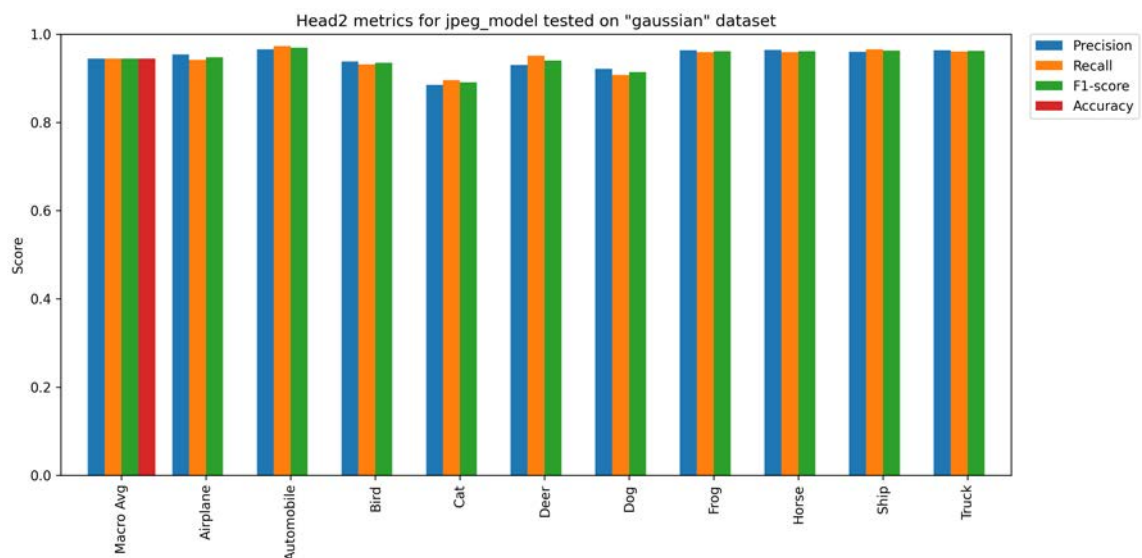
Σχήμα Α.35: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε στα αντίστοιχα δεδομένα.



Σχήμα Α.36: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε στα αντίστοιχα δεδομένα.

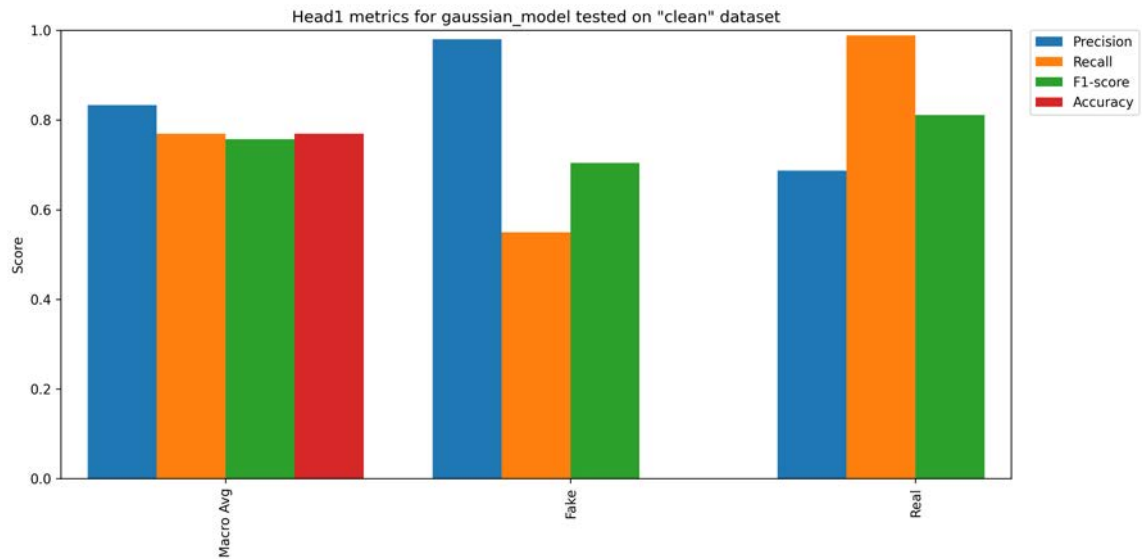


Σχήμα A.37: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

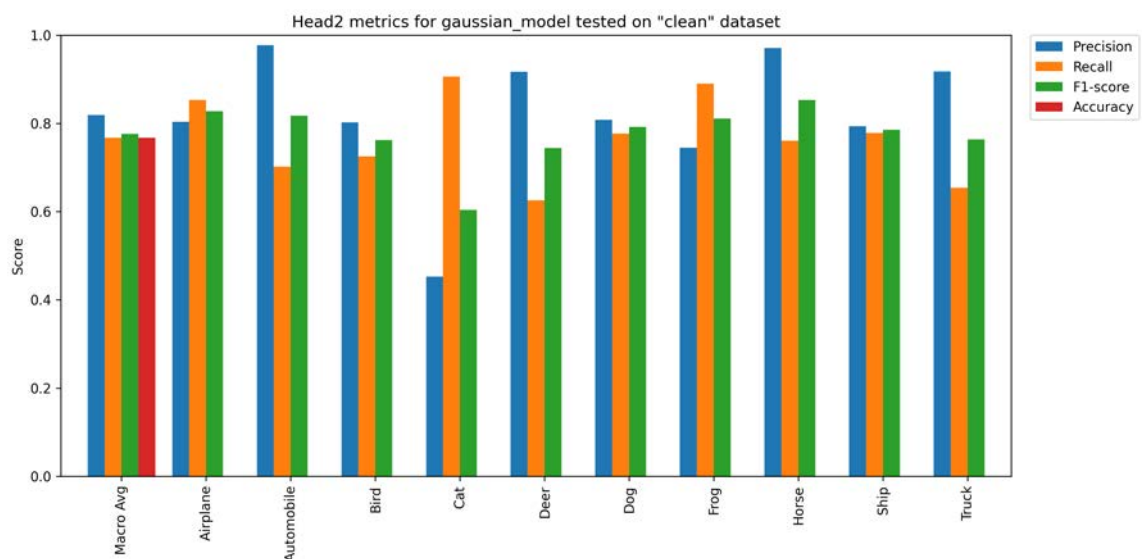


Σχήμα A.38: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε εικόνες με θόρυβο jpeg compression και δοκιμάστηκε σε δεδομένα με κανονικό θόρυβο.

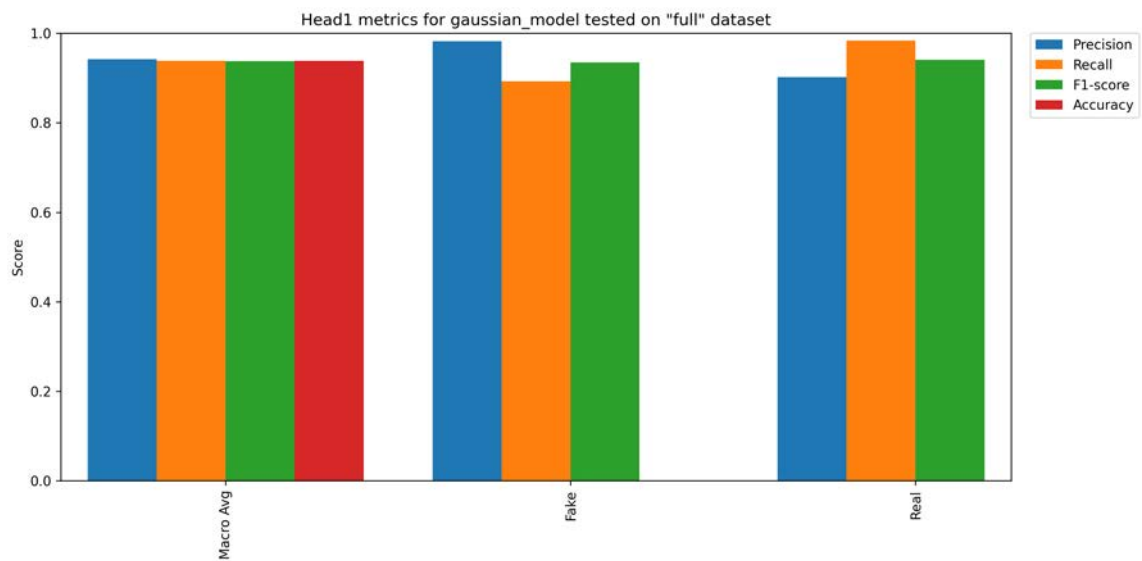
A.2.5 Γραφήματα μοντέλου “gaussian_model”



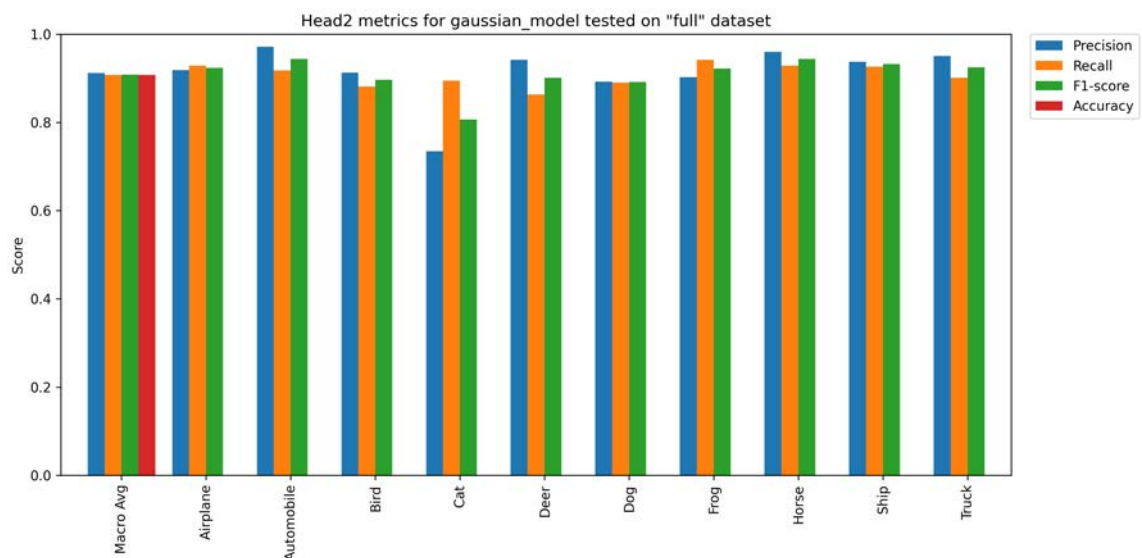
Σχήμα A.39: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.



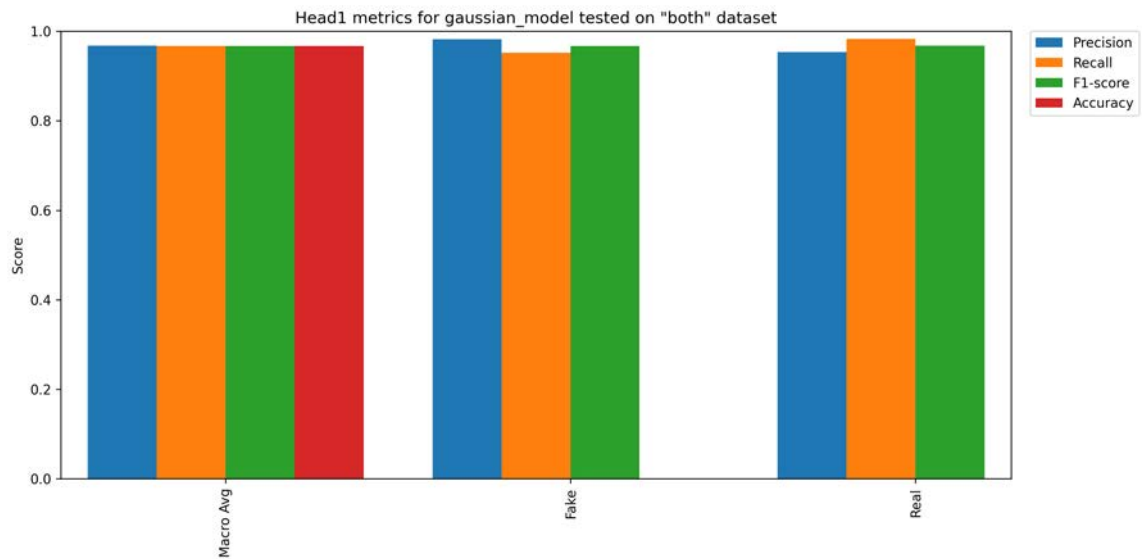
Σχήμα A.40: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες χωρίς θόρυβο.



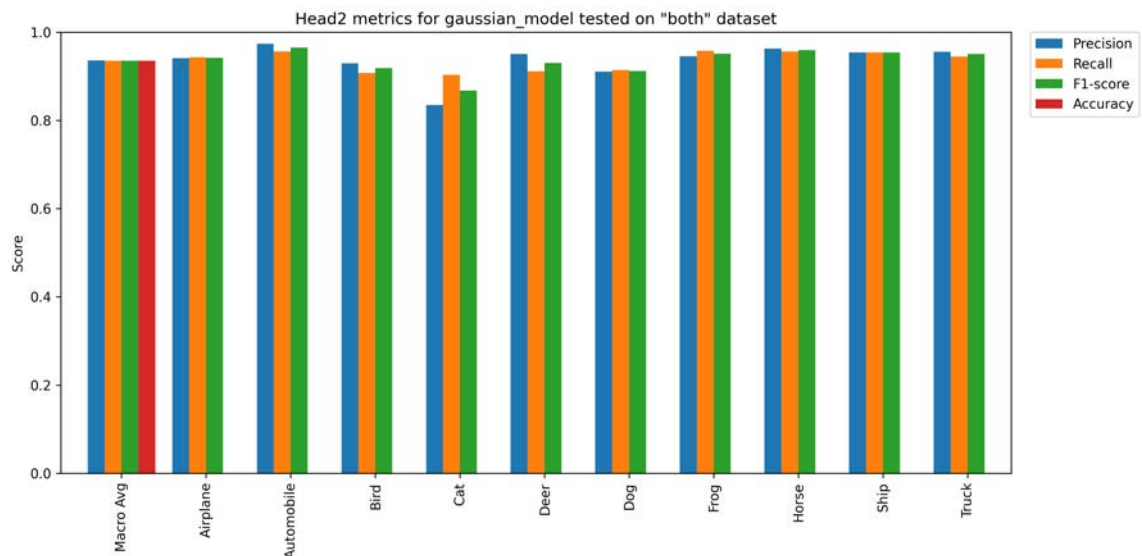
Σχήμα A.41: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.



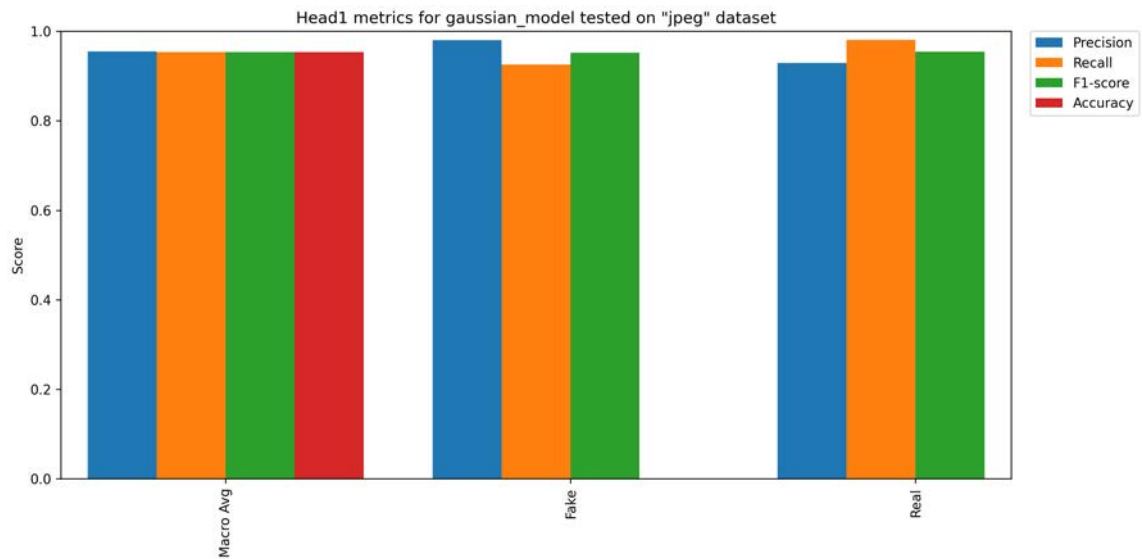
Σχήμα A.42: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με όλα τα είδη θορύβου.



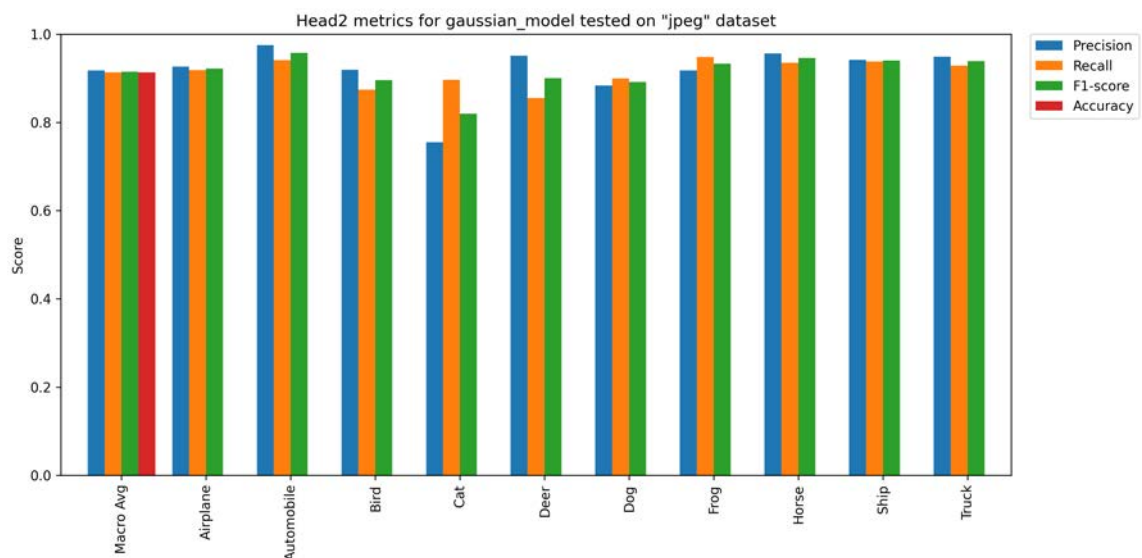
Σχήμα Α.43: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.



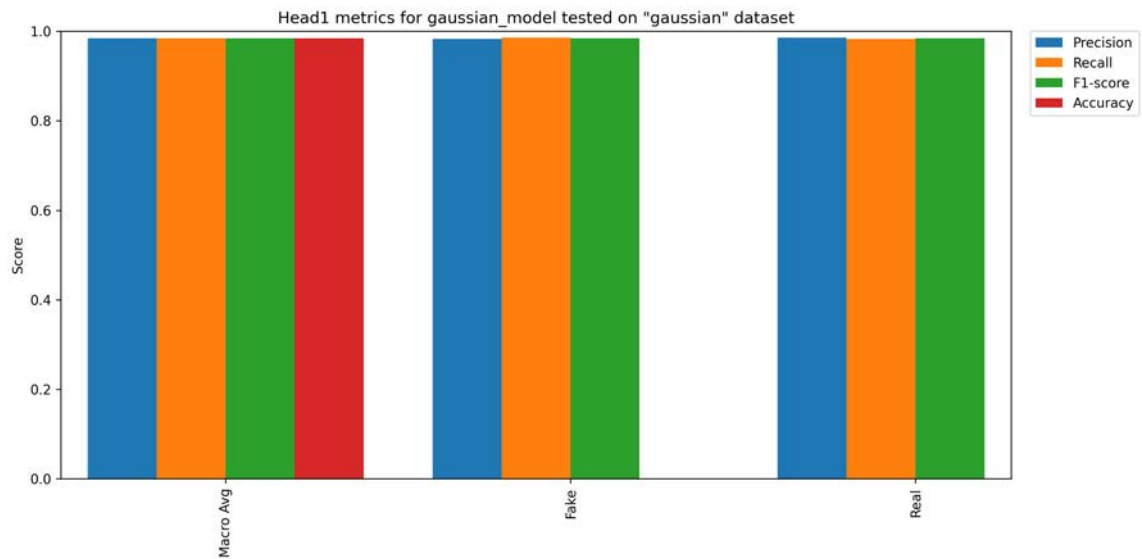
Σχήμα Α.44: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με έως ένα είδος θορύβου.



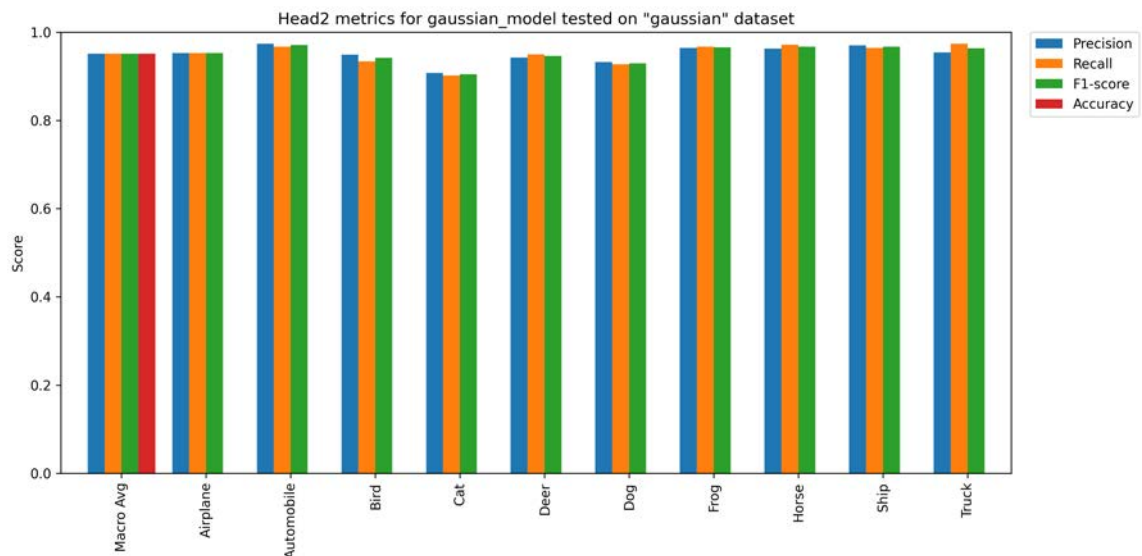
Σχήμα A.45: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.



Σχήμα A.46: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε εικόνες με θόρυβο jpeg compression.



Σχήμα Α.47: Οι μετρικές της πρώτης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε αντίστοιχες εικόνες.

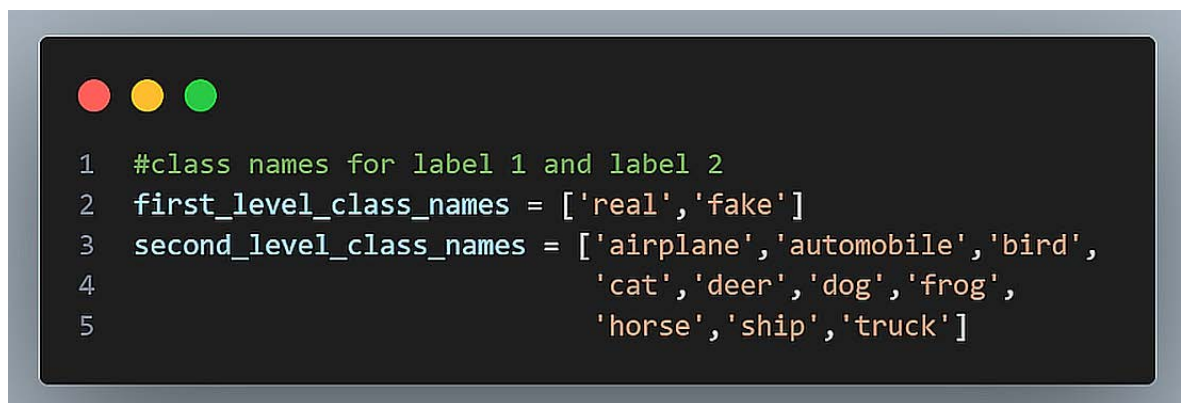


Σχήμα Α.48: Οι μετρικές της δεύτερης κεφαλής για το μοντέλο που εκπαιδεύτηκε σε κανονικό θόρυβο και δοκιμάστηκε σε αντίστοιχες εικόνες.

Παράρτημα Β

Προγραμματιστική υλοποίηση

Σε αυτό το παράρτημα παρουσιάζονται σχήματα με τον κώδικα της υλοποίησης του μοντέλου και της διεξαγωγής των πειραμάτων.

A screenshot of a code editor window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The code is as follows:

```
1 #class names for label 1 and label 2
2 first_level_class_names = ['real', 'fake']
3 second_level_class_names = ['airplane', 'automobile', 'bird',
4                             'cat', 'deer', 'dog', 'frog',
5                             'horse', 'ship', 'truck']
```

Σχήμα Β.1: Οι κλάσεις

```

1  def __init__(self, root_directory, transform):
2      #__init__ initializes important parameters and
3      #navigates root_directory folder structure.
4      self.root_directory = root_directory
5      self.images = []
6      self.transform = transform
7
8      first_level_label_mapping = {'real':0,'fake':1}
9      class_to_idx = {}
10     #creating the dictionaries necessary for both labels
11
12     for first_label in ['real','fake']:
13         complete_first_level_path = os.path.join(root_directory,first_label)
14         for class_name in sorted(os.listdir(complete_first_level_path)):
15             class_dir = os.path.join(complete_first_level_path,
16                                     class_name)
17             if not os.path.isdir(class_dir):
18                 continue
19             if class_name not in class_to_idx:
20                 class_to_idx[class_name] = len(class_to_idx)
21             for fname in os.listdir(class_dir):
22                 if fname.endswith(('.png','.jpeg','.jpg')):
23                     fpath = os.path.join(class_dir,fname)
24                     self.images.append((fpath,
25                                       first_level_label_mapping[first_label],
26                                       class_to_idx[class_name]))
27     self.class_to_idx = class_to_idx
28
29     def __len__(self):
30         #__len__ returns the length of the data samples
31         return len(self.images)
32
33     def __getitem__(self,index):
34         #__getitem__ gets the path and labels of the
35         #image at index position.
36         #Afterwards it opens the image and applies the
37         #transformation on the image.
38         #Returns the image, the labels as a tuple if
39         #successful and prints an error message and proceeds
40         #to next image if unsuccessful.
41         image_path, first_level_label, second_level_label = self.images[index]
42         try:
43             image = Image.open(image_path).convert('RGB')
44             if self.transform:
45                 image = self.transform(image)
46             return image, (first_level_label, second_level_label)
47         except Exception as e:
48             print(f"[ERROR] Couldnt load image {image_path} with error {e}")
49             return self.__getitem__((index+1)%len(self))

```

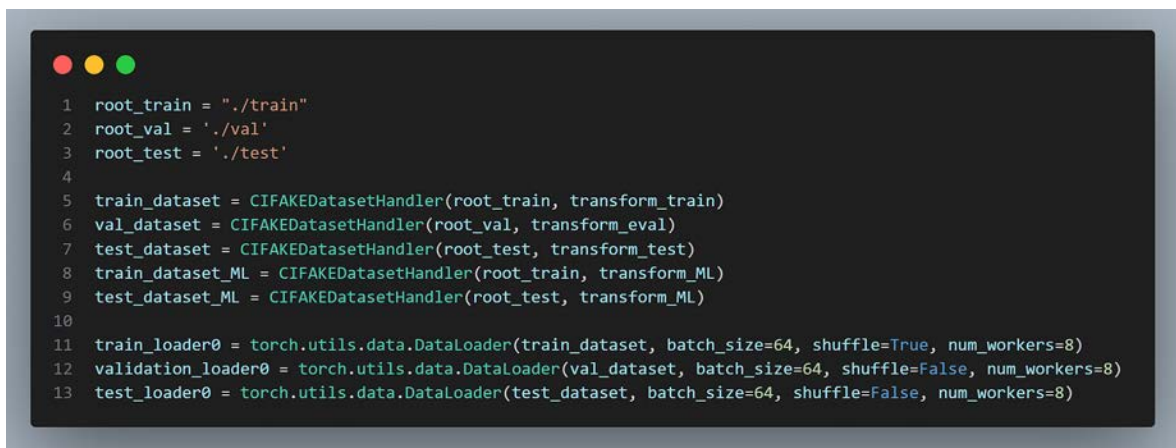
Σχήμα Β.2: Η υλοποίηση των συναρτήσεων `__init__`, `__len__` και `__getitem__` του διαχειριστή δεδομένων `CIFAKEDatasetHandler`

```
1 #ResNet50 is utilized in the experiment.
2 #To use it to its full potential a pretrained on ImageNet version of ResNet50 is used.
3 weights = torchvision.models.ResNet50_Weights.DEFAULT
4
5 prep = weights.transforms()
6 imagenet_mean = (0.485, 0.456, 0.406)
7 imagenet_std = (0.229, 0.224, 0.225)
```

Σχήμα B.3: Ο καθορισμός των βαρών

```
1 #The transformations applied to clean datasets
2 transform_train = transforms.Compose([
3     transforms.Resize(224),
4     transforms.RandomHorizontalFlip(),
5     transforms.RandomRotation(10),
6     transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
7     transforms.ToTensor(),
8     transforms.Normalize(imagenet_mean, imagenet_std)
9 ])
10
11 transform_eval = transforms.Compose([
12     transforms.Resize(224),
13     transforms.ToTensor(),
14     transforms.Normalize(imagenet_mean, imagenet_std)
15 ])
16
17 transform_test = transforms.Compose([
18     transforms.Resize(224),
19     transforms.ToTensor(),
20     transforms.Normalize(imagenet_mean, imagenet_std)
21 ])
```

Σχήμα B.4: Οι λίστες μετασχηματισμών



```
1 root_train = "./train"
2 root_val = './val'
3 root_test = './test'
4
5 train_dataset = CIFAR100DatasetHandler(root_train, transform_train)
6 val_dataset = CIFAR100DatasetHandler(root_val, transform_eval)
7 test_dataset = CIFAR100DatasetHandler(root_test, transform_test)
8 train_dataset_ML = CIFAR100DatasetHandler(root_train, transform_ML)
9 test_dataset_ML = CIFAR100DatasetHandler(root_test, transform_ML)
10
11 train_loader0 = torch.utils.data.DataLoader(train_dataset, batch_size=64, shuffle=True, num_workers=8)
12 validation_loader0 = torch.utils.data.DataLoader(val_dataset, batch_size=64, shuffle=False, num_workers=8)
13 test_loader0 = torch.utils.data.DataLoader(test_dataset, batch_size=64, shuffle=False, num_workers=8)
```

Σχήμα Β.5: Τα σύνολα δεδομένων και οι dataloaders. Τα `train_dataset_ML` και `test_dataset_ML` είναι σύνολα δεδομένων απαραίτητα για το τρίτο πείραμα, αλλά όχι για το συγκεκριμένο.

```

1 class ResNet50Classifier(nn.Module):
2     #The ResNet50Classifier creates the Convolutional Neural Network used for the experiment.
3     #It utilizes a ResNet50 model pretrained on the ImageNet dataset and builds upon it
4     #to create a CNN fit for the experiment.
5     def __init__(self,num_classes1, num_classes2):
6         #The __init__ function takes num_classes1 which represents the amount of
7         #classes for head 1 and num_classes2 which represents the amount of classes
8         #for head 2. It extracts the final fc layer and replaces it with a custom layer.
9         #The final_layer normalizes the data, uses ReLU as activation function and
10        #applies dropout. Then 2 distinct layers connect to this 'final_layer' which are the 2 heads.
11        #The 3 custom layers are not interconnected as head1 and head2 connect only with 'final_layer'
12        #and not with eachother.
13        #Head 1 handles the binary classification between real and ai-generated images.
14        #Head 2 classifies the images into the CIFAR-10 classes.
15        super().__init__()
16        base_model = torchvision.models.resnet50(weights=weights)
17        self.main_arch = nn.Sequential(*list(base_model.children()))[:-1]
18        in_features = base_model.fc.in_features
19        self.final_layer = nn.Sequential(
20            nn.Linear(in_features,512),
21            nn.BatchNorm1d(512),
22            nn.ReLU(),
23            nn.Dropout(0.3)
24        )
25
26        self.head1 = nn.Sequential(
27            nn.Linear(512,256),
28            nn.ReLU(),
29            nn.Dropout(0.2),
30            nn.Linear(256,num_classes1)
31        )
32
33        self.head2 = nn.Sequential(
34            nn.Linear(512,256),
35            nn.ReLU(),
36            nn.Dropout(0.2),
37            nn.Linear(256,num_classes2)
38        )
39
40    def __forward__(self,x):
41        #forward function implements the forwarding logic of the model.
42        x = torch.flatten(self.main_arch(x),1)
43        x = self.final_layer(x)
44        return self.head1(x), self.head2(x)

```

Σχήμα B.6: Ο ορισμός του αντικειμένου ResNet50Classifier και οι συναρτήσεις `__init__` και `__forward__`.

```

1 #initilization of the model. We freeze the resnet50 backbone and only keep the added layers unfrozen
2 #to keep a smoother training phase
3 resnet = ResNet50Classifier(2,10).to(device)
4 for param in resnet.parameters():
5     param.requires_grad = False
6 for param in resnet.head1.parameters():
7     param.requires_grad = True
8 for param in resnet.head2.parameters():
9     param.requires_grad = True
10 for param in resnet.final_layer.parameters():
11     param.requires_grad = True
12 loss_function = nn.CrossEntropyLoss()
13 optimizer = optim.Adam([
14     {"params":resnet.final_layer.parameters(), "lr":3e-4},
15     {"params":resnet.head1.parameters(), "lr":3e-4},
16     {"params":resnet.head2.parameters(), "lr":5e-4}
17 ], weight_decay = 5e-6)
18 scheduler = optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=100)

```

Σχήμα Β.7: Η αρχική προσαρμογή των υπερπαραμέτρων εκπαίδευσης

```

1 def __compute_metrics__(out_true, out_pred, out_prob, head):
2     accuracy = accuracy_score(out_true, out_pred)
3     precision = precision_score(out_true, out_pred, average='macro',zero_division=0)
4     recall=recall_score(out_true,out_pred,average='macro',zero_division=0)
5     f1 = f1_score(out_true,out_pred,average='macro',zero_division=0)
6
7     try:
8         num_classes = out_prob.shape[1]
9
10    except Exception as e:
11        print(f"Error in computing AUC: {e}")
12        auc = float('nan')
13    print(f"{head} Accuracy: {accuracy:.4f}")
14    print(f"{head} Precision: {precision:.4f}")
15    print(f"{head} Recall: {recall:.4f}")
16    print(f"{head} F1 score: {f1:.4f}")
17    return accuracy, precision, recall, f1

```

Σχήμα Β.8: Η συνάρτηση __compute_metrics__

```

1 def __plot_creation__(out_true, out_predicted, out_probabilities, head_name, model_path, class_names, save_directory):
2     #The plot_creation is a helper function which utilizes the results of testing to create and save important plots
3     #and metrics.
4     os.makedirs(save_directory, exist_ok=True)
5     model_name = os.path.splitext(os.path.basename(model_path))[0]
6     df = pd.DataFrame({
7         'ground_truth': out_true,
8         'model_prediction': out_predicted,
9         'probability': np.max(out_probabilities, axis=1)
10    })
11    df[class_names] = out_probabilities
12    df.to_csv(f'{save_directory}/{head_name}_{model_name}_predictions.csv', index = False)
13
14    report = classification_report(out_true, out_predicted, target_names=class_names, output_dict=True, zero_division=0)
15    with open(f'{save_directory}/{head_name}_{model_name}_metrics.json', 'w') as f:
16        json.dump(report, f, indent=2)
17
18    conf_matrix = confusion_matrix(out_true, out_predicted)
19    plt.figure(figsize = (8,6))
20    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels = class_names, yticklabels = class_names)
21    plt.title(f"Confusion Matrix of {head_name}_{model_name}")
22    plt.xlabel("Predicted")
23    plt.ylabel("True")
24    plt.tight_layout()
25    plt.savefig(f'{save_directory}/{head_name}_{model_name}_confusion_matrix.png')
26    plt.close()
27    return report

```

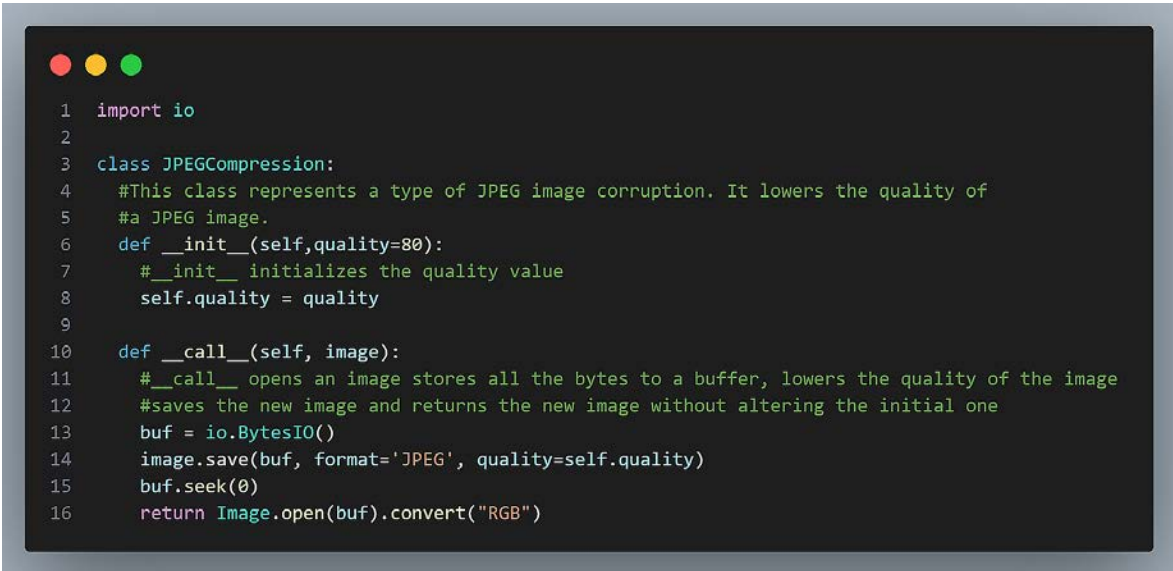
Σχήμα Β.9: Η συνάρτηση __plot_creation__

```

1 class GaussianNoise:
2     #This class represents a type of image corruption. It replaces a set percentage
3     #of pixels' values with a new value obtained from a gaussian distribution
4     def __init__(self, mean=0.0, std=0.05, prob=0.1):
5         # __init__ initializes important parameters. Mean and std determine the new value
6         #prob determines the amount of pixels affected.
7         self.mean = mean
8         self.std = std
9         self.prob = prob
10
11    def __call__(self, tensor):
12        # __call__ utilizes the image's tensor and creates a new tensor where its values are altered
13        #A mask is created so only some of the pixels get affected. The amount of pixels
14        #affected is determined by prob. prob=1 means all pixels are affected, 0 none, 0.1 which is
15        #the standard in this case, means 10% of all pixels are affected.
16        #Lastly all the values of the tensor are normalized to (0,1) range
17        noisy_tensor = torch.randn_like(tensor) * self.std + self.mean
18        mask = torch.rand_like(tensor) < self.prob
19        return torch.clamp(tensor + noisy_tensor * mask.float(), 0.0, 1.0)

```

Σχήμα Β.10: Το αντικείμενο GaussianNoise



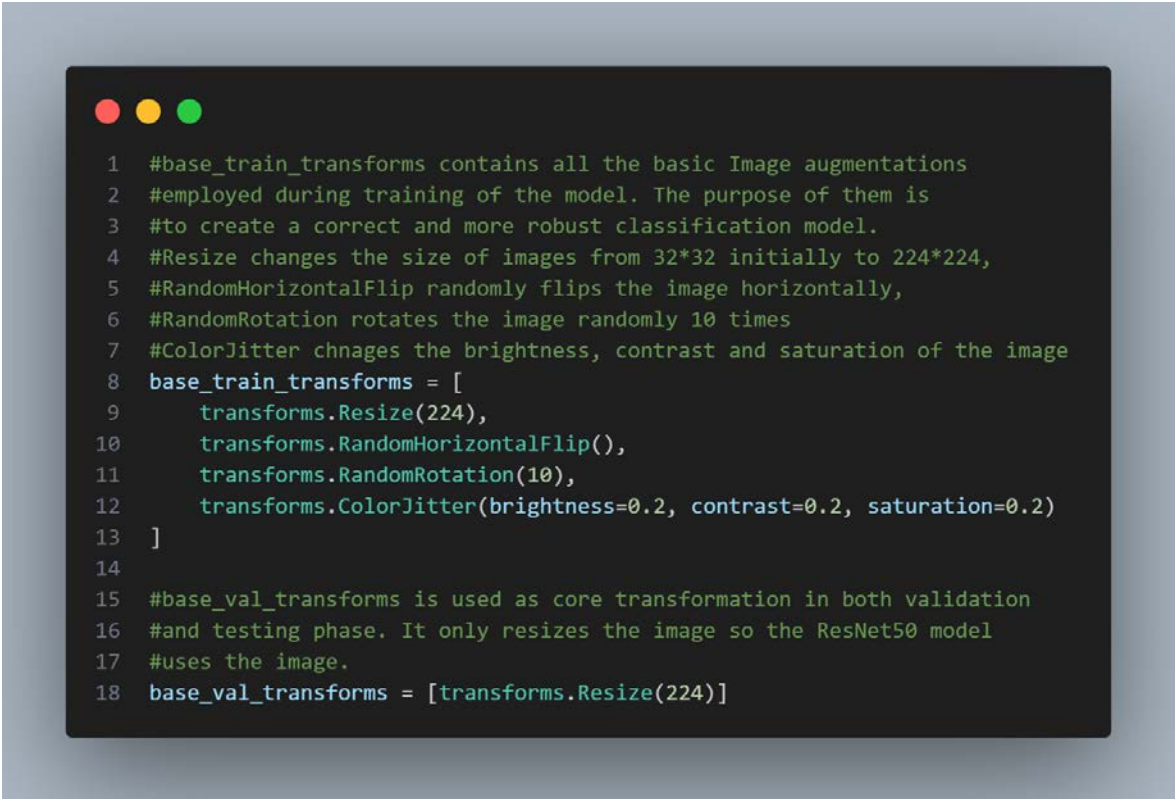
```
1 import io
2
3 class JPEGCompression:
4     #This class represents a type of JPEG image corruption. It lowers the quality of
5     #a JPEG image.
6     def __init__(self,quality=80):
7         #__init__ initializes the quality value
8         self.quality = quality
9
10    def __call__(self, image):
11        #__call__ opens an image stores all the bytes to a buffer, lowers the quality of the image
12        #saves the new image and returns the new image without altering the initial one
13        buf = io.BytesIO()
14        image.save(buf, format='JPEG', quality=self.quality)
15        buf.seek(0)
16        return Image.open(buf).convert("RGB")
```

Σχήμα Β.11: Το αντικείμενο JPEGCompression

```

1 def gen_encoding_map(dataset,noise_level):
2     #The gen_encoding_map creates an encoding map for the whole dataset.
3     #Noise_level determines the type of corruptions to be included in the dataset.
4     #If this function is called then the dataset must contain at least one type of image
5     #corruption. Firstly it makes sure that 50% of the images have no corruption.
6     #The other 50% is guaranteed to have at least one form of corruption.
7     #If the noise_level is set to "all" then that means that approximately 16.67%
8     #of images have gaussian noise, 16.67% have jpeg compression and 16.67% have
9     #both corruption types present. If the noise_level is set to "both" then 25%
10    #of images have gaussian noise and 25% JPEG compression. If the noise_level
11    #is set to "jpeg" all 50% have jpeg compression and finally if noise_level
12    #is set to "gaussian" all 50% have gaussian noise. Also this function guarantees
13    #that all types of corruptions are distributed evenly among images of different labels.
14    #The point of this is to create a fair dataset where no class is influenced more by
15    #image corruptions than others. Also the 50/50 split between clean and modified images
16    #ensures that the model may generalize better without overfitting to specific "noise"
17    #patterns, thus making sure that even a model trained on "noisy" data can still perform
18    #well on a dataset with no disturbances.
19    class_bins = defaultdict(list)
20    for image, label1, label2 in dataset.images:
21        class_bins[(label1,label2)].append(image)
22    encoding_map = {}
23
24    for key,image_paths in class_bins.items():
25        random.shuffle(image_paths)
26        total = len(image_paths)
27        num_clean = total // 2
28        rest = total - num_clean
29
30        if noise_level == "all":
31            num_each_noise = rest // 3
32            for i, image_path in enumerate(image_paths):
33                if i < num_clean:
34                    encoding = 0
35                elif i < num_clean + num_each_noise:
36                    encoding = 10
37                elif i < num_clean + 2*num_each_noise:
38                    encoding = 1
39                else:
40                    encoding = 11
41                encoding_map[image_path] = encoding
42        elif noise_level == "both":
43            num_each_noise = rest // 2
44            for i, image_path in enumerate(image_paths):
45                if i < num_clean:
46                    encoding = 0
47                elif i < num_clean + num_each_noise:
48                    encoding = 10
49                else:
50                    encoding = 1
51                encoding_map[image_path] = encoding
52        elif noise_level == "jpeg":
53            for i, image_path in enumerate(image_paths):
54                if i < num_clean:
55                    encoding = 0
56                else:
57                    encoding = 10
58                encoding_map[image_path] = encoding
59        elif noise_level == "gaussian":
60            for i, image_path in enumerate(image_paths):
61                if i < num_clean:
62                    encoding = 0
63                else:
64                    encoding = 1
65                encoding_map[image_path] = encoding
66    return encoding_map

```

A screenshot of a code editor with a dark background and light green text. The code defines two lists of image transformations: `base_train_transforms` and `base_val_transforms`. The training transforms include `Resize(224)`, `RandomHorizontalFlip()`, `RandomRotation(10)`, and `ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2)`. The validation transforms only include `Resize(224)`. Comments explain the purpose of these transforms for training and validation.

```
1  #base_train_transforms contains all the basic Image augmentations
2  #employed during training of the model. The purpose of them is
3  #to create a correct and more robust classification model.
4  #Resize changes the size of images from 32*32 initially to 224*224,
5  #RandomHorizontalFlip randomly flips the image horizontally,
6  #RandomRotation rotates the image randomly 10 times
7  #ColorJitter chnages the brightness, contrast and saturation of the image
8  base_train_transforms = [
9      transforms.Resize(224),
10     transforms.RandomHorizontalFlip(),
11     transforms.RandomRotation(10),
12     transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2)
13 ]
14
15 #base_val_transforms is used as core transformation in both validation
16 #and testing phase. It only resizes the image so the ResNet50 model
17 #uses the image.
18 base_val_transforms = [transforms.Resize(224)]
```

Σχήμα Β.13: Οι βασικοί μετασχηματισμοί ανεξαρτήτως encoding

```

1 def decode_transform(encoding,mode='train'):
2     #The purpose of decode_transform is to determine which
3     #type of corruption is used in the image based on the encoding given
4     #and to compile the correct list of image transformations for given image.
5     #Encoding is a 2 digit number and determines if the image has no corruption,
6     #only gaussian noise, only jpeg compression or both.
7     #Also checks if the image is used during training, If used in training it
8     #incorporates all the extra training transformations.
9     #This function returns the compiled list of all transformations for the image.
10    group1 = encoding // 10
11    group2 = encoding % 10
12    if mode == 'train':
13        transform_list = base_train_transforms.copy()
14    else:
15        transform_list = base_val_transforms.copy()
16    if group2 == 0:
17        transform_list.append(JPEGCompression(quality=80))
18    transform_list.append(transforms.ToTensor())
19    if group1 == 0:
20        transform_list.append(GaussianNoise(std=0.05))
21    transform_list.append(transforms.Normalize(imagenet_mean,imagenet_std))
22
23    return transforms.Compose(transform_list)

```

Σχήμα B.14: Η συνάρτηση decode_transform

```

1 def create_transforms_cache(mode):
2     #create_transforms_cache is a helper function. It just creates a
3     #dictionary for all encodings and their use. It acts as a cache
4     #since the encodings are stored as keys in a dictionary (hash table) providing
5     #O(1) access time on average. Therefore this function speeds up
6     #significantly the transformation of images.
7     encodings = [0,10,1,11]
8     return {enc: decode_transform(enc,mode) for enc in encodings}

```

Σχήμα B.15: Η συνάρτηση create_transforms_cache

```

1 class CIFAKENoisyDatasetHandler(torch.utils.data.Dataset):
2     #This is a wrapper dataset class for the class CIFAKEDatasetHandler. It's function is to
3     #handle datasets where image corruptions are present.
4     def __init__(self, dataset, encoding_map, trans_cache):
5         #The __init__ function initializes the important parameters for the dataset.
6         #The parameter dataset refers to the CIFAKEDatasetHandler dataset that is used as base,
7         #the parameter encoding_map refers to the encoding map for all image corruptions
8         #and trans_cache is the cache for all transformations.
9         self.dataset = dataset
10        self.encoding_map = encoding_map
11        self.trans_cache = trans_cache
12
13    def __len__(self):
14        #The __len__ function returns the length of the base dataset samples which
15        #is the same as the length of this dataset.
16        return len(self.dataset)
17
18    def __getitem__(self, index):
19        #The __getitem__ gets the image path and the labels of the image with the given index
20        #in the base dataset. It's the same image for both datasets.
21        #Afterwards, the encoding of the encoding map for the given image is extracted,
22        #the transformations are created and then applied to the image.
23        #The function returns the image, both of the labels if successful else,
24        #returns an error message and proceeds to process the next image in the loader.
25        image_path, first_level_label, second_level_label = self.dataset.images[index]
26        try:
27            image = Image.open(image_path).convert('RGB')
28            encoding = self.encoding_map[image_path]
29            transform = self.trans_cache[encoding]
30            image = transform(image)
31            return image, (first_level_label, second_level_label)
32        except Exception as e:
33            print(f"[ERROR] Couldnt load image {image_path} with error {e}")
34            return self.__getitem__((index+1)%len(self.dataset))
35

```

Σχήμα Β.16: Ο διαχειριστής δεδομένων CIFAKENoisyDatasetHandler

```

1 root_train = "./train"
2 root_val = './val'
3 root_test = './test'
4 #The creation of the datasets and dataloaders with all types of corruptions
5 train_dataset_mix_1 = CIFAKEDatasetHandler(root_train, None)
6 val_dataset_mix_1 = CIFAKEDatasetHandler(root_val, None)
7 test_dataset_mix_1 = CIFAKEDatasetHandler(root_test, None)
8 encoding_map_train_mix_1 = gen_encoding_map(train_dataset_mix_1,"all")
9 encoding_map_val_mix_1 = gen_encoding_map(val_dataset_mix_1,"all")
10 encoding_map_test_mix_1 = gen_encoding_map(test_dataset_mix_1,"all")
11 train_trans_cache = create_transforms_cache(mode='train')
12 val_trans_cache = create_transforms_cache(mode='val')
13 test_trans_cache = create_transforms_cache(mode='test')
14 wrapped_train_dataset_mix_1 = CIFAKENoisyDatasetHandler(train_dataset_mix_1, encoding_map_train_mix_1, train_trans_cache)
15 wrapped_val_dataset_mix_1 = CIFAKENoisyDatasetHandler(val_dataset_mix_1, encoding_map_val_mix_1, val_trans_cache)
16 wrapped_test_dataset_mix_1 = CIFAKENoisyDatasetHandler(test_dataset_mix_1, encoding_map_test_mix_1, test_trans_cache)
17 train_loader1 = torch.utils.data.DataLoader(wrapped_train_dataset_mix_1, batch_size=64, shuffle=True, num_workers=8)
18 validation_loader1 = torch.utils.data.DataLoader(wrapped_val_dataset_mix_1, batch_size=64, shuffle=False, num_workers=8)
19 test_loader1 = torch.utils.data.DataLoader(wrapped_test_dataset_mix_1, batch_size=64, shuffle=False, num_workers=8)

```

Σχήμα B.17: Δημιουργία συνόλων με όλα τα είδη θορύβου και συνύπαρξη των δύο ειδών θορύβου ταυτόχρονα σε φωτογραφίες.

```

1 #The creation of the datasets and dataloaders with 'both' type of corruption
2 train_dataset_mix_2 = CIFAKEDatasetHandler(root_train, None)
3 val_dataset_mix_2 = CIFAKEDatasetHandler(root_val, None)
4 test_dataset_mix_2 = CIFAKEDatasetHandler(root_test, None)
5 encoding_map_train_mix_2 = gen_encoding_map(train_dataset_mix_2,"both")
6 encoding_map_val_mix_2 = gen_encoding_map(val_dataset_mix_2,"both")
7 encoding_map_test_mix_2 = gen_encoding_map(test_dataset_mix_2,"both")
8 train_trans_cache = create_transforms_cache(mode='train')
9 val_trans_cache = create_transforms_cache(mode='val')
10 test_trans_cache = create_transforms_cache(mode='test')
11 wrapped_train_dataset_mix_2 = CIFAKENoisyDatasetHandler(train_dataset_mix_2, encoding_map_train_mix_2, train_trans_cache)
12 wrapped_val_dataset_mix_2 = CIFAKENoisyDatasetHandler(val_dataset_mix_2, encoding_map_val_mix_2, val_trans_cache)
13 wrapped_test_dataset_mix_2 = CIFAKENoisyDatasetHandler(test_dataset_mix_2, encoding_map_test_mix_2, test_trans_cache)
14 train_loader2 = torch.utils.data.DataLoader(wrapped_train_dataset_mix_2, batch_size=64, shuffle=True, num_workers=8)
15 validation_loader2 = torch.utils.data.DataLoader(wrapped_val_dataset_mix_2, batch_size=64, shuffle=False, num_workers=8)
16 test_loader2 = torch.utils.data.DataLoader(wrapped_test_dataset_mix_2, batch_size=64, shuffle=False, num_workers=8)

```

Σχήμα B.18: Δημιουργία συνόλων με όλα τα είδη θορύβου χωρίς συνύπαρξη των δύο ειδών θορύβου ταυτόχρονα σε φωτογραφίες.

```

1 #The creation of the datasets and dataloaders with gaussian noise corruption
2 train_dataset_mix_3 = CIFAKEDatasetHandler(root_train, None)
3 val_dataset_mix_3 = CIFAKEDatasetHandler(root_val, None)
4 test_dataset_mix_3 = CIFAKEDatasetHandler(root_test, None)
5 encoding_map_train_mix_3 = gen_encoding_map(train_dataset_mix_3,"jpeg")
6 encoding_map_val_mix_3 = gen_encoding_map(val_dataset_mix_3,"jpeg")
7 encoding_map_test_mix_3 = gen_encoding_map(test_dataset_mix_3,"jpeg")
8 train_trans_cache = create_transforms_cache(mode='train')
9 val_trans_cache = create_transforms_cache(mode='val')
10 test_trans_cache = create_transforms_cache(mode='test')
11 wrapped_train_dataset_mix_3 = CIFAKENoisyDatasetHandler(train_dataset_mix_3, encoding_map_train_mix_3, train_trans_cache)
12 wrapped_val_dataset_mix_3 = CIFAKENoisyDatasetHandler(val_dataset_mix_3, encoding_map_val_mix_3, val_trans_cache)
13 wrapped_test_dataset_mix_3 = CIFAKENoisyDatasetHandler(test_dataset_mix_3, encoding_map_test_mix_3, test_trans_cache)
14 train_loader3 = torch.utils.data.DataLoader(wrapped_train_dataset_mix_3, batch_size=64, shuffle=True, num_workers=8)
15 validation_loader3 = torch.utils.data.DataLoader(wrapped_val_dataset_mix_3, batch_size=64, shuffle=False, num_workers=8)
16 test_loader3 = torch.utils.data.DataLoader(wrapped_test_dataset_mix_3, batch_size=64, shuffle=False, num_workers=8)

```

Σχήμα B.19: Δημιουργία συνόλων με JPEG Compression θόρυβο.

```

1 #The creation of datasets and dataloader jpeg compression
2 train_dataset_mix_4 = CIFAKEDatasetHandler(root_train, None)
3 val_dataset_mix_4 = CIFAKEDatasetHandler(root_val, None)
4 test_dataset_mix_4 = CIFAKEDatasetHandler(root_test, None)
5 encoding_map_train_mix_4 = gen_encoding_map(train_dataset_mix_4,"gaussian")
6 encoding_map_val_mix_4 = gen_encoding_map(val_dataset_mix_4,"gaussian")
7 encoding_map_test_mix_4 = gen_encoding_map(test_dataset_mix_4,"gaussian")
8 train_trans_cache = create_transforms_cache(mode='train')
9 val_trans_cache = create_transforms_cache(mode='val')
10 test_trans_cache = create_transforms_cache(mode='test')
11 wrapped_train_dataset_mix_4 = CIFAKENoisyDatasetHandler(train_dataset_mix_4, encoding_map_train_mix_4, train_trans_cache)
12 wrapped_val_dataset_mix_4 = CIFAKENoisyDatasetHandler(val_dataset_mix_4, encoding_map_val_mix_4, val_trans_cache)
13 wrapped_test_dataset_mix_4 = CIFAKENoisyDatasetHandler(test_dataset_mix_4, encoding_map_test_mix_4, test_trans_cache)
14 train_loader4 = torch.utils.data.DataLoader(wrapped_train_dataset_mix_4, batch_size=64, shuffle=True, num_workers=8)
15 validation_loader4 = torch.utils.data.DataLoader(wrapped_val_dataset_mix_4, batch_size=64, shuffle=False, num_workers=8)
16 test_loader4 = torch.utils.data.DataLoader(wrapped_test_dataset_mix_4, batch_size=64, shuffle=False, num_workers=8)
17
18 #In each of these cases its important that while training the validation's set
19 #distribution matches the training's set. Also for cross-testing its important
20 #to create test datasets with a valid distribution matching the equivalent training
21 #distribution.

```

Σχήμα Β.20: Δημιουργία συνόλων με Gaussian noise.

```

1 train_ml_loader = torch.utils.data.DataLoader(train_dataset_ML, batch_size=64,shuffle=True,num_workers=0)
2 test_ml_loader = torch.utils.data.DataLoader(test_dataset_ML, batch_size=64,num_workers=0)

```

Σχήμα Β.21: Οι dataloaders

```

1 transform_ML = transforms.Compose([
2     transforms.ToTensor()
3 ])

```

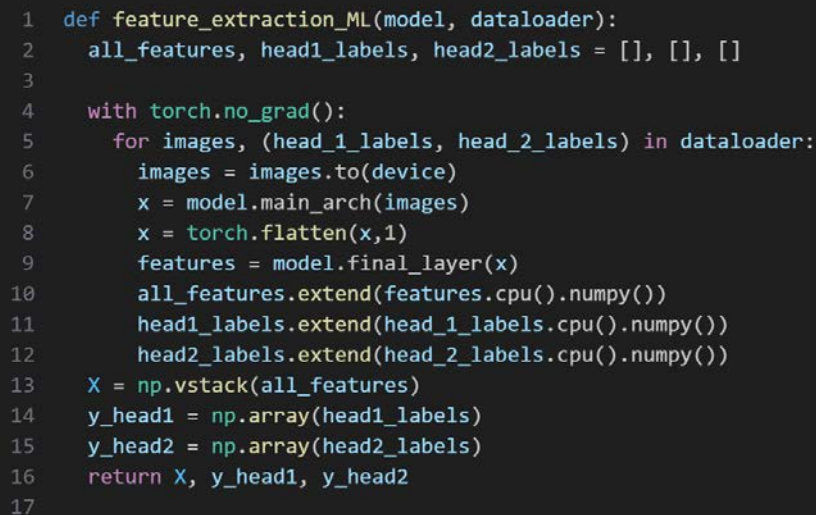
Σχήμα Β.22: Ο μετασχηματισμός για ML δεδομένα

```
1 from tqdm import tqdm
2 def flatten_loader(loader):
3     X_list, y_head1_list, y_head2_list = [], [], []
4
5     for images, (head_1_labels, head_2_labels) in tqdm(loader, desc='Flattening'):
6         #images = images.to(device)
7         #head_1_labels = head_1_labels.to(device)
8         #head_2_labels = head_2_labels.to(device)
9         images_flat = images.view(images.size(0), -1).cpu().numpy()
10        X_list.append(images_flat)
11        y_head1_list.append(head_1_labels.cpu().numpy())
12        y_head2_list.append(head_2_labels.cpu().numpy())
13
14    X = np.concatenate(X_list, axis=0)
15    y_head1 = np.concatenate(y_head1_list, axis=0)
16    y_head2 = np.concatenate(y_head2_list, axis=0)
17    return X, y_head1, y_head2
```

Σχήμα B.23: Η συνάρτηση flatten_loader

```
1 X_raw_train, y_head1_train, y_head2_train = flatten_loader(train_ml_loader)
2 X_raw_test, y_head1_test, y_head2_test = flatten_loader(test_ml_loader)
```

Σχήμα B.24: Η απόκτηση των διανυσμάτων χαρακτηριστικών



```
1 def feature_extraction_ML(model, dataloader):
2     all_features, head1_labels, head2_labels = [], [], []
3
4     with torch.no_grad():
5         for images, (head_1_labels, head_2_labels) in dataloader:
6             images = images.to(device)
7             x = model.main_arch(images)
8             x = torch.flatten(x,1)
9             features = model.final_layer(x)
10            all_features.extend(features.cpu().numpy())
11            head1_labels.extend(head_1_labels.cpu().numpy())
12            head2_labels.extend(head_2_labels.cpu().numpy())
13    X = np.vstack(all_features)
14    y_head1 = np.array(head1_labels)
15    y_head2 = np.array(head2_labels)
16    return X, y_head1, y_head2
17
```

Σχήμα Β.25: Η συνάρτηση feature_extraction_ML