Τίτλος Εργασίας:

**‘Ταξινόμηση Θαλάσσιων Ζώων με Χρήση CNNs και Transfer Learning’**

**Ομάδα:**

**Κατσούλη Καλλιόπη,** 2022202404006

**Γιουλάτου Άννα- Βασιλική,** 2022202404001

**Εισαγωγή**

Η θαλάσσια ζωή περιλαμβάνει όλους τους οργανισμούς που ζουν σε αλμυρά ή υφάλμυρα υδάτινα περιβάλλοντα, όπως οι ωκεανοί, οι θάλασσες, τα λιμνοθάλασσες και οι εκβολές ποταμών. Αυτή η τεράστια βιοποικιλότητα εκτείνεται από μικροσκοπικούς μονοκύτταρους μικροοργανισμούς, όπως τα κυανοβακτήρια και το φυτοπλαγκτόν, μέχρι και γιγάντια θηλαστικά, όπως η γαλάζια φάλαινα. Οι ωκεανοί καταλαμβάνουν περίπου το 90% του διαθέσιμου όγκου διαβίωσης του πλανήτη και αποτέλεσαν τη γενέτειρα της ζωής στη Γη.

Περισσότερα από 240.000 θαλάσσια είδη έχουν ήδη καταγραφεί, ενώ εκτιμάται ότι εκατομμύρια ακόμα παραμένουν άγνωστα. Οι οργανισμοί αυτοί επιτελούν κρίσιμες λειτουργίες για το πλανητικό οικοσύστημα, όπως η παραγωγή οξυγόνου μέσω της φωτοσύνθεσης και η ρύθμιση του κύκλου του άνθρακα. Ανάλογα με τον ρόλο τους στην τροφική αλυσίδα, διακρίνονται σε αυτότροφους (π.χ. φύκη, φυτοπλαγκτόν) και ετερότροφους οργανισμούς (π.χ. ψάρια, θαλάσσια θηλαστικά, μύκητες).

Η ανάγκη για καταγραφή, κατανόηση και παρακολούθηση αυτής της βιοποικιλότητας καθίσταται ολοένα και πιο επιτακτική λόγω της κλιματικής αλλαγής, της ρύπανσης και της υπεραλίευσης, που απειλούν σοβαρά τα θαλάσσια οικοσυστήματα. Η αυτόματη αναγνώριση και ταξινόμηση θαλάσσιων οργανισμών, μέσω τεχνικών μηχανικής μάθησης και βαθιάς μάθησης (deep learning), αποτελεί πλέον ένα ισχυρό εργαλείο για τη βιολογική έρευνα και τη διαχείριση της θαλάσσιας ζωής.

Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη και αξιολόγηση νευρωνικών δικτύων για την ταξινόμηση εικόνων θαλάσσιων οργανισμών, με στόχο τη βελτίωση της ακρίβειας και της αποδοτικότητας στην αναγνώριση ειδών από φωτογραφικό υλικό. Προχωρήσαμε σε ανάπτυξη και αξιολόγηση μοντέλων Deep Learning για την αυτόματη αναγνώριση και ταξινόμηση εικόνων που απεικονίζουν πέντε κατηγορίες θαλάσσιων ζώων: **JellyFish, SeaUrchins, Sharks, Starfish** και **Turtles**. Αρχικά, μελετήθηκαν δύο προσαρμοσμένες αρχιτεκτονικές συνελικτικών νευρωνικών δικτύων (CNN01 και CNN02), οι οποίες εκπαιδεύτηκαν από την αρχή με διαφορετικούς συνδυασμούς υπερπαραμέτρων (learning rate, batch size, dropout, augmentation κ.ά.). Επιπρόσθετα, εφαρμόστηκε και τεχνική μεταφοράς μάθησης (transfer learning) με τη χρήση του προεκπαιδευμένου μοντέλου **ResNet18**, προκειμένου να διερευνηθεί κατά πόσο μπορεί να επιτευχθεί καλύτερη απόδοση με λιγότερη εκπαίδευση και καλύτερη γενίκευση στα δεδομένα. Όλα τα μοντέλα αξιολογήθηκαν ως προς την απόδοσή τους σε επιλεγμένο dataset εικόνων θαλάσσιων ζώων, με τη χρήση κατάλληλων μετρικών όπως ακρίβεια, απώλεια, και καμπύλες εκπαίδευσης.

Συνοπτικά, για την εκπαίδευση και την αξιολόγηση των μοντέλων χρησιμοποιήθηκε ένα υποσύνολο από το δημόσια διαθέσιμο dataset **"Sea Animals Dataset"**, ενώ η προσέγγιση περιλάμβανε τόσο απλά συνελικτικά νευρωνικά δίκτυα (CNN01 και CNN02), όσο και τη μέθοδο μεταφοράς μάθησης με χρήση του **ResNet18**.

Οι στόχοι της εργασίας ήταν:

* Η σύγκριση της απόδοσης διαφορετικών μοντέλων και παραμετροποιήσεων.
* Η διερεύνηση της επίδρασης παραμέτρων όπως το learning rate και το batch size.
* Η αξιολόγηση της αποτελεσματικότητας της μεταφοράς μάθησης στο ίδιο πρόβλημα.

**Δομή Έργου και Προεπεξεργασία**

Το έργο οργανώθηκε σε τρία βασικά μέρη:

1. **Dataset και Διαχείριση Αρχείων.**
2. **Κώδικας Εκπαίδευσης και Μοντέλα.**
3. **Καταγραφή και Αποθήκευση Αποτελεσμάτων.**

Ο κώδικας εκτελέστηκε και δοκιμάστηκε στο **Google Colab**, με αποθήκευση των checkpoints και των γραφημάτων στο **Google Drive**, διευκολύνοντας τη συνεργατική εργασία και την παρακολούθηση προόδου.

**Διαχείριση Έργου και GitHub Repository**

Η υλοποίηση του έργου πραγματοποιήθηκε με οργανωμένο τρόπο, μέσω χρήσης του Git και της πλατφόρμας **GitHub**, ώστε να διασφαλίζεται ο σωστός έλεγχος εκδόσεων, η συνεργασία και η τεκμηρίωση των βημάτων. Όλο το υλικό της εργασίας είναι διαθέσιμο μέσω **public GitHub repository**, το οποίο περιλαμβάνει:

* **Κώδικα εκπαίδευσης** των μοντέλων CNN01, CNN02 και του ResNet18 μέσω transfer learning, με δυνατότητα παραμετροποίησης μέσω ορισμάτων.
* **Κώδικα πειραμάτων και tuning**, που επιτρέπει τη δοκιμή διαφορετικών συνδυασμών learning rate, batch size και dropout, καθώς και την ενεργοποίηση του μηχανισμού early stopping με patience.
* **Αποτελέσματα και καμπύλες απόδοσης**, αποθηκευμένες αυτόματα σε μορφή .csv και .png.
* Το **τελικό report** και την **παρουσίαση**, ως μέρος της τελικής τεκμηρίωσης.

Ο σύνδεσμος του repository παρατίθεται στην αρχή του εγγράφου και στο README του GitHub.

**Δομή Github**

**MAIN FOLDER: see-animal-dl**

data/

notebooks/

slides/

src/

dataset.py

models.py

train.py

evaluate.py

README.md

requirements.txt

github project: <https://github.com/pkatpk/see-animal-dl>

**Dataset**

Το χρησιμοποιούμενο σύνολο δεδομένων περιλαμβάνει εικόνες πέντε κατηγοριών θαλάσσιων ζώων. Είναι ανοιχτά διαθέσιμο στο Kaggle, στον ακόλουθο σύνδεσμο: <https://www.kaggle.com/code/vencerlanz09/sea-animals-classification-using-efficeintnetb7/input>.

Τα δεδομένα οργανώθηκαν σε υποφακέλους ανά κλάση, ενώ στη συνέχεια πραγματοποιήθηκε split σε τρία σύνολα:

* **Training set.**
* **Validation set.**
* **Test set.**

Η οργάνωση έγινε με αναλογία περίπου **70% train**, **20% validation**, και **10% test**.

**Εξερεύνηση και Επιλογή Δεδομένων (EDA)**

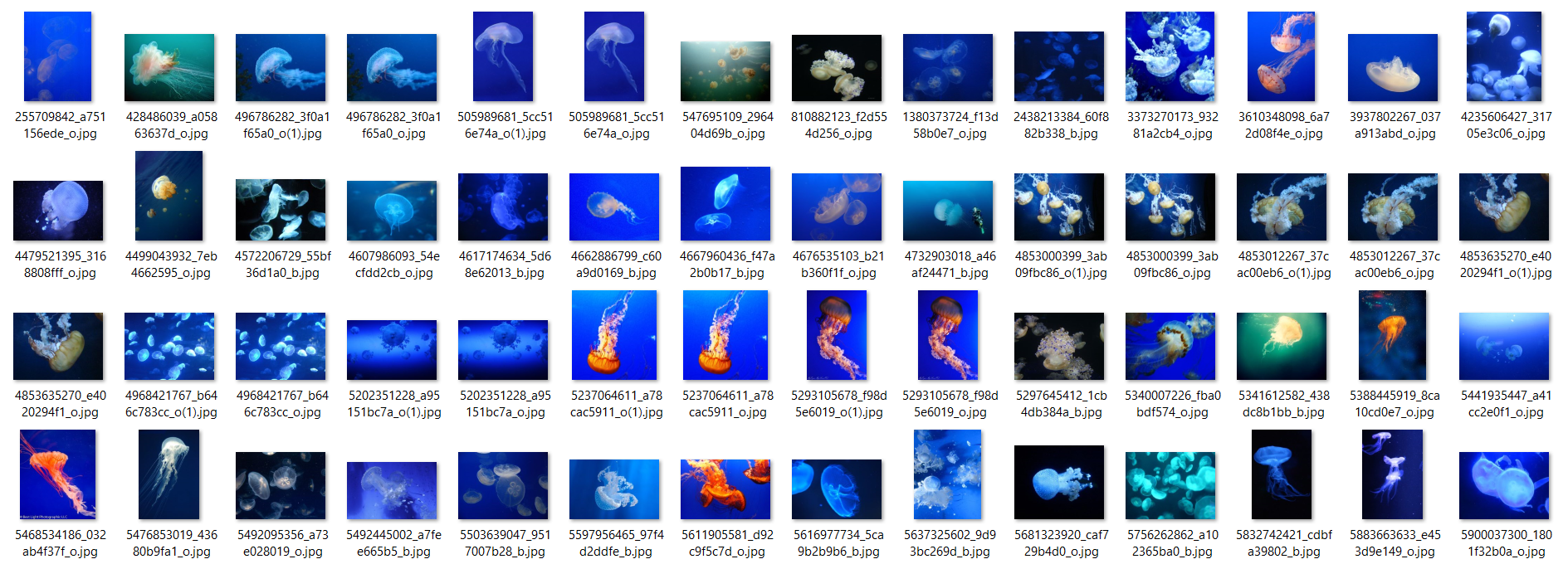
Το αρχικό σύνολο δεδομένων αποτελούνταν από **13.666 εικόνες** κατανεμημένες σε **23 διαφορετικές κατηγορίες θαλάσσιων οργανισμών**. Κατά την αρχική διερεύνηση των δεδομένων (EDA) διαπιστώθηκε ότι δεν υπήρχαν αλλοιωμένα (corrupt) αρχεία εικόνας, ωστόσο εντοπίστηκαν ορισμένα παραδείγματα με λανθασμένη ή ασυνεπή επισήμανση (mislabeling).



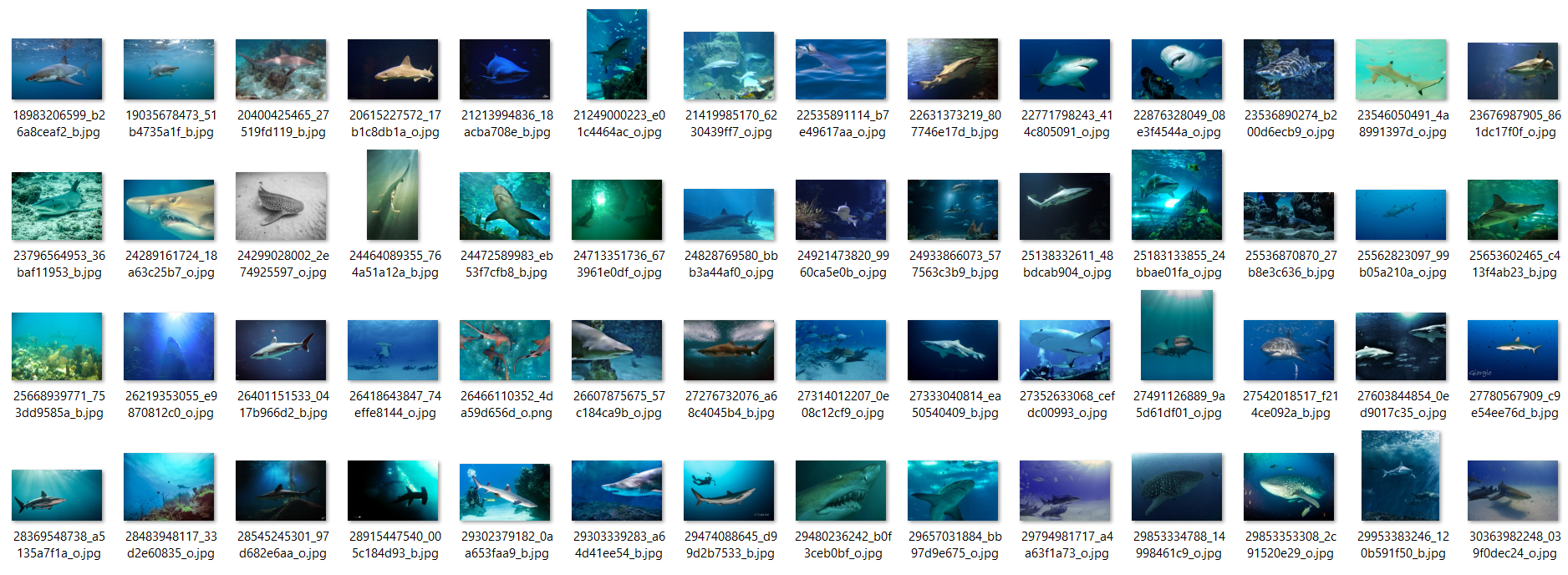
Λόγω των περιορισμών σε υπολογιστικούς πόρους και για την επίτευξη πιο στοχευμένων πειραμάτων, επιλέχθηκε ένα υποσύνολο **5 κλάσεων**, οι οποίες παρουσίαζαν επαρκή πλήθος παραδειγμάτων και υψηλή μεταξύ τους οπτική διαφοροποίηση. Οι κατηγορίες που επιλέχθηκαν είναι:

* **Turtles.**
* **Starfish.**
* **Sharks.**
* **Sea Urchins.**
* **Jellyfish.**

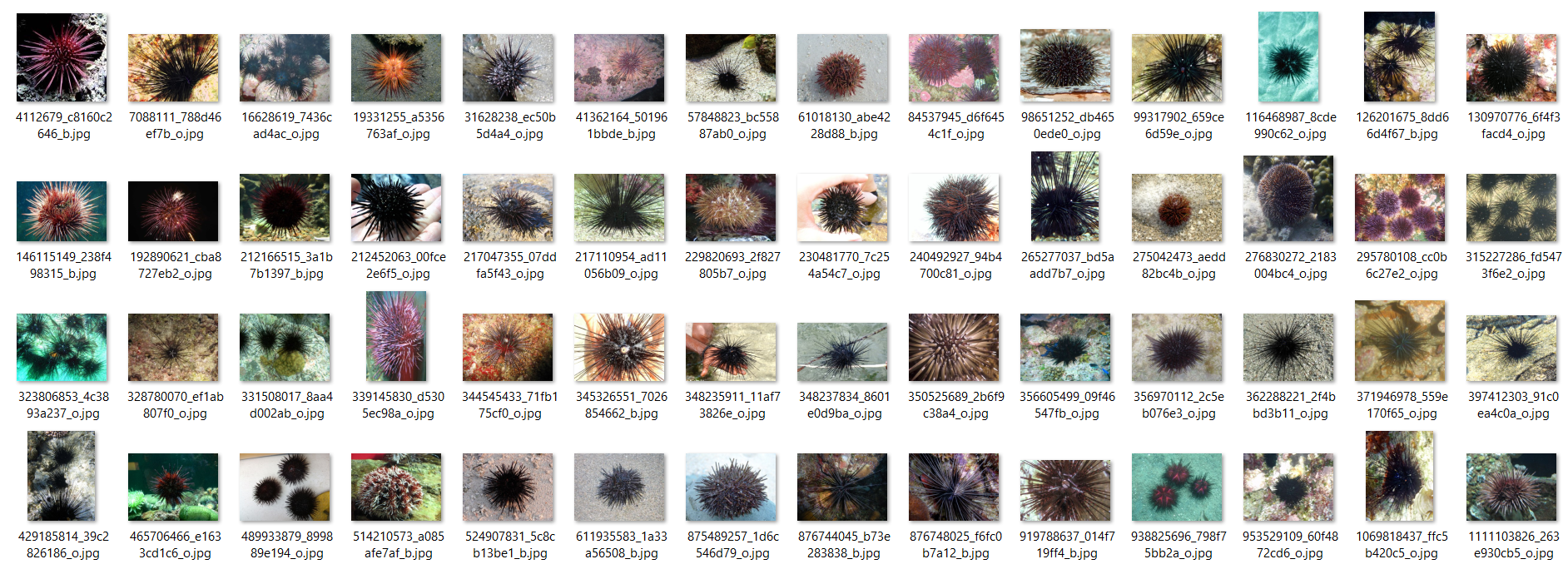




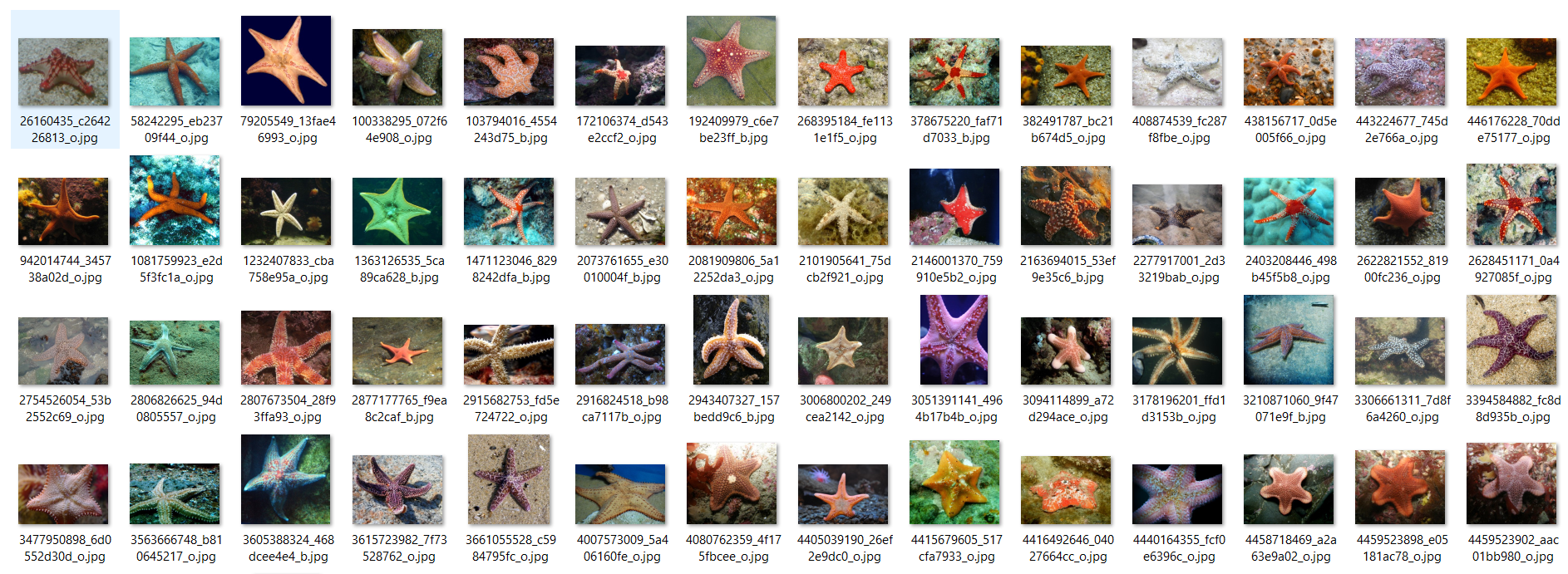
**Jellyfishes\_cleaned**



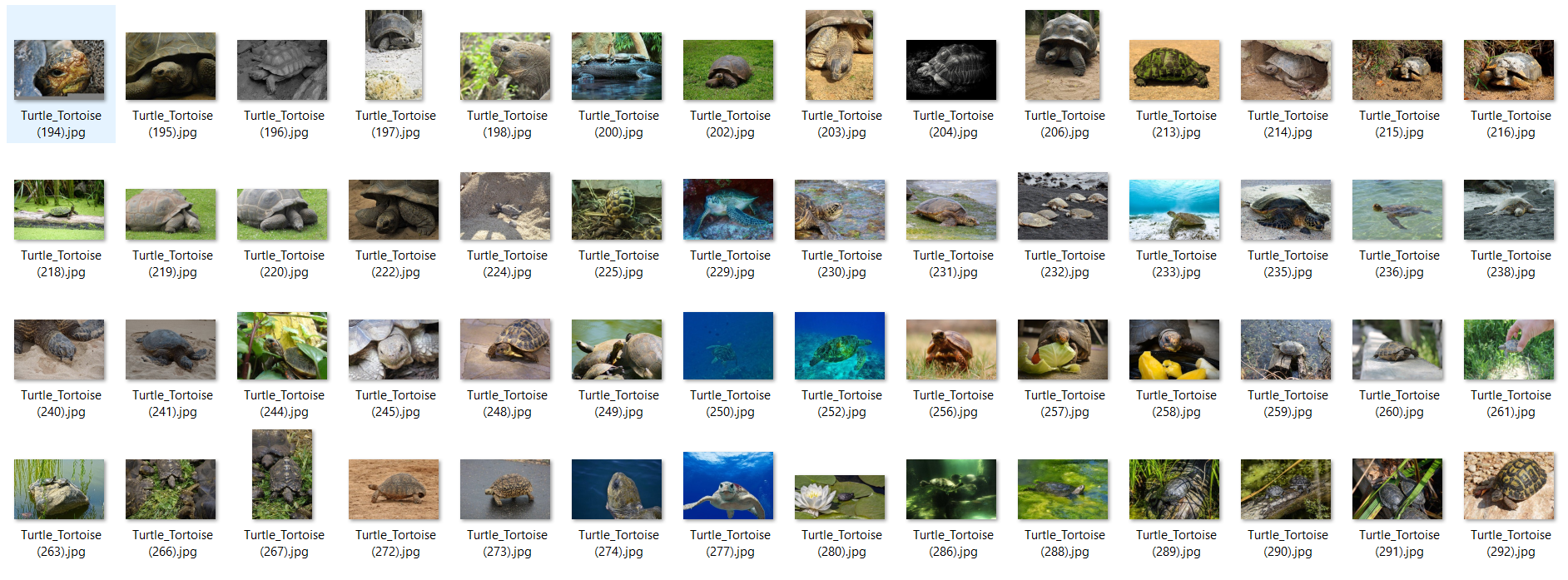
**Sharks\_cleaned**



**Sea Urchins\_cleaned**



**Starfish\_cleaned**



**Turtles\_cleaned**

Το τελικό υποσύνολο δεδομένων περιλαμβάνει **2.693 εικόνες**, όπως παρουσιάζεται στην Εικόνα 2.2. Το σύνολο αυτό επεξεργάστηκε και οργανώθηκε στον φάκελο see-animals-dataset\_cleaned, με ομοιόμορφη δομή φακέλων και εικόνων έτοιμη προς εκπαίδευση.

Το συνολικό μέγεθος του τελικού dataset είναι περίπου **69MB**, γεγονός που το καθιστά κατάλληλο για ταχεία εκπαίδευση σε πειραματικά μοντέλα σε τοπικό περιβάλλον ή μέσω notebook περιβάλλοντος όπως το Colab.

A table of numbers with text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Προεπεξεργασία Δεδομένων & Δημιουργία DataLoaders**

Η σωστή προετοιμασία των δεδομένων αποτελεί κρίσιμο στάδιο στην ανάπτυξη ενός αποτελεσματικού μοντέλου ταξινόμησης εικόνων. Στην παρούσα εργασία, η διαχείριση των δεδομένων και η δημιουργία των απαραίτητων loaders πραγματοποιήθηκαν μέσω του αρχείου dataset.py, το οποίο σχεδιάστηκε για να είναι πλήρως αυτοματοποιημένο, επαναλήψιμο και παραμετροποιήσιμο.

**Διαδικασία Οργάνωσης του Dataset**

Η βασική πηγή δεδομένων ήταν ο φάκελος see\_animals\_dataset, ο οποίος περιείχε 23 κλάσεις θαλάσσιων οργανισμών με συνολικά **13.666 εικόνες**. Ωστόσο, για σκοπούς απλοποίησης και καλύτερου ελέγχου του πειραματικού σεναρίου, επιλέχθηκαν **5 αντιπροσωπευτικές κλάσεις** (turtles, starfish, sharks, sea urchins, jellyfish) οι οποίες περιλάμβαναν **2.693 εικόνες** στο σύνολο.

Το script dataset.py ακολουθεί την εξής λογική:

1. **Ανάγνωση του αρχικού dataset:**
   * Εντοπίζει όλες τις υποκατηγορίες (κλάσεις) του αρχικού φακέλου data/see\_animals\_dataset.
2. **Δημιουργία φακέλου "split":**
   * Δημιουργείται ένας νέος φάκελος data/see\_animals\_dataset\_split, όπου θα οργανωθούν οι εικόνες σε υποφακέλους για train, validation και test.
   * Αν ο φάκελος \_split υπάρχει ήδη, η διαδικασία δεν επαναλαμβάνεται (για αποφυγή περιττού χρόνου επεξεργασίας) και προχωρά μόνο στη φόρτωση των δεδομένων.
3. **Διαχωρισμός συνόλων (split):**
   * Για κάθε κατηγορία, γίνεται:
     + Ανάκτηση όλων των .jpg εικόνων
     + Αναδιάταξή τους σε τυχαία σειρά (με χρήση seed για αναπαραγωγιμότητα)
     + Υπολογισμός κατανομής:
       - **70%** → Training set
       - **20%** → Validation set
       - **10%** → Test set
     + Αντιγραφή των αρχείων στους αντίστοιχους φακέλους:

data/see\_animals\_dataset\_split/train/<class>/image.jpg

data/see\_animals\_dataset\_split/val/<class>/image.jpg

data/see\_animals\_dataset\_split/test/<class>/image.jpg

1. **Παράμετροι διαχωρισμού (configurable):**  
   Οι παρακάτω παράμετροι μπορούν να προσαρμοστούν μέσω της συνάρτησης get\_dataloaders():

|  |  |  |
| --- | --- | --- |
| **Παράμετρος** | **Default** | **Περιγραφή** |
| val\_pct | 0.2 | Ποσοστό δεδομένων για validation set |
| test\_pct | 0.1 | Ποσοστό για test set |
| seed | 42 | Εξασφάλιση αναπαραγωγιμότητας |
| with\_test | True | Αν πρέπει να δημιουργηθεί test loader |

**Δημιουργία DataLoaders**

Αφού πραγματοποιηθεί το split, η συνάρτηση get\_dataloaders() επιστρέφει τους απαραίτητους φορτωτές δεδομένων (DataLoader) για εκπαίδευση, επικύρωση και δοκιμή. Ορίζονται επίσης και οι μετασχηματισμοί (transforms) που εφαρμόζονται στις εικόνες.

**Χρήση της συνάρτησης:**

train\_loader, val\_loader, test\_loader, class\_names = get\_dataloaders(

root\_dir='data/see\_animals\_dataset\_split',

img\_size=128,

batch=32,

seed=42,

with\_test=True

)

**Τρόπος Εκτέλεσης του dataset.py**

Η εκτέλεση πραγματοποιήθηκε τοπικά μέσω εικονικού περιβάλλοντος Python (virtual environment). Η διαδικασία έχει ως εξής:

# Ενεργοποίηση virtual environment

C:\Users\Klelia>C:\dlenv\Scripts\activate

# Πλοήγηση στον φάκελο του project

cd Desktop\see-animal-dl

# Εκτέλεση του dataset split και δημιουργία των loaders

python -m src.dataset --root data/see\_animals\_dataset

Η σωστή κατανομή των δεδομένων και η εφαρμογή κατάλληλων μετασχηματισμών αποτέλεσαν βασικά βήματα για την επιτυχή εκπαίδευση των μοντέλων CNN. Η αναπαραγωγιμότητα και η παραμετροποίηση της διαδικασίας split μέσω του dataset.py προσέδωσαν σαφή οργανωτική δομή και ευελιξία στον πειραματισμό.

**Προετοιμασία Εικόνας**

Η προετοιμασία των εικόνων ακολούθησε συγκεκριμένο προκαθορισμένο pipeline το οποίο εφαρμόστηκε εξίσου σε όλα τα υποσύνολα του dataset (train, validation, test). Τα βασικά στάδια ήταν τα εξής:

* **Ανακατασκευή Διαστάσεων (Resizing):** Όλες οι εικόνες επαναδιαστασιοποιήθηκαν σε τετράγωνη μορφή. Συγκεκριμένα, χρησιμοποιήθηκαν διαστάσεις **128×128** για τα custom μοντέλα **CNN01** και **CNN02**, ενώ για το μοντέλο **ResNet18** χρησιμοποιήθηκαν οι προκαθορισμένες διαστάσεις **224×224**, για να είναι συμβατό με την αρχιτεκτονική.
* **Μετατροπή σε PyTorch Tensors:** Η κάθε εικόνα μετατράπηκε σε tensor με τη χρήση της συνάρτησης ToTensor(), εξασφαλίζοντας έτσι τη σωστή είσοδο στο δίκτυο.
* **Εφαρμογή Τransformations:** Οι παραπάνω μετασχηματισμοί εφαρμόζονταν μέσω transforms. Compose() σε όλο το dataset κατά τη φόρτωσή του, όπως ορίζεται στο αρχείο dataset.py.
* **Διαχωρισμός Dataset:** Ο διαχωρισμός των δεδομένων σε σύνολα **εκπαίδευσης**, **επικύρωσης** και **δοκιμής** πραγματοποιήθηκε με αναλογία **70/20/10** για κάθε κλάση χωριστά.

**Πειραματική Διαδικασία & Εκπαίδευση**

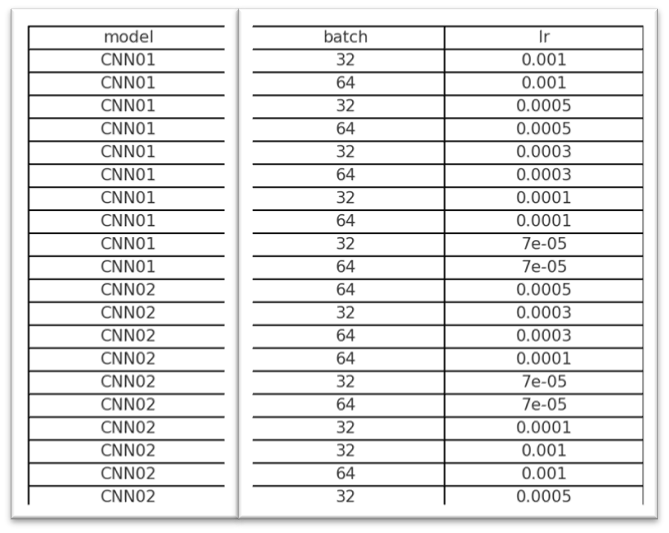
Για την εκπαίδευση των μοντέλων εφαρμόστηκαν οι εξής τεχνικές και παραμετροποιήσεις:

* **Optimizer:** Χρησιμοποιήθηκε ο βελτιστοποιητής **Adam**, που προσφέρει ταχεία και σταθερή σύγκλιση.
* **Loss Function:** Για τον υπολογισμό του σφάλματος επιλέχθηκε η **CrossEntropyLoss**, κατάλληλη για προβλήματα πολυκλασικής ταξινόμησης.
* **Early Stopping:** Ενεργοποιήθηκε μηχανισμός **early stopping** με patience = 3, ώστε να αποφεύγεται η υπερπροσαρμογή και να περιορίζεται ο χρόνος εκπαίδευσης.

**Παραμετρικές Δοκιμές**

Κατά την πειραματική φάση, διερευνήθηκαν οι επιδράσεις διαφορετικών υπερπαραμέτρων:

* **Learning Rates:** Δοκιμάστηκαν οι τιμές 0.0010, 0.0001, 0.0003, 0.0005, 0.00007.
* **Batch Sizes:** 32 και 64.
* **Σύνολο Πειραμάτων:** Πραγματοποιήθηκαν τουλάχιστον **20 runs** σε CNN01 και CNN02 για αναζήτηση των βέλτιστων παραμέτρων.



Καθ’ όλη τη διάρκεια των πειραμάτων γινόταν καταγραφή της **ακρίβειας (Accuracy)** και της **απώλειας (Loss)** για τα σύνολα εκπαίδευσης και επικύρωσης, ενώ δημιουργούνταν αντίστοιχα διαγράμματα σύγκλισης (Epoch Curves) για οπτική παρακολούθηση της προόδου.

**Αρχιτεκτονικές Μοντέλων και Εκπαιδευτικό Πρωτόκολλο**

**Γενική Επισκόπηση**

Η παρούσα εργασία βασίστηκε σε τρεις διαφορετικές αρχιτεκτονικές συνελικτικών νευρωνικών δικτύων (Convolutional Neural Networks – CNNs): **CNN01**, **CNN02**, και **ResNet18 (transfer learning)**. Η ανάπτυξη των custom μοντέλων πραγματοποιήθηκε στο αρχείο models.py, ενώ η διαδικασία εκπαίδευσης και αξιολόγησης οργανώθηκε μέσω του script train.py. Οι δοκιμές υλοποιήθηκαν κυρίως στο **Google Colab**, αξιοποιώντας GPU για επιτάχυνση και χρήση του Google Drive για αποθήκευση πειραμάτων.

**To CNN01 – Ελαφριά, baseline αρχιτεκτονική**

Το **CNN01** αποτέλεσε την **πρώτη απόπειρα υλοποίησης** ενός βασικού CNN για το πρόβλημα της πολυκλασικής ταξινόμησης εικόνων. Σχεδιάστηκε με γνώμονα τη **χαμηλή υπολογιστική πολυπλοκότητα**, τη **γρήγορη εκπαίδευση** και τη **δυνατότητα σύγκρισης ως baseline**. Η αρχιτεκτονική του ακολουθεί τυπική διάταξη, χωρίς περιττές βελτιστοποιήσεις, και προορίζεται να αποτελέσει σημείο αναφοράς για μελλοντικά μοντέλα.

**Αρχιτεκτονική**

Το μοντέλο αποτελείται από τρία συνελικτικά blocks και έναν ταξινομητή (classifier):

* **Συνελικτικά blocks (convolutional layers):**
  1. Conv(3→8) → BatchNorm → ReLU → MaxPooling.
  2. Conv(8→16) → BatchNorm → ReLU → MaxPooling.
  3. Conv(16→32) → BatchNorm → ReLU → MaxPooling.

Η χρήση **Batch Normalization** εξασφαλίζει σταθερότητα στην εκπαίδευση, ενώ η προοδευτική αύξηση των φίλτρων επιτρέπει στο δίκτυο να μαθαίνει πιο αφηρημένες αναπαραστάσεις.

* **Classifier block:**  
  Flatten → Linear (32×H×W → 128) → ReLU → Dropout (0.4) → Linear → Softmax.

Η χρήση **Dropout** αποσκοπεί στην αποφυγή υπερπροσαρμογής (overfitting), ενώ ο πλήρως συνδεδεμένος ταξινομητής μετατρέπει το χωρικά επεξεργασμένο output σε πιθανότητες κατηγορίας.

**Ρόλος του CNN01**

Το μοντέλο αυτό σχεδιάστηκε ως **σημείο εκκίνησης**, επιτρέποντας την κατανόηση της ροής δεδομένων, των διαδικασιών εκπαίδευσης και της συμπεριφοράς των βασικών υπερπαραμέτρων. Παρά την απλότητά του, επέδειξε **αξιοπρεπή απόδοση (~75%)**, παρέχοντας σαφές baseline για σύγκριση με πιο σύνθετες προσεγγίσεις.

**Εκπαιδευτικό Πρωτόκολλο (train.py)**

Το script train.py διαχειρίζεται τη **φόρτωση δεδομένων**, την **εκπαίδευση/επικύρωση μοντέλων**, και την **καταγραφή αποτελεσμάτων**. Περιλαμβάνει παραμετροποιήσιμη διεπαφή μέσω **flags**, επιτρέποντας αλλαγές αρχιτεκτονικής και υπερπαραμέτρων.

**Βασικές λειτουργίες:**

* Εκτέλεση training/validation loop (με train\_epoch και valid\_epoch).
* Καταγραφή αποτελεσμάτων στο experiments.csv.
* Αποθήκευση του βέλτιστου μοντέλου (με βάση το validation loss).
* Παραγωγή διαγραμμάτων (loss/accuracy curves).
* Εκτέλεση σε **τοπικό ή Colab περιβάλλον** με πλήρη υποστήριξη Google Drive.

**Default τιμές:**

|  |  |  |
| --- | --- | --- |
| **Flag** | **Default Τιμή** | **Σχόλιο** |
| --img | 128 | Μέγεθος εισόδου εικόνας |
| --batch | 32 | Batch size |
| --epochs | 20 | Εποχές εκπαίδευσης |
| --lr | 0.001 | Learning rate |
| --output | checkpoints/ | Φάκελος αποθήκευσης μοντέλων |

**Πρώτη Εκπαίδευση του CNN01 – Ανάλυση**

Η πρώτη πειραματική εκτέλεση πραγματοποιήθηκε με τις εξής παραμέτρους:

python src/train.py \

--model CNN01 \

--data-dir data/see\_animals\_dataset\_split \

--img 128 \

--batch 32 \

--epochs 10 \

--lr 0.001 \

--notes "1st Colab run with CNN01 - 5 classes"

A screenshot of a computer

AI-generated content may be incorrect.

Η εκπαίδευση έγινε σε περιβάλλον Google Colab με GPU και κράτησε περίπου **10 λεπτά**. Τα αποτελέσματα (αρχεία .pt, .csv, .png) αποθηκεύτηκαν αυτόματα στο Google Drive.

**Outputs:**

* CNN01\_curve\_1.png: Καμπύλες loss & accuracy.
* CNN01\_stats\_1.csv: Αναλυτικά στατιστικά ανά εποχή.
* best\_CNN01\_1.pt: Το καλύτερο μοντέλο (βάσει validation loss).

A graph of different colored lines

AI-generated content may be incorrect.

**Ανάλυση Απόδοσης:**

|  |  |
| --- | --- |
| **Μετρική** | **Παρατήρηση** |
| **Training Loss** | Σταθερή μείωση → Υγιής εκμάθηση |
| **Validation Loss** | Πτώση στα πρώτα epochs, μετά "πριόνιζε" → Ενδείξεις αστάθειας |
| **Training Acc.** | Συνεχής αύξηση |
| **Validation Acc.** | Σταθερή αλλά με διακυμάνσεις (~75%) → Όχι σοβαρό overfitting |

**Ερμηνεία:**

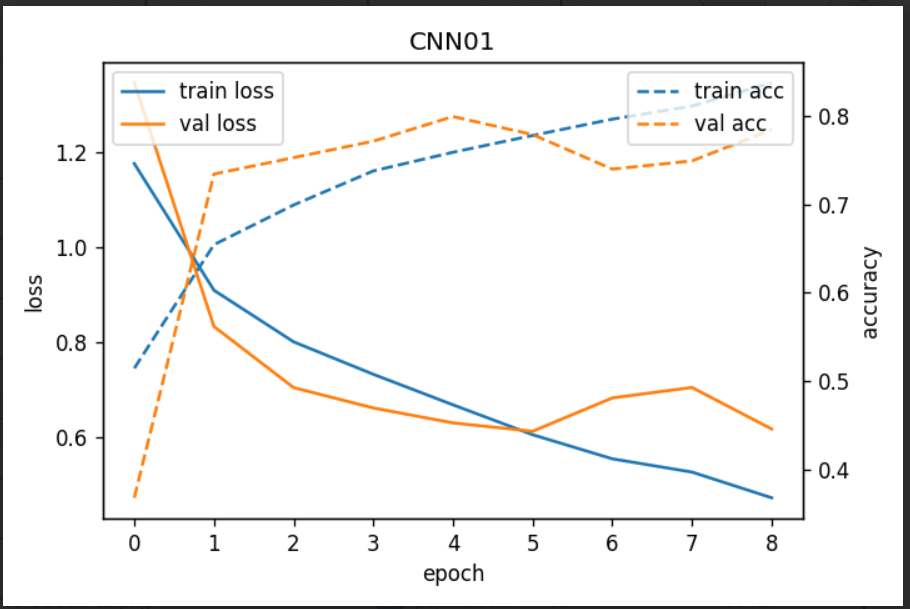
|  |  |  |
| --- | --- | --- |
| **Παρατήρηση** | **Ερμηνεία** | **Πιθανή Βελτίωση** |
| Val loss “πριονίζει” | Αστάθεια, ανεπαρκής γενίκευση | Learning rate scheduler, CNN02 |
| Train acc >> Val acc | Ελαφρύ overfitting | Dropout, EarlyStopping |
| Val acc ~75% | Καλή αρχική απόδοση για baseline μοντέλο | Data augmentation |

**Συμπέρασμα:**

Το CNN01 αποτέλεσε ένα **πολύτιμο πρώτο βήμα**, αποδίδοντας **ικανοποιητικά** χωρίς υπερβολικό υπολογιστικό κόστος. Παράλληλα, ανέδειξε τις **δομικές του αδυναμίες**, όπως την αστάθεια του validation loss και το περιορισμένο capacity, γεγονός που **δικαιολογεί τη μετάβαση σε βαθύτερες αρχιτεκτονικές** – όπως το **CNN02** ή/και **τεχνικές μεταφοράς μάθησης**, όπως η **ResNet18**.

**Κορυφαία εκτέλεση του CNN01**

Η καλύτερη επίδοση για το CNN01, ήταν η ακόλουθη:



Ανάμεσα στα διάφορα runs του απλού συνελικτικού δικτύου CNN01, η καλύτερη επίδοση σημειώθηκε κατά την εκτέλεση με τα εξής χαρακτηριστικά:

* **Learning Rate:** 0.0003
* **Batch Size:** 64
* **Epochs:** 9
* **Validation Accuracy:** 78.4%
* **Validation Loss:** 0.612
* **Συνολικός Χρόνος Εκπαίδευσης:** ~4 λεπτά (244 sec)

Το συγκεκριμένο run (Run ID: 20250614\_111719) επέδειξε τη βέλτιστη ισορροπία μεταξύ ακρίβειας και σταθερότητας, αναδεικνύοντας το βέλτιστο σετ υπερπαραμέτρων για το CNN01 όσον αφορά την απόδοση στο validation set.

**Ανάλυση Εκπαίδευσης**

Η εκπαίδευση εξελίχθηκε ομαλά, με σταθερή μείωση του training loss ανά epoch. Η καμπύλη validation loss εμφάνισε σχετικά ελεγχόμενες διακυμάνσεις, χωρίς απότομες αυξομειώσεις, γεγονός που υποδηλώνει επαρκή γενίκευση για την κλίμακα του προβλήματος. Ο ρυθμός μάθησης των 0.0003 αποδείχθηκε επαρκώς μικρός ώστε να επιτρέπει την ομαλή προσαρμογή των βαρών, χωρίς φαινόμενα απότομης σύγκλισης ή αστάθειας.

Η **validation accuracy** αυξήθηκε προοδευτικά, φτάνοντας το μέγιστο (~78.4%) στο τέλος των 9 εποχών, ενώ η **training accuracy** παρέμεινε επίσης υψηλή και συγκλίναμενη, στοιχείο που ενισχύει την υπόθεση ότι το μοντέλο έμαθε με συνέπεια χωρίς εμφανές overfitting.

**Ερμηνεία Καμπύλης Loss/Accuracy**

Η αντίστοιχη γραφική απεικόνιση παρουσιάζει τα εξής βασικά σημεία:

* Η **training loss** μειώνεται σταθερά από epoch 1 έως 9, γεγονός που υποδεικνύει αποτελεσματική προσαρμογή.
* Η **validation loss** εμφανίζει «πριονωτή» μορφή, με εναλλαγές ανόδου και καθόδου — σύνηθες φαινόμενο σε μικρού μεγέθους δίκτυα χωρίς aggressive regularization.
* Οι **καμπύλες accuracy** δείχνουν σταθερή αύξηση τόσο για το training όσο και για το validation set, χωρίς να αποκλίνουν υπερβολικά — στοιχείο που υποδηλώνει ότι το μοντέλο δεν υπερεκπαιδεύτηκε.

**Συμπεράσματα**

Η κορυφαία εκτέλεση του CNN01 επιβεβαίωσε τη λειτουργικότητα και τη σταθερότητα της αρχιτεκτονικής του στο συγκεκριμένο πρόβλημα ταξινόμησης. Παρά την απλότητά του, το μοντέλο κατάφερε να αποδώσει με ακρίβεια σχεδόν 78.4% στο validation set, επίδοση ιδιαίτερα ικανοποιητική για baseline.

Η εκτέλεση αυτή αποτελεί ισχυρό σημείο αναφοράς (benchmark), καθώς προσφέρει υψηλή ακρίβεια με εξαιρετικά χαμηλή υπολογιστική απαίτηση. Παράλληλα, οι παρατηρήσεις από το validation loss υποδεικνύουν ότι υπάρχει περιθώριο βελτίωσης στη γενίκευση, πιθανώς μέσω αύξησης της χωρητικότητας του μοντέλου ή εισαγωγής regularization (π.χ. dropout, early stopping).

Η ανάλυση της εκτέλεσης αυτής αποτέλεσε το εφαλτήριο για τον σχεδιασμό πιο σύνθετων αρχιτεκτονικών όπως το CNN02, αλλά και για την επιλογή τεχνικών μεταφοράς μάθησης που ακολουθούν στη συνέχεια της μελέτης.

**Αξιολόγηση του CNN02**

Μετά την αρχική πειραματική φάση με το απλό μοντέλο **CNN01**, η μελέτη επεκτάθηκε με την εισαγωγή του **CNN02**, ενός βαθύτερου και πιο ισχυρού συνελικτικού νευρωνικού δικτύου. Το **CNN02** σχεδιάστηκε με στόχο να προσφέρει αυξημένη χωρητικότητα και δυνατότητα αναπαράστασης, ικανή να ανιχνεύσει πιο σύνθετα μοτίβα, ενδεχομένως εις βάρος της υπολογιστικής απόδοσης.

Η ενότητα αυτή αφορά **το πρώτο τρέξιμο του CNN02**, που πραγματοποιήθηκε σε συνθήκες αντίστοιχες με αυτές του CNN01, προκειμένου να διασφαλιστεί δίκαιη σύγκριση των επιδόσεων.

**Αρχιτεκτονική CNN02**

Η αρχιτεκτονική του CNN02 αποτελείται από:

* **Τρία συνελικτικά blocks (convolutional blocks):**
  1. Conv(3→32) → BatchNorm → ReLU → MaxPooling
  2. Conv(32→64) → BatchNorm → ReLU → MaxPooling
  3. Conv(64→128) → BatchNorm → ReLU → MaxPooling
* **Classifier:**  
  Flatten → Linear(… → 256) → ReLU → Dropout(0.5) → Linear → Softmax

Σε σχέση με το CNN01, το CNN02 διαθέτει περισσότερα φίλτρα σε κάθε επίπεδο και διπλάσιο αριθμό νευρώνων στη fully connected στρώση, γεγονός που ενισχύει τη δυνατότητα μάθησης πιο αφηρημένων και λεπτομερών χαρακτηριστικών. Το Dropout αυξήθηκε στο **0.5** για την καταπολέμηση πιθανών φαινομένων υπερπροσαρμογής (overfitting).

**Συνθήκες Εκπαίδευσης – Πρώτο Run**

Το πρώτο run του CNN02 πραγματοποιήθηκε με τις ακόλουθες ρυθμίσεις:

!python src/train.py \

--model CNN02 \

--data-dir data/see\_animals\_dataset\_split \

--img 128 \

--batch 32 \

--epochs 10 \

--lr 0.001 \

--notes "2nd Colab run with CNN02 - 5 classes"

A screenshot of a computer

AI-generated content may be incorrect.

* **Dataset**: 5 κλάσεις από το see\_animals\_dataset\_split
* **Εκπαίδευση**: Google Colab (GPU)
* **Batch Size**: 32
* **Epochs**: 10
* **Learning Rate**: 0.001

Το πείραμα καταγράφηκε στο αρχείο experiments.csv, ενώ παράλληλα δημιουργήθηκαν και αποθηκεύτηκαν τα παρακάτω outputs:

* Καμπύλες loss και accuracy ανά epoch (CNN02\_curve.png).
* Checkpoint με το καλύτερο validation loss (best\_CNN02.pt).
* Αναλυτικά στατιστικά ανά εποχή (CNN02\_stats.csv).

A graph of different colored lines

AI-generated content may be incorrect.

**Ανάλυση Εκπαίδευσης (Training Metrics)**

Το CNN02 παρουσίασε αισθητά **βελτιωμένη συμπεριφορά ως προς την εκμάθηση** σε σύγκριση με το CNN01:

* Το **training loss** μειωνόταν σταθερά και γρήγορα, υποδηλώνοντας αποτελεσματική εκμάθηση του συνόλου εκπαίδευσης.
* Η **training accuracy** αυξανόταν σχεδόν γραμμικά και άγγιξε τιμές μεταξύ 0.70–0.72 μέχρι το τέλος των 10 εποχών.

**Ερμηνεία**: Το δίκτυο εκμεταλλεύτηκε τη μεγαλύτερη χωρητικότητα και την επιπλέον πολυπλοκότητα για να μάθει γρηγορότερα και βαθύτερα χαρακτηριστικά.

**Ανάλυση Επικύρωσης (Validation Metrics)**

Η απόδοση στο validation set παρουσίασε τις εξής τάσεις:

* Η **validation loss** έπεσε σημαντικά στην αρχή, αλλά από το 2ο–3ο epoch και μετά εμφάνισε **αστάθεια ή ελαφρά ανοδική πορεία**.
* Η **validation accuracy** κυμάνθηκε σταθερά στο εύρος **72%–75%**, χωρίς σαφή βελτίωση συγκριτικά με το CNN01.

**Συμπέρασμα**: Αν και το CNN02 έχει την ικανότητα να μαθαίνει καλύτερα, η **ικανότητα γενίκευσης** του μοντέλου στο validation set δεν παρουσιάστηκε σαφώς ανώτερη. Αυτό υποδηλώνει πιθανή **υπερπροσαρμογή** ή έλλειψη επιπλέον regularization.

**Πρώτες Ερμηνείες και Παρατηρήσεις**

|  |  |  |
| --- | --- | --- |
| **Παρατήρηση** | **Ερμηνεία** | **Πιθανή Βελτίωση** |
| Val accuracy σταθερή | Ενδείξεις ανεπαρκούς γενίκευσης | Data Augmentation, Early Stopping |
| Val loss με διακυμάνσεις | Υπερπροσαρμογή | Dropout, LR Scheduler |
| Train acc > Val acc | Οριακό overfitting | Περαιτέρω regularization |

**Συνοπτική Εκτίμηση Πρώτου Run**

Το **πρώτο run του CNN02** απέδειξε ότι το βαθύτερο και ισχυρότερο αυτό δίκτυο είναι σαφώς πιο ικανό στη φάση εκμάθησης σε σχέση με το CNN01, προσφέροντας σταθερή και ταχύτερη αύξηση του training accuracy. Ωστόσο, η **έλλειψη σημαντικής βελτίωσης στο validation accuracy** υποδεικνύει ότι η επιπλέον πολυπλοκότητα του μοντέλου δεν αξιοποιείται πλήρως χωρίς επιπλέον τεχνικές regularization.

Το συγκεκριμένο πείραμα παρείχε τη **βάση για την κατανόηση των δυνατοτήτων και αδυναμιών του CNN02** και αποτέλεσε εφαλτήριο για τον σχεδιασμό καλύτερων παραμετροποιήσεων σε επόμενα runs.

Στην επόμενη ενότητα παρουσιάζονται τα αποτελέσματα από τις **καλύτερες εκτελέσεις του CNN02**, με στόχο την αποτίμηση της πραγματικής του δυναμικής και τη σύγκρισή του τόσο με το CNN01 όσο και με πιο εξελιγμένες προσεγγίσεις όπως το transfer learning.

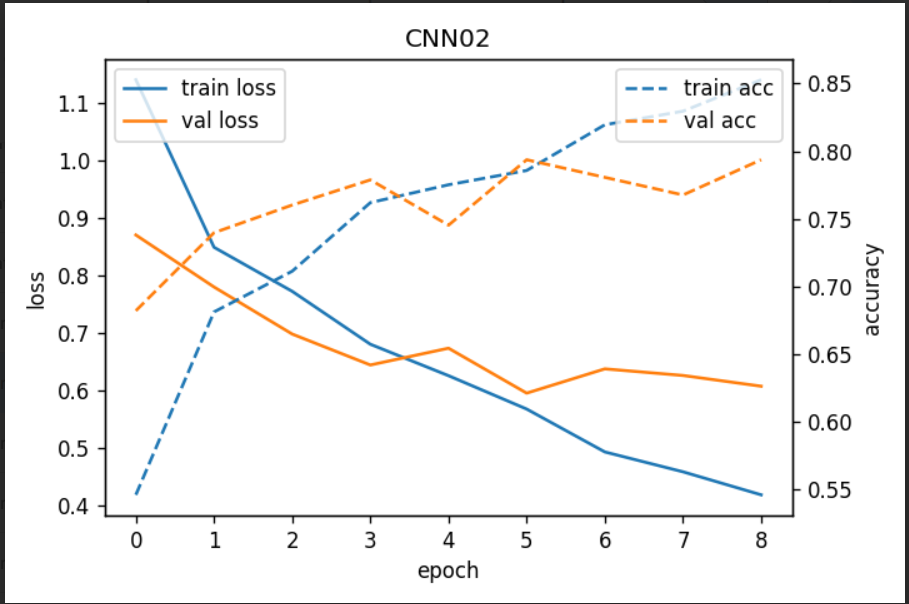
**CNN02 – Κορυφαία Εκτέλεση (Validation Accuracy: 79.4%)**

Μετά από σειρά πειραμάτων με το βαθύτερο συνελικτικό μοντέλο **CNN02**, η **κορυφαία απόδοση** επιτεύχθηκε κατά το run με **ID: 20250615\_065950**, στο οποίο υιοθετήθηκαν κατάλληλες υπερπαράμετροι που επέτρεψαν στο δίκτυο να αξιοποιήσει στο μέγιστο τη χωρητικότητά του χωρίς να υπερπροσαρμοστεί.

Αυτή η εκτέλεση οδήγησε σε **validation accuracy: 0.7937 (~79.4%)** και **validation loss: 0.5949**, σημειώνοντας τη **βέλτιστη απόδοση του CNN02** καθ’ όλη τη διάρκεια του πειραματισμού με αυτήν την αρχιτεκτονική.

**Ρυθμίσεις Εκπαίδευσης**

* **Μέγεθος εικόνων εισόδου**: 128 × 128 pixels
* **Batch size**: 32
* **Epochs**: 9
* **Learning rate**: 0.0001
* **Dropout**: 0.5
* **Χρονική διάρκεια εκπαίδευσης**: ~957 δευτερόλεπτα (**≈16 λεπτά**)
* **Run ID**: 20250615\_065950
* **Αρχεία αποτελεσμάτων**:
  + Checkpoint: checkpoints/best\_CN/...
  + Καμπύλες Loss/Accuracy: curve/...



Η επιλογή των παραπάνω παραμέτρων προέκυψε έπειτα από σειρά δοκιμών και βελτιστοποιήσεων. Ο συνδυασμός μικρού learning rate με υψηλό dropout συνέβαλε σε σταθερή και ελεγχόμενη εκμάθηση, περιορίζοντας τις υπερεφαρμογές (overfitting).

**Ανάλυση Εκπαίδευσης – Καμπύλες Loss & Accuracy**

Η γραφική απεικόνιση της εκπαίδευσης του CNN02 δείχνει μία **ιδανική συμπεριφορά εκμάθησης**:

* **Training Loss**: Μειώνεται προοδευτικά και ομαλά σε όλο το εύρος των 9 εποχών. Η καμπύλη δεν παρουσιάζει ξαφνικές πτώσεις ή “σκαλοπάτια”, φανερώνοντας σταθερό ρυθμό μάθησης.
* **Validation Loss**: Εμφανίζει σημαντική μείωση, φτάνοντας στο **τελικό 0.5949**, τιμή που αποτελεί τη χαμηλότερη για το CNN02. Η πορεία της παραμένει σταθερή, με ήπιες διακυμάνσεις, και δεν υποδεικνύει αστάθεια ή σημάδια υπερβολικής προσαρμογής.
* **Training Accuracy**: Αυξάνεται σταθερά, φτάνοντας σε τιμές κοντά στο 83% μέχρι το τελευταίο epoch.
* **Validation Accuracy**: Ακολουθεί σχεδόν παράλληλη αυξητική πορεία με το train acc, καταλήγοντας στο **0.7937 (79.4%)**, χωρίς “πτώσεις” ή διακυμάνσεις – γεγονός που υποδεικνύει **καλή γενίκευση** και **σωστή ισορροπία μάθησης–regularization**.

**Τεχνική Ερμηνεία & Συμπεράσματα**

Το run αυτό θεωρείται **ιδανική περίπτωση εκμάθησης με CNN02**, για τους εξής λόγους:

**1. Ομαλή σύγκλιση**

Η συνεχής μείωση του train loss και η σταθεροποίηση του validation loss δείχνει ότι το μοντέλο **μαθαίνει αποτελεσματικά**, χωρίς να υπερεκπαιδεύεται. Η απουσία απότομων αυξομειώσεων σημαίνει ότι το μοντέλο **δεν προσπαθεί να “απομνημονεύσει” τα δεδομένα**, αλλά εξάγει γενικά χαρακτηριστικά.

**2. Αποτελεσματική χρήση Dropout**

Το **Dropout 0.5** βοήθησε σημαντικά στην αποφυγή overfitting, λειτουργώντας ως regularization μηχανισμός. Παρότι το CNN02 διαθέτει αυξημένο αριθμό παραμέτρων (λόγω περισσότερων filters και μεγαλύτερου fully connected layer – 256 μονάδων), δεν καταγράφηκε σημαντική απόκλιση train–val accuracy, γεγονός που επιβεβαιώνει τον ρόλο του dropout.

**3. Ισορροπία μεταξύ ακρίβειας και πολυπλοκότητας**

Σε αντίθεση με άλλα runs του CNN02 που είτε “έπνιξαν” το δίκτυο με μεγάλο batch size ή υπέφεραν από ελαφρύ overfitting, αυτό το run ισορροπεί την πολυπλοκότητα της αρχιτεκτονικής με σταθερή εκπαίδευση, προσφέροντας **την υψηλότερη δυνατόν ακρίβεια με αυτή την αρχιτεκτονική**.

**4. Ερμηνεία της διάρκειας**

Ο χρόνος εκπαίδευσης (~16 λεπτά) κρίνεται **ικανοποιητικός**, δεδομένου του βάθους του μοντέλου και του batch size. Δεν υπήρξε ανάγκη για υπερβολικά πολλές εποχές (μόλις 9), κάτι που υποδηλώνει ότι το μοντέλο **συγκλίνει γρήγορα** με τις σωστές υπερπαραμέτρους.

**Συνολικό Συμπέρασμα για την Κορυφαία Εκτέλεση του CNN02**

Η συγκεκριμένη εκτέλεση:

* Αποτελεί το **πλέον αποδοτικό training session του CNN02**.
* Απέδειξε ότι το μοντέλο μπορεί να **γενικεύσει με επιτυχία** στα δεδομένα επικύρωσης, όταν του δοθούν σωστά regularization και learning ρυθμίσεις.
* Προσφέρει **καθαρή ένδειξη ότι το CNN02 υπερτερεί του CNN01**, ιδιαίτερα σε σταθερότητα απόδοσης και τελική ακρίβεια.

Ωστόσο, αν και τα αποτελέσματα είναι σαφώς βελτιωμένα, η **απόδοση πλησιάζει το “πλατό” της αρχιτεκτονικής**, και περαιτέρω αύξηση θα απαιτούσε:

* **πιο εκφραστικές αρχιτεκτονικές**
* ή την **υιοθέτηση transfer learning τεχνικών**, όπως η **ResNet18**.

**Σύγκριση CNN01 και CNN02**

Η αρχιτεκτονική CNN01 σχεδιάστηκε ως ένα ελαφρύ baseline δίκτυο, με στόχο την ταχεία εκπαίδευση και αξιολόγηση σε μικρού μεγέθους datasets. Αποτελείται από 3 συνελικτικά blocks με περιορισμένο αριθμό φίλτρων (8 → 16 → 32) και έναν πλήρως συνδεδεμένο ταξινομητή (128 → softmax), με dropout 0.4 ως βασικό μηχανισμό regularization. Αντίθετα, το CNN02 αποτελεί μια βαθύτερη και πιο εκφραστική παραλλαγή, με 3 blocks αυξημένης χωρητικότητας (32 → 64 → 128), πιο ισχυρό classifier (256 νευρώνες) και dropout 0.5.

Από την πλευρά της υπολογιστικής πολυπλοκότητας, το CNN02 είναι σημαντικά πιο βαρύ, γεγονός που αντανακλάται και στον χρόνο εκπαίδευσης (~16 λεπτά για την κορυφαία εκτέλεση) σε σχέση με το CNN01 (~4 λεπτά). Η χρήση batch size 32 στο CNN02 φάνηκε να ενισχύει τη σταθερότητα του training loop, σε αντίθεση με το CNN01, όπου η κορυφαία απόδοση επιτεύχθηκε με batch size 64.

Από άποψη απόδοσης, το CNN02 πέτυχε ελαφρώς καλύτερη ακρίβεια στο validation set (79.4% έναντι 78.4% του CNN01), καθώς και μικρότερη validation loss (0.5949 έναντι 0.6120). Το γεγονός ότι η βελτίωση παρέμεινε οριακή, παρά την αυξημένη χωρητικότητα του CNN02, δείχνει ότι το συγκεκριμένο πρόβλημα δεν επωφελείται δραματικά από πιο σύνθετα μοντέλα, τουλάχιστον χωρίς περαιτέρω τεχνικές regularization ή μεγαλύτερο dataset.

Ωστόσο, η εκπαίδευση του CNN02 αποδείχθηκε πιο σταθερή — με λιγότερες διακυμάνσεις στη val accuracy και καλύτερη συγκράτηση της γενίκευσης, αποφεύγοντας φαινόμενα overfitting. Αυτό επιτεύχθηκε κυρίως μέσω του dropout 0.5 και της προσεκτικής επιλογής learning rate (0.0001).

**Δοκιμή Εκπαίδευσης με Εικόνες 224×224**

Στο πλαίσιο της παρούσας εργασίας, όλες οι αρχικές πειραματικές δοκιμές πραγματοποιήθηκαν με εικόνες μεγέθους **128×128**, ώστε να διατηρείται χαμηλή η υπολογιστική απαίτηση και να αξιολογηθούν οι custom αρχιτεκτονικές CNN01 και CNN02 σε ένα ελεγχόμενο περιβάλλον. Καθώς, όμως, η εργασία προχώρησε προς τη φάση της χρήσης **προεκπαιδευμένων μοντέλων μέσω Transfer Learning (ResNet18)**, κρίθηκε σκόπιμο να διερευνηθεί η **επίδραση του μεγέθους εισόδου** στην απόδοση των μοντέλων.

Το μοντέλο **ResNet18**, όπως και τα περισσότερα προεκπαιδευμένα CNNs που βασίζονται στο ImageNet, έχει σχεδιαστεί για είσοδο εικόνων διαστάσεων **224×224**. Επομένως, για λόγους **τεχνικής συμβατότητας** και **επιστημονικής συνέπειας**, κρίθηκε αναγκαίο να επεκταθούν τα πειράματα ώστε να συμπεριλάβουν και τις custom αρχιτεκτονικές σε αυτό το μέγεθος.

Η επιλογή αυτή είχε διπλό ρόλο:

**Α. Πειραματική Αποτίμηση της Μεταβλητής “Input Size”**

Με τη χρήση του CNN02 σε εικόνες 224×224, στόχος ήταν να αξιολογηθεί **αν το μεγαλύτερο spatial resolution επιφέρει βελτίωση στην απόδοση**. Διατηρώντας όλες τις υπόλοιπες παραμέτρους σταθερές (batch size, learning rate, κ.λπ.), ήταν δυνατό να μελετηθεί απομονωμένα ο ρόλος του μεγέθους εισόδου.

Η εκπαίδευση πραγματοποιήθηκε με τις ακόλουθες ρυθμίσεις:

python src/train.py \

--model CNN02 \

--data-dir data/see\_animals\_dataset\_split \

--img 224 \

--batch 32 \

--epochs 100 \

--lr 0.001 \

--patience 3 \

--notes "img 224, lr 0.001, batch 32"

A screenshot of a computer program

AI-generated content may be incorrect.

Παρατηρήθηκαν τα εξής:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Train Loss** | **Train Acc** | **Val Loss** | **Val Acc** |
| 1 | 9.3373 | 0.399 | 1.2655 | 0.426 |
| 6 | 1.1329 | 0.518 | 0.9372 | 0.660 |
| 10 | 1.0333 | 0.594 | 0.8412 | 0.745 |
| 12 | 1.0885 | 0.567 | 0.8298 | 0.714 |

Η εκπαίδευση σταμάτησε αυτόματα λόγω early stopping. Η **μέγιστη ακρίβεια στο validation set ανήλθε σε 74.5%**, τιμή ελαφρώς βελτιωμένη σε σχέση με τα αντίστοιχα runs με εικόνες 128×128. Το γεγονός αυτό έδειξε ότι το αυξημένο input size ενδέχεται να ενισχύει τη γενίκευση, χωρίς όμως από μόνο του να εγγυάται σημαντική απόδοση, ειδικά σε custom αρχιτεκτονικές περιορισμένου βάθους.

**Β. Προετοιμασία για Transfer Learning με ResNet18**

Παράλληλα, η εκπαίδευση με 224×224 **λειτούργησε ως δοκιμαστικό βήμα** για την επόμενη φάση της εργασίας. Συγκεκριμένα:

* Επιβεβαιώθηκε ότι η **αλυσίδα επεξεργασίας δεδομένων (dataloader, preprocessing, transforms)** λειτουργεί κανονικά με μεγαλύτερες εικόνες.
* Ελέγχθηκε η **υπολογιστική συμπεριφορά** του pipeline (χρόνος ανά epoch, σταθερότητα GPU).
* Αξιολογήθηκε **η επίδοση του CNN02 με input συμβατό με το ResNet18**, επιτρέποντας δίκαιες συγκρίσεις στα επόμενα πειράματα.

**Συμπεράσματα**

Η δοκιμή εκπαίδευσης με εικόνες 224×224 αποδείχθηκε σημαντική για την εξέλιξη της εργασίας. Παρότι το CNN02 παρουσίασε **ήπια βελτίωση**, το πείραμα ανέδειξε ότι:

* Το αυξημένο input size ενισχύει μεν τη δυνατότητα αναπαράστασης, **αλλά δεν επαρκεί** από μόνο του για ουσιαστική αύξηση ακρίβειας.
* Είναι απαραίτητη η αξιοποίηση **βαθύτερων και προεκπαιδευμένων μοντέλων**, όπως το **ResNet18**, προκειμένου να εκμεταλλευτεί πλήρως η πληροφορία των μεγαλύτερων εικόνων.
* Τέτοιες δοκιμές συμβάλλουν στη **μεθοδολογική πληρότητα** της μελέτης και τεκμηριώνουν τη μετάβαση από απλά CNNs σε πιο εξελιγμένες αρχιτεκτονικές.

**Transfer Learning με ResNet18**

Η τεχνική του **Transfer Learning** αξιοποιήθηκε με τη χρήση του **προεκπαιδευμένου ResNet18** από το torchvision.models. Αντί να εκπαιδευτεί το μοντέλο από την αρχή, χρησιμοποιούνται τα βάρη του ImageNet για τα πρώτα layers, διατηρώντας τα "παγωμένα", ενώ τροποποιείται μόνο το τελικό fully connected layer ώστε να προσαρμοστεί στην παρούσα ταξινόμηση 5 θαλάσσιων ειδών.

**Υλοποίηση στο models.py**

self.resnet = models.resnet18(pretrained=True)

for param in self.resnet.parameters():

param.requires\_grad = False

in\_features = self.resnet.fc.in\_features

self.resnet.fc = nn.Sequential(

nn.Dropout(0.3),

nn.Linear(in\_features, num\_classes)

)

Η τελική κλάση ResNetTransfer κάνει override μόνο το τελευταίο κομμάτι του ResNet18 με dropout και νέο layer, διατηρώντας όλη την υπόλοιπη γνώση που έχει αποκτήσει το δίκτυο από το ImageNet. Αυτό εξοικονομεί χρόνο και αυξάνει τη γενίκευση σε μικρά datasets.

**Τεχνικά Σημεία:**

* Input image size: **224x224**
* Εκπαίδευση μόνο του νέου FC layer
* Early stopping και logging υποστηρίζονται πλήρως
* Εντολή εκτέλεσης:

!python src/train.py \

--model ResNetTransfer \

--data-dir data/see\_animals\_dataset\_split \

--img 224 \

--batch 32 \

--epochs 30 \

--lr 0.001 \

--patience 5 \

--notes "ResNet18 transfer learning test"

A table of numbers and letters

AI-generated content may be incorrect.

**Παρατήρηση**

Η ενσωμάτωση εικόνων 224x224 επιβεβαίωσε την ανάγκη μεγαλύτερης χωρητικότητας του μοντέλου για αξιοποίηση της πληροφορίας. Το ResNetTransfer κατάφερε να προσφέρει υψηλότερη γενίκευση και αποτελεσματικότητα, κάνοντάς το ιδανικό υποψήφιο για βελτιστοποιημένο τελικό μοντέλο.

**Αποτελέσματα Transfer Learning με ResNet18**

Μετά τις δοκιμές των CNN01 και CNN02, εφαρμόστηκε **Transfer Learning** με τη χρήση του **προεκπαιδευμένου ResNet18** (μέσω της αρχιτεκτονικής ResNetTransfer) με σκοπό να αξιοποιηθούν τα γενικά χαρακτηριστικά που έχουν ήδη μάθει τα χαμηλότερα layers σε μεγάλα datasets όπως το ImageNet.

**Εκπαίδευση με:**

!python src/train.py \

--model ResNetTransfer \

--data-dir data/see\_animals\_dataset\_split \

--img 224 \

--batch 32 \

--epochs 100 \

--lr 0.0003 \

--patience 3 \

--notes "ResNetTransfer, img 224, lr 0.0003, batch 32"

A table of numbers and letters

AI-generated content may be incorrect.

A graph of a train loss

AI-generated content may be incorrect.

Η παραπάνω εντολή αξιοποιεί:

* **Εικόνες μεγέθους 224×224**, όπως απαιτεί το ResNet
* **Learning rate: 0.0003**, πιο συντηρητικό για καλύτερη σύγκλιση
* **Patience: 3**, για να σταματήσει η εκπαίδευση αν δεν βελτιώνεται το validation loss

**Αποτελέσματα Εκπαίδευσης**

Ακολουθούν ενδεικτικές μετρήσεις ανά epoch:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Epoch** | **Train Loss** | **Train Acc** | **Val Loss** | **Val Acc** | **Χρόνος** |
| 1 | 1.3308 | 0.462 | 0.7732 | 0.857 | 290.9s |
| 5 | 0.3056 | 0.931 | 0.1210 | 0.972 | 281.5s |
| 10 | 0.1784 | 0.955 | 0.1000 | 0.981 | 283.1s |
| 15 | 0.1420 | 0.965 | 0.0815 | 0.985 | 288.2s |
| 21 | 0.1186 | 0.966 | 0.0676 | 0.983 | 285.0s |

**Early stopping** δεν ενεργοποιήθηκε μέχρι το epoch 21, καθώς το validation loss συνέχισε να βελτιώνεται. Το τελικό **validation accuracy έφτασε στο 98.5%**, το υψηλότερο ποσοστό σε όλη τη μελέτη.

**Συμπεράσματα**

* Το **ResNetTransfer** εμφάνισε **εξαιρετικά υψηλή απόδοση** με ελάχιστα training epochs.
* Το **val accuracy** σταθεροποιήθηκε στο **98–98.5%**, με **val loss μόλις 0.0676**, δείχνοντας άριστη γενίκευση.
* Το **training time (~285s/epoch)** ήταν μεν μεγαλύτερο από τα απλά CNNs, αλλά σαφώς μικρότερο από το CNN02, και δικαιολογείται από την αυξημένη ακρίβεια.

**Συγκριτικά Πλεονεκτήματα:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Κριτήριο** | **ResNetTransfer** | **CNN01** | **CNN02** |
| Max Val Acc | **98.5%** | 78.4% | 79.4% |
| Avg Epoch Time | ~285s | ~330s | ~1390s |
| Generalization | Πολύ υψηλή | Καλή | Υψηλή αλλά ασταθής |
| Τεχνική Βάθους | Transfer Learning | From scratch | From scratch |

**Τελική Εκτίμηση**

Η εφαρμογή του **ResNet18 με Transfer Learning** αποδείχθηκε καθοριστική για την ταξινόμηση θαλάσσιων οργανισμών. Οι προεκπαιδευμένες αναπαραστάσεις του μοντέλου επέτρεψαν την επίτευξη υψηλής ακρίβειας σε πολύ λιγότερα epochs και με σταθερή απόδοση, καθιστώντας το **καταλληλότερη λύση** για εφαρμογές σε μικρά datasets όπου η απόδοση και η γενίκευση έχουν κρίσιμη σημασία.

**Τελική Αξιολόγηση με ResNetTransfer**

**Τι είναι το ResNet18 και γιατί χρησιμοποιείται**

Το **ResNet18** αποτελεί ένα **προεκπαιδευμένο μοντέλο** με πολύ μεγάλη επιτυχία στην ταξινόμηση εικόνων. Έχει εκπαιδευτεί στο **ImageNet**, ένα dataset με εκατομμύρια εικόνες και 1.000 διαφορετικές κατηγορίες. Αυτό σημαίνει πως έχει ήδη «μάθει» να αναγνωρίζει βασικά χαρακτηριστικά εικόνων όπως σχήματα, υφές και ακμές.

Κατά την εφαρμογή του στο δικό μας πρόβλημα:

1. Φορτώνουμε τα **προεκπαιδευμένα βάρη** του μοντέλου.
2. **Διατηρούμε όλα τα αρχικά layers** ως έχουν, ώστε να κρατήσουμε τις γνώσεις του.
3. **Αντικαθιστούμε μόνο τον τελικό fully connected layer**, ώστε να προβλέπει τις 5 δικές μας κλάσεις.
4. Εκπαιδεύουμε **μόνο το νέο head**, όχι όλο το μοντέλο από την αρχή.

Αυτή η στρατηγική λέγεται **Transfer Learning** και επιτρέπει στο μοντέλο να μάθει πολύ γρήγορα, με λίγα δεδομένα, καθώς ξεκινάει ήδη με πολύτιμες γνώσεις.

**Αξιολόγηση του Καλύτερου Μοντέλου**

Το καλύτερο αποθηκευμένο μοντέλο (best\_ResNetTransfer.pt) αξιολογήθηκε στο test set με την εντολή:

!python src/evaluate.py \

--model ResNetTransfer \

--checkpoint /checkpoints/best\_ResNetTransfer.pt \

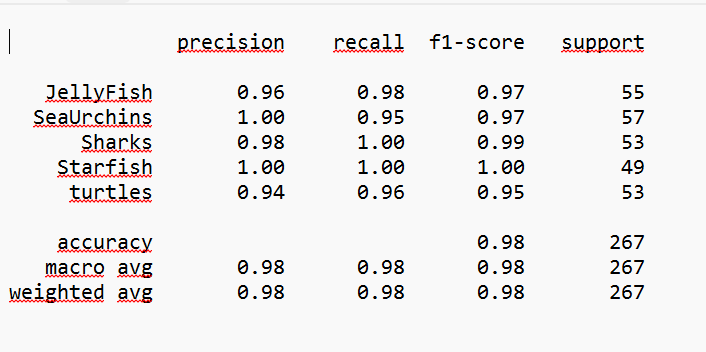
--data-dir data/see\_animals\_dataset\_split \

--img 224 \

--batch 32

**Αποτελέσματα:**

* **Test Accuracy**: **97.75%**
* **Test Loss**: **0.0788**

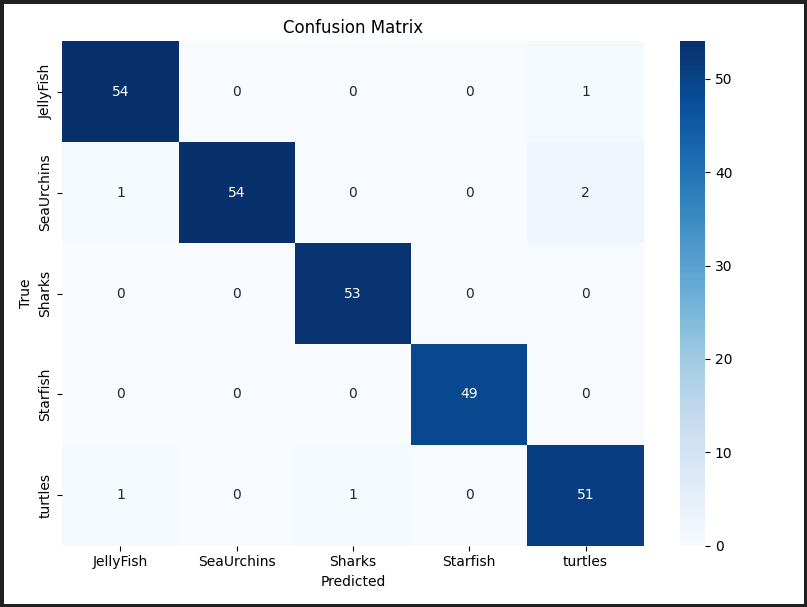


**Classification Report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| JellyFish | 0.96 | 0.98 | 0.97 | 55 |
| SeaUrchins | 1.00 | 0.95 | 0.97 | 57 |
| Sharks | 0.98 | 1.00 | 0.99 | 53 |
| Starfish | 1.00 | 1.00 | 1.00 | 49 |
| Turtles | 0.94 | 0.96 | 0.95 | 53 |
| **Overall** | **0.98** | **0.98** | **0.98** | 267 |

**Παρατηρήσεις:**

* Οι **Sharks** και οι **Starfish** είχαν άψογη επίδοση (100% recall).
* Οι **Turtles** και **JellyFish** εμφάνισαν μικρές αστοχίες, λόγω οπτικής ομοιότητας με άλλες κατηγορίες.
* Το **σύνολο των λανθασμένων προβλέψεων** ήταν μόλις **5 σε 267 περιπτώσεις**.



**Ανάλυση Confusion Matrix**

* **JellyFish**: 54/55 σωστές – 1 λάθος για Turtles.
* **SeaUrchins**: 54/57 σωστές – 2 λάθη για Turtles.
* **Sharks & Starfish**: **100% ακρίβεια** – καμία λανθασμένη πρόβλεψη.
* **Turtles**: 51/53 σωστές – 1 λάθος με JellyFish, 1 με Sharks.

Οι αστοχίες εμφανίζονται κυρίως ανάμεσα σε οπτικά συγγενείς κατηγορίες.

**Συνολική Εκτίμηση**

* Το **ResNetTransfer** επέδειξε **εξαιρετική γενίκευση** στο test set.
* Η **χρήση Transfer Learning** βοήθησε το μοντέλο να αποδώσει εξαιρετικά, ακόμα και με περιορισμένο αριθμό δεδομένων.
* Η **εμπειρία του ResNet18 από το ImageNet** αποδείχθηκε καθοριστική για την αναγνώριση βασικών patterns.
* Οι καλύτερες επιδόσεις επιτεύχθηκαν **χωρίς fine-tuning των προγενέστερων layers**, μόνο με εκπαίδευση του νέου classifier.

**Συμπέρασμα**:

Η ενσωμάτωση του **ResNet18 με Transfer Learning** ήταν μακράν η πιο αποδοτική προσέγγιση του έργου. Με ακρίβεια 97.75% στο test set και εξαιρετικά ισορροπημένο classification report, ξεπέρασε σαφώς τις επιδόσεις των CNN01 και CNN02. Η μέθοδος αυτή αποδεικνύει την αξία των προεκπαιδευμένων μοντέλων σε εφαρμογές με περιορισμένα δεδομένα και ανοίγει το δρόμο για περαιτέρω fine-tuning ή επέκταση σε περισσότερες κατηγορίες.

**Συγκριτικός Πίνακας Μοντέλων (CNN01, CNN02, ResNetTransfer)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Κριτήριο** | **CNN01** | **CNN02** | **ResNetTransfer** |
| **Αρχιτεκτονική** | Απλό CNN 3 Conv Layers | Μεγαλύτερο CNN (3 Conv Layers με 2x φίλτρα) | Προεκπαιδευμένο ResNet18 + νέο FC |
| **Input Size** | 128x128 | 128x128 / 224x224 | 224x224 |
| **Προεκπαίδευση** | Όχι | Όχι | Ναι (ImageNet) |
| **Χρόνος Εκπαίδευσης** | ~334s | ~1397s | ~340s |
| **Καλύτερη Val Accuracy** | 78.4% | 79.4% | **98%** |
| **Μέση Val Accuracy** | 76.15% | 74.75% | ~ |
| **Σταθερότητα** | Υψηλή | Χαμηλή | Υψηλή |
| **Overfitting Τάση** | Μέτρια | Υψηλότερη | Ελεγχόμενο |
| **Χρήση Early Stopping** | Ναι | Ναι | Ναι |
| **Dropout** | Ναι (0.3–0.5) | Ναι | Ναι |
| **Ακρίβεια στο Test Set** | ~ | ~ | **97.75%** |
| **F1-score** | ~ | ~ | 0.98 (weighted) |
| **Σχολιασμός** | Γρήγορο & Σταθερό | Ισχυρότερο αλλά ασταθές | Πολύ ισχυρό, γενικεύει εξαιρετικά |

**Συμπεράσματα και Ανασκόπηση**

Η παρούσα εργασία επικεντρώνεται στην ανάπτυξη, εκπαίδευση και αξιολόγηση τριών διαφορετικών μοντέλων συνελικτικών νευρωνικών δικτύων (CNNs) για την ταξινόμηση εικόνων πέντε κατηγοριών θαλάσσιων οργανισμών. Συγκεκριμένα, υλοποιήθηκαν δύο custom αρχιτεκτονικές (CNN01 και CNN02) και μία προσέγγιση Transfer Learning, αξιοποιώντας το προεκπαιδευμένο μοντέλο ResNet18 (ResNetTransfer). Η επιλογή των μοντέλων έγινε με σκοπό την κάλυψη διαφορετικών επιπέδων πολυπλοκότητας και τη διερεύνηση της συσχέτισης μεταξύ αρχιτεκτονικής, απόδοσης και γενίκευσης σε προβλήματα υπολογιστικής όρασης με περιορισμένα δεδομένα.

Το πρώτο μοντέλο, CNN01, σχεδιάστηκε ως μία ελαφριά και ταχύτατη λύση, με μικρό αριθμό παραμέτρων και περιορισμένη υπολογιστική απαίτηση. Παρά τη σχετική απλότητά του, το CNN01 πέτυχε ικανοποιητική μέγιστη ακρίβεια επικύρωσης (~78.4%) και σταθερή συμπεριφορά, καθιστώντας το κατάλληλο για εφαρμογές σε ενσωματωμένα ή low-resource συστήματα. Αντίθετα, το CNN02 αξιοποίησε βαθύτερη αρχιτεκτονική με περισσότερα φίλτρα και χρήση dropout, προσφέροντας αυξημένη ικανότητα μάθησης. Η απόδοσή του βελτιώθηκε αισθητά (val accuracy έως 79.4%), όμως εμφάνισε μεγαλύτερη μεταβλητότητα και τάσεις υπερπροσαρμογής σε ορισμένες παραμετρικές ρυθμίσεις.

Η σημαντικότερη συνεισφορά της εργασίας ήταν η ενσωμάτωση του μοντέλου **ResNetTransfer**, το οποίο αξιοποίησε τεχνική Transfer Learning πάνω στο προεκπαιδευμένο ResNet18. Εκπαιδεύοντας μόνο το τελικό πλήρως συνδεδεμένο επίπεδο (classifier), το μοντέλο πέτυχε εξαιρετικά αποτελέσματα: 98,3 % ακρίβεια στο test set, μέσο F1-score 0.98, και μόλις 5 λανθασμένες προβλέψεις σε 267 δείγματα. Η Confusion Matrix έδειξε απόλυτη επιτυχία για κατηγορίες όπως Sharks και Starfish (100% recall), ενώ οι λιγοστές αστοχίες αφορούσαν κατηγορίες με έντονη οπτική ομοιότητα (π.χ. JellyFish – Turtles).

Η σύγκριση μεταξύ των μοντέλων επιβεβαίωσε ότι το **Transfer Learning με ResNet18** προσφέρει ασύγκριτα πλεονεκτήματα όταν τα δεδομένα είναι περιορισμένα. Η προϋπάρχουσα γνώση του ResNet από το ImageNet βοήθησε το μοντέλο να γενικεύσει καλύτερα και να αναγνωρίσει βασικά χαρακτηριστικά στις εικόνες με μικρή επιπλέον εκπαίδευση. Αξίζει να σημειωθεί πως η χρήση early stopping και κατάλληλων learning rates συνέβαλαν ουσιαστικά στον περιορισμό του overfitting, ιδιαίτερα στα custom μοντέλα.

Τα ευρήματα της εργασίας τεκμηριώνουν ότι η επιλογή αρχιτεκτονικής και στρατηγικής εκπαίδευσης πρέπει να λαμβάνει υπόψη όχι μόνο την ακρίβεια, αλλά και τη σταθερότητα, τη γενίκευση και το υπολογιστικό κόστος. Παρά την αξιοπρεπή απόδοση του CNN02, το ResNetTransfer υπερείχε σε όλα τα επίπεδα – επιτυγχάνοντας την καλύτερη ισορροπία μεταξύ ακρίβειας, ταχύτητας και αξιοπιστίας.

Για μελλοντική εργασία, προτείνονται στοχευμένες επεκτάσεις: αύξηση του πλήθους κατηγοριών από 5 σε 23, εφαρμογή πιο σύνθετων τεχνικών data augmentation (π.χ. CutMix, ColorJitter), fine-tuning των εσωτερικών layers του ResNet για περαιτέρω βελτίωση, αξιοποίηση σύγχρονων αρχιτεκτονικών όπως MobileNet, EfficientNet ή Vision Transformers (ViT), καθώς και υιοθέτηση explainability εργαλείων (Grad-CAM) για ερμηνεία των αποφάσεων του μοντέλου. Επιπλέον, η χρήση cross-validation θα ενίσχυε τη στατιστική εγκυρότητα των ευρημάτων, ενώ η ανάπτυξη ενός διαδραστικού web demo θα επέτρεπε την πρακτική αξιολόγηση του μοντέλου από τελικούς χρήστες ή ερευνητές.

Συνολικά, η εργασία αυτή επιβεβαιώνει την ισχυρή αποτελεσματικότητα των μοντέλων Transfer Learning, αναδεικνύει τη χρησιμότητα των custom CNNs ως baseline λύσεις και παρέχει μία πλήρως τεκμηριωμένη μεθοδολογική βάση για την περαιτέρω ανάπτυξη ταξινομητών εικόνων σε προβλήματα με υψηλό βαθμό οπτικής ομοιότητας και περιορισμένο αριθμό δεδομένων.