Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο
ΕΠ34 Εφαρμογές Τεχνητής Νοημοσύνης

2η Άσκηση: Συνελικτικά Νευρωνικά Δίκτυα

Έκδοση 1.4

Διδάσκων: Χρήστος Δίου

1 Εισαγωγή

Στην άσκηση αυτή θα δούμε μία εφαρμογή των Συνελικτικών Νευρωνικών Δικτύων στην αυτόματη κατηγοριοποίηση ακτινογραφιών θώρακα για τον εντοπισμό ασθενών με COVID-19. Στόχος είναι να εξοικειωθούμε με τη δημιουργία και την εκπαίδευση διαφορετικών αρχιτεκτονικών ΣΝΔ.

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε συλλέχθηκε στα πλαίσια των εργασιών [1, 2]. Κάθε δείγμα των δεδομένων είναι μία ακτινογραφία θώρακα, ενώ οι ακτινογραφίες έχουν συλλεγεί από πολλαπλές πηγές. Κάθε ακτινογραφία ανήκει σε μία από τις παρακάτω κλάσεις:

- Φυσιολογική (Normal)
- Πνευμονικές αδιαφάνειες (Lung opacity)
- Ιογενής πνευμονία (Viral pneumonia)
- COVID-19

Παρακάτω φαίνονται τέσσερις φωτογραφίες, μία από κάθε κατηγορία

Παραδοτέα

Για την εργασία θα χρειαστεί να παραδώσετε:

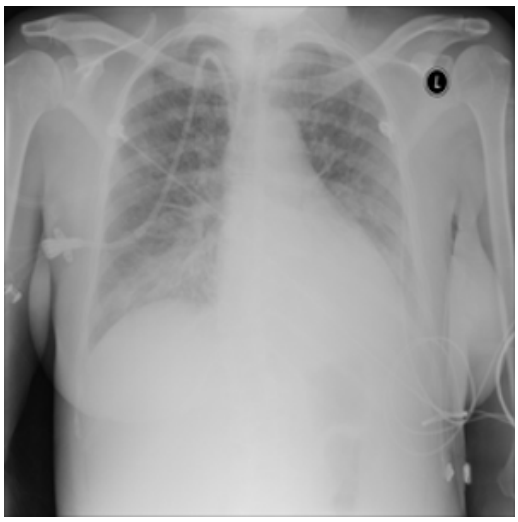
- Κώδικα σε γλώσσα python που να απαντά στα ερωτήματα της εργασίας.
- Μία συνοπτική αναφορά που να περιγράφει συνοπτικά τις επιλογές σας στην ανάπτυξη, τυχόν σχόλια που έχετε, καθώς και απαντήσεις στα ερωτήματα της εργασίας.

Μπορείτε επίσης να παραδώσετε την εργασία σας σε μορφή Jupyter Notebook. Για την υλοποίηση της εργασίας θα διευκολυνθείτε αν χρησιμοποιήσετε GPU (πχ μέσω του Google Colab ή του Kaggle). Όπως πάντα, δεν επιτρέπεται η αντιγραφή από άλλες πηγές.

Πρακτικές συμβουλές

- Αν κολλήσετε κάπου, ρωτήστε
- Ορισμένα από τα δίκτυα ενδεχομένως να χρειάζονται αρκετό χρόνο για να εκπαιδευτούν (τουλάχιστον σε CPU). Κατά τη φάση της ανάπτυξης χρησιμοποιήστε μικρά σύνολα δεδομένων ώστε να σιγουρευτείτε ότι όλα λειτουργούν πριν περάσετε στο πλήρες σύνολο
- Αξιοποιήστε την υποδομή GPU και TPU των colab ή/και Kaggle
- Αν για κάποιο λόγο δυσκολεύεστε πολύ να εκτελέσετε τον κώδικα για κάποιο ερώτημα, μειώστε τον αριθμό των εποχών

Φυσιολογική



Πνευμονικές αδιαφάνειες

Ιογενής πνευμονία

COVID-19

- Χρησιμοποιήστε διαφορετικά notebooks για διαφορετικά μοντέλα, ώστε να αποφύγετε ορισμένα σφάλματα για μη επαρκή μνήμη στα πιο μεγάλα μοντέλα

2 Σύνολο δεδομένων

Αρχικά κατεβάστε το σύνολο δεδομένων από τον παρακάτω σύνδεσμο:
https://drive.google.com/file/d/1UoicLVIFkMpEUhKnM9lpR1ME2xsfStI6/view?usp=sharing
Σε περίπτωση που θέλετε να εργαστείτε σε περιβάλλον Google Colab, μπορείτε να αντιγράψετε το αρχείο στο Google Drive και να το χρησιμοποιήσετε με τις εντολές, δίνοντας την έγκρισή σας για πρόσβαση στα δεδομένα από το notebook.

```
from google.colab import drive
drive.mount('/content/drive')

!cp '/content/drive/MyDrive/datasets/COVID-19_Radiography_Dataset.zip' .
!unzip -q -n COVID-19_Radiography_Dataset.zip
data_dir = '/content/COVID-19_Radiography_Dataset'
```

2.1 Κλάση `COVID19Dataset`

Αν αποσυμπιέσετε και μελετήσετε λίγο τα δεδομένα που σας δίνονται, θα διαπιστώσετε ότι έχουμε 4 φακέλους, όπου καθένας από αυτούς περιέχει τις εικόνες της αντίστοιχης κλάσης. Ο απλούστερος τρόπος να δημιουργήσουμε ένα αντικείμενο `Dataset` για τις εικόνες μας είναι μέσω της κλάσης `ImageFolder` από το module `torchvision.datasets`. Ωστόσο για την εργασία αυτή θα φτιάξετε μία δική σας κλάση, που θα την ονομάσετε `COVID19Dataset`, έτσι ώστε η `__getitem__()` να επιστρέφει ένα ζευγάρι `image, label` με την εικόνα και την ετικέτα.

Η κλάση σας πρέπει να έχει και μια μέθοδο `display_batch(self, indexes)` η οποία θα δέχεται μία λίστα από δείκτες `indexes`, και θα τα απεικονίζει σε ένα τετράγωνο από subfigures όπου κάθε subfigure έχει ως τίτλο την ετικέτα της αντίστοιχης εικόνας.

Ζητούμενο 1: Υλοποιήστε την κλάση `COVID19Dataset` και δημιουργήστε ένα αντικείμενο της με τις εικόνες που κατεβάσατε. Δημιουργήστε επίσης μία λίστα με 25 τυχαίων indexes εικόνων του συνόλου δεδομένων και παρουσιάστε τα χρησιμοποιώντας τη μέθοδο `display_batch`. Διατρέξτε το σύνολο δεδομένων και υπολογίστε το πλήθος των εικόνων της κάθε κλάσης και απεικονίστε το σε ένα ραβδόγραμμα (bar chart). Σχολιάστε το αποτέλεσμα.

3 Συναρτήσεις εκπαίδευσης και δοκιμής

Αρχικά υλοποιήστε μια συνάρτηση `confusion_matrix` η οποία δέχεται δύο tensors `y` και `y_pred` με τις πραγματικές και τις εκτιμώμενες κλάσεις του κάθε δείγματος και επιστρέφει τον πίνακα σύγχυσης.

Έπειτα υλοποιήστε συναρτήσεις `train_one_epoch` και `test`. Η πρώτη πρέπει να δέχεται ένα μοντέλο, έναν `DataLoader`, έναν `optimizer`, μία συνάρτηση απώλειας, και ένα `device` και εκτελεί μια εποχή εκπαίδευσης του μοντέλου.

Η δεύτερη πρέπει να δέχεται ένα μοντέλο, έναν `DataLoader`, μία συνάρτηση απώλειας, και ένα `device` και υπολογίζει και επιστρέφει την τιμή της συνάρτησης απώλειας, την ευστοχία και τον πίνακα σύγχυσης του μοντέλου στο σύνολο δεδομένων που αντιστοιχεί ο `DataLoader`.

4 Απλό συνελικτικό δίκτυο

Υλοποιήστε ένα μοντέλο `CNN1` το οποίο αποτελείται από

1. Ένα συνελικτικό επίπεδο με 8 φίλτρα 3×3 και συνάρτηση ενεργοποίησης ReLU
2. Ένα επίπεδο συγκέντρωσης με βήμα 2
3. Ένα συνελικτικό επίπεδο με 16 φίλτρα 3×3 και συνάρτηση ενεργοποίησης ReLU
4. Ένα επίπεδο συγκέντρωσης με βήμα 2
5. Ένα επίπεδο μετατροπής σε 1 διάσταση
6. Ένα πλήρως συνδεδεμένο επίπεδο με 32 νευρώνες και συνάρτηση ενεργοποίησης ReLU
7. Ένα επίπεδο με 4 εξόδους

Μη χρησιμοποιήσετε padding στις συνελίξεις.

Χρησιμοποιήστε την κλάση `COVID19Dataset` που υλοποιήσατε προηγουμένως ώστε να φορτώσετε το σύνολο δεδομένων, εφαρμόζοντας τους κατάλληλους μετασχηματισμούς. Έπειτα χρησιμοποιήστε τη συνάρτηση `random_split` του `torch.utils.data` ώστε να χωρίσετε το σύνολο δεδομένων σε 60%, 20%, 20% για τα σύνολα εκπαίδευσης, επικύρωσης και δοκιμής, αντίστοιχα, χρησιμοποιώντας το 42 ως random seed. Αυτό γίνεται ως εξής:

```
generator = torch.Generator().manual_seed(42)
train_ds, val_ds, test_ds = \
     random_split(data, [0.6, 0.2, 0.2], generator=generator)
```

Ζητούμενο 2: Εκπαιδεύστε το μοντέλο σας χρησιμοποιώντας τις συναρτήσεις `train_one_epoch` και `test` που υλοποιήσατε προηγουμένως και

- Τον αλγόριθμο βελτιστοποίησης Adam με ρυθμό εκμάθησης 10^{-3}, $\beta_1 = 0.9$ (ρυθμός ενημέρωσης πρώτης ροπής) $\beta_2 = 0.99$ (ρυθμός ενημέρωσης δεύτερης ροπής).
- Τη διεντροπία ως συνάρτηση απώλειας
- Τα σύνολα `train_ds` και `val_ds` για εκπαίδευση και επικύρωση αντίστοιχα
- `batch_size` 64
- 20 εποχές μέγιστη διάρκεια εκπαίδευσης
- Πρόωρο τερματισμό της εκπαίδευσης (Early Stopping) αν υπάρχει απόκλιση πάνω από 0.5 της απώλειας στο σύνολο εκπαίδευσης και το σύνολο επικύρωσης για 5 διαδοχικές εποχές

Ποια είναι η ευστοχία (accuracy) του μοντέλου σας στο σύνολο εκπαίδευσης, στο σύνολο επικύρωσης και στο σύνολο δοκιμής;

Δεδομένης της κατανομής των κατηγοριών στο σύνολο δεδομένων, θα χρειαστεί να εξετάσετε λίγο πιο λεπτομερώς τι συμβαίνει στο μοντέλο σας.

Με βάση τον πίνακα σύγχυσης, σχολιάστε την επίδοση του μοντέλου σε κάθε κλάση, σε σχέση και με την κατανομή των κλάσεων στο σύνολο δεδομένων. Ποια θεωρείτε ότι είναι η σημασία των αποτελεσμάτων σας στην κλινική πράξη;

5 Συνελικτικό δίκτυο μεγαλύτερου βάθους

Όπως και στην προηγούμενη ενότητα, δημιουργήστε τώρα ένα βαθύτερο ΣΝΔ υλοποιώντας συνάρτηση

`cnn2(num_classes)`

η οποία επιστρέφει ένα μοντέλο ΣΝΔ το οποίο αποτελείται από

1. Δύο διαδοχικά συνελικτικά επίπεδα με 32 φίλτρα 3×3, με έξοδο ίση με την είσοδο και συνάρτηση ενεργοποίησης ReLU

2. Ένα επίπεδο συγκέντρωσης Max Pooling με βήμα 4
3. Δύο διαδοχικά συνελικτικά επίπεδο με 64 φίλτρα 3×3, με έξοδο ίση με την είσοδο και συνάρτηση ενεργοποίησης ReLU
4. Ένα επίπεδο συγκέντρωσης Max Pooling με βήμα 2
5. Δύο διαδοχικά συνελικτικά επίπεδο με 128 φίλτρα 3×3, με έξοδο ίση με την είσοδο και συνάρτηση ενεργοποίησης ReLU
6. Ένα επίπεδο συγκέντρωσης Max Pooling με βήμα 2
7. Τρία διαδοχικά συνελικτικά επίπεδο με 256 φίλτρα 3×3, με έξοδο ίση με την είσοδο και συνάρτηση ενεργοποίησης ReLU
8. Ένα επίπεδο συγκέντρωσης Max Pooling με βήμα 2
9. Ένα συνελικτικό επίπεδο με 512 φίλτρα 3×3, με έξοδο ίση με την είσοδο και συνάρτηση ενεργοποίησης ReLU
10. Ένα επίπεδο συγκέντρωσης Max Pooling με βήμα 2
11. Ένα επίπεδο μετατροπής σε 1 διάσταση (`Flatten`)
12. Ένα πλήρως συνδεδεμένο επίπεδο με 1024 νευρώνες και συνάρτηση ενεργοποίησης ReLU
13. Ένα επίπεδο με `num_classes` εξόδους

Εκπαιδεύστε και αξιολογήστε το μοντέλο όπως και προηγουμένως.

Ζητούμενο 3: Συγκρίνετε αυτό το μοντέλο με το προηγούμενο. Παρατηρείτε διαφορά στην επίδοση; Αν ναι, που θεωρείτε ότι οφείλεται;

6 Με χρήση προεκπαιδευμένου δικτύου

Κάντε ότι κάνατε στις προηγούμενες περιπτώσεις, αλλά χρησιμοποιήστε ένα προεκπαιδευμένο δίκτυο ResNet50, το οποίο μπορείτε να χρησιμοποιήσετε έτοιμο από το Pytorch (module `torchvision.models`)

Ζητούμενο 4: Εκπαιδεύστε το δίκτυο με μέγεθος batch 64 για 5 εποχές με ρυθμό εκμάθησης 10^{-4} και αξιολογήστε την επίδοσή του (οι υπόλοιπες παράμετροι όπως προηγουμένως). Σχολιάστε τα αποτελέσματα. Πειραματιστείτε επίσης χρησιμοποιώντας το προεκπαιδευμένο μοντέλο ως μηχανισμό εξαγωγής χαρακτηριστικών, με εκπαίδευση μόνο του τελικού επιπέδου ταξινόμησης.

7 Προαιρετικά (Bonus): Συνελικτικό δίκτυο με παραλειπόμενες συνδέσεις

Ένας τρόπος να βελτιώσουμε την εκπαίδευση των νευρωνικών δικτύων με μεγάλο βάθος (αποφεύγοντας το πρόβλημα τις εξαφάνισης παραγώγων) είναι και η χρήση συνδέσεων που παραλείπουν ορισμένα επίπεδα (skip connections). Στην ενότητα αυτή θα υλοποιήσουμε ένα ΣΝΔ που αξιοποιεί skip connections για να βελτιώσει την επίδοση του δικτύου.

7.1 Μονάδα με παραλειπόμενες συνδέσεις

Θεωρήστε μία μονάδα με παραλειπόμενες συνδέσεις που αποτελείται από

1. Ένα συνελικτικό επίπεδο με n φίλτρα 3×3, με padding 1, με `stride` 1 ή 2 (δίνεται ως είσοδος)
2. Ένα επίπεδο ομαδικής κανονικοποίησης (batch normalization) κατά τον άξονα των φίλτρων
3. Ένα επίπεδο ενεργοποίησης ReLU
4. Ένα συνελικτικό επίπεδο με n φίλτρα 3×3, με έξοδο ίση με την είσοδο
5. Ένα επίπεδο ομαδικής κανονικοποίησης (batch normalization) κατά τον άξονα των φίλτρων
6. Την πρόσθεση του τελευταίου επιπέδου με την είσοδο
7. Ένα επίπεδο ενεργοποίησης ReLU

Σε περίπτωση που `stride = 2`, τότε επιπλέον η κλάση υλοποιεί

1. Ένα συνελικτικό επίπεδο με n φίλτρα 1×1, με είσοδο την είσοδο της μονάδας, βήμα 2, που έχει ως στόχο την αλλαγή διάστασης της εξόδου
2. ένα βήμα το οποίο προσθέτει αυτή την έξοδο με την έξοδο από την επεξεργασία της παραπάνω λίστας
3. Ένα επίπεδο ενεργοποίησης ReLU

Υλοποιήστε κλάση

```
BasicBlock(n_in, n_filters, stride=1)
```

με παραμέτρους

- `n_in`: Αριθμός των καναλιών εισόδου
- `n_filters`: Αριθμός φίλτρων
- `stride`: Παράμετρος που προσδιορίζει το βήμα του πρώτου συνελικτικού επιπέδου (και βάσει του οποίου προστίθενται οι 1×1 συνελίξεις ή όχι)

η οποία υλοποιεί το παραπάνω μοντέλο.

Ζητούμενο bonus: Πειραματιστείτε με τη χρήση του `BasicBlock` και διάφορες τιμές υπερπαραμέτρων για την ανάπτυξη ενός αποτελεσματικού μοντέλου ταξινόμησης ακτινογραφιών θώρακα για τις δεδομένες παθήσεις. Μπορείτε να εμπνευστείτε από τον τρόπο που οι αρχιτεκτονικές ResNet χρησιμοποιούν παρόμοια μοντέλα για κατηγοριοποίηση εικόνων.

Αναφορές

[1] Muhammad EH Chowdhury, Tawsifur Rahman, Amith Khandakar, Rashid Mazhar, Muhammad Abdul Kadir, Zaid Bin Mahbub, Khandakar Reajul Islam, Muhammad Salman Khan, Atif Iqbal, Nasser Al Emadi, et al. Can ai help in screening viral and covid-19 pneumonia? *IEEE Access*, 8:132665–132676, 2020.

[2] Tawsifur Rahman, Amith Khandakar, Yazan Qiblawey, Anas Tahir, Serkan Kiranyaz, Saad Bin Abul Kashem, Mohammad Tariqul Islam, Somaya Al Maadeed, Susu M Zughaier, Muhammad Salman Khan, et al. Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images. *Computers in biology and medicine*, 132:104319, 2021.