Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο
ΕΠ34 Μηχανική Μάθηση και Εφαρμογές

2η Άσκηση: Συνελικτικα Νευρωνικά Δίκτυα

Έκδοση 1.0

Διδάσκων: Χρήστος Δίου

1 Εισαγωγή

Στη δεύτερη άσκηση θα χρησιμοποιήσουμε το `pytorch` για την κατηγοριοποίηση εικόνων με συνελικτικά νευρωνικά δίκτυα (ΣΝΔ). Θα υλοποιήσουμε και θα εκπαιδεύσουμε δίκτυα από την αρχή, ενώ θα χρησιμοποιήσουμε και προεκπαιδευμένα δίκτυα ως εξαγωγείς χαρακτηριστικών, με την τεχνική της μεταφοράς μάθησης (transfer learning). Η εφαρμογή στην οποία θα εστιάσουμε είναι η αυτόματη κατηγοριοποίηση εικόνων δερματοσκόπησης. Η δερματοσκόπηση είναι η χρήση ψηφιακών εικόνων για τη διάγωνση βλαβών του δέρματος, συμπεριλαμβανομένου και του δερματικού καρκίνου. Η χρήση αυτοματοποιημένων μεθόδων ανάλυσης τέτοιων εικόνων μπορεί να προσφέρει πολλαπλά ωφέλη στο μέλλον, που περιλαμβάνουν (α) τον εύκολο έλεγχο και εντοπισμό βλαβών που σχετίζονται με υψηλό ρίσκο για την εμφάνιση καρκίνου του δέρματος, (β) την τηλεδιάγνωση δερματολογικών παθήσεων και (γ) την υποστήριξη των δερματολόγων ώστε να επιτευχθεί ακριβέστερη διάγνωση.

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε παρουσιάστηκε στην ερευνητική εργασία [1] και προέρχεται από περίπου 10.000 εικόνες δερματοσκόπησης από δύο οργανισμούς στην Αυστρία και την Αυστραλία. Οι εικόνες έρχονται με ετικέτες που αφορούν 7 παθήσεις:

- MEL: Μελάνωμα (Melanoma)
- NV: Σπίλος (ελιά, Nevi)
- BCC: Βασικοκυτταρικό Καρκίνωμα (Basal cell carcinoma)
- AKIEC: Ακτινική κεράτωση (Actinic keratosis)
- BKL: Καλοήθης κεράτωση (Benign keratosis)
- DF: Δερματοΐνωμα (Dermatofibroma)
- VASC: Αγγειακή βλάβη (Vascular lesion)

Σε εκτεταμένη αξιολόγηση που έγινε στα πλαίσια μελέτης των Tschandl et al. [2], μεμονωμένοι δερματολόγοι είχαν κατά μέσο όρο ευστοχία 64% στην αξιολόγηση των παραπάνω ασθενειών σε 600 εικόνες του συνόλου δεδομένων. Στην ίδια μελέτη ο συνδυασμός μοντέλων μηχανικής μάθησης με τους ειδικούς, πέτυχε τη μεγαλύτερη ευστοχία (83.9%), δείχνοντας την αξία της χρήσης τέτοιων μεθόδων στην ιατρική πράξη.

Παραδοτέα - σημειώσεις

Παραδοτέα θα είναι ο κώδικας σε γλώσσα python, καθώς και αναφορά. Ο κώδικας μπορεί να αποτελείται από πολλαπλά αρχεία. Κώδικας σε python notebooks είναι επίσης αποδεκτός.

Την προετοιμασία του κώδικα για τη δημιουργία και την εκπαίδευση των μοντέλων μπορείτε να την κάνετε σε κάποιο άλλο σύνολο δεδομένων που δίνει έτοιμο το pytorch, όπως το CIFAR-10. Σε περίπτωση που δυσκολευτείτε να φορτώσετε τα δεδομένα, παρουσιάστε τα αποτελέσματά σας στο CIFAR-10 (ακόμα και

σ' αυτή την περίπτωση, θα λάβετε το 50% του βαθμού, εφόσον είναι όλα σωστά). Επομένως μπορείτε αρχικά να αναπτύξετε την εργασία σας στο CIFAR-10, και έπειτα να περάσετε στη χρήση του συνόλου δεδομένων της εργασίας. Όπως και στην προηγούμενη άσκηση, δεν επιτρέπεται η απευθείας αντιγραφή από άλλες πηγές.

Πρακτικές συμβουλές

- Αν κολλήσετε κάπου, ρωτήστε
- Ορισμένα από τα δίκτυα χρειάζονται κάποιο χρόνο για να εκπαιδευτούν (τουλάχιστον σε CPU). Κατά τη φάση της ανάπτυξης μπορείτε να χρησιμοποιήσετε μικρά σύνολα δεδομένων ώστε να σιγουρευτείτε ότι όλα λειτουργούν πριν περάσετε στο πλήρες σύνολο
- Τις πρώτες δοκιμές θα πρότεινα να τις κάνετε σε CPU στον προσωπικό σας υπολογιστή. Συνήθως κάνει ευκολότερη την ανάπτυξη.
- Για τις τελικές σας δοκιμές, μπορείτε να αξιοποιήσετε 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

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/dermoscopy_classification.tar.gz' .
!tar -xvzf dermoscopy_classification.tar.gz'
data_dir = '/content/dermoscopy_classification'
```

2 Φόρτωση δεδομένων (Datasets / DataLoader) - 4 μονάδες

Μπορείτε να κατεβάσετε τα δεδομένα από τον ακόλουθο σύνδεσμο με τον ιδρυματικό σας λογαριασμό: https://drive.google.com/file/d/1DaFBO4K9Z7fE9k5C_LrdyWJSs76OsvYl/view?usp=sharing

Υλοποιήστε μία κλάση `MLProject2Dataset` η οποία θα είναι υποκλάση της κλάσης `Dataset` του pytorch.

2.1 Μέθοδος `__init__`

Η μέθοδος `__init__` της κλάσης θα δέχεται μία διαδρομή του καταλόγου που βρίσκονται τα δεδομένα, το όνομα ενός αρχείου μεταδεδομένων (default `'metadata.csv'`) καθώς και έναν μετασχηματισμό (`torchvision.transform`) με προεπιλεγμένη τιμή `None`, ο οποίος θα εφαρμόζεται στα δεδομένα.

```
def __init__(self, data_dir, metadata_fname='metadata.csv', transform=None)
```

Η μέθοδος αρχικά θα εκχωρεί τις μεταβλητές `self.data_dir` και `self.transform` της κλάσης. Έπειτα, θα δημιουργεί ένα `pandas.DataFrame` με στήλες `'image_id'` και `'path'` με τη διαδρομή κάθε εικόνας, όπου `'image_id'` είναι το όνομα αρχείου χωρίς επέκταση, και `'path'` είναι η πλήρης διαδρομή κάθε εικόνας. Ένας τρόπος να το πετύχετε αυτό είναι με χρήση του `glob` module. Το `'image_id'` αντιστοιχεί στη στήλη `'image_id'` του αρχείου `'metadata.csv'`.

Η μέθοδος επιπλέον θα φορτώνει τον πίνακα metadata (πχ μέσω της `read_csv()` του pandas) και θα μετατρέπει τις συμβολοσειρές της στήλης `'dx'` σε ακέραιους (πχ μέσω της `pandas.Categorical()`).

Έπειτα, θα δημιουργεί ένα `pandas.DataFrame` το οποίο για κάθε εικόνα έχει τη διαδρομή και το αντίστοιχο label (αριθμητική τιμή της στήλης `'dx'`). Μπορείτε να το πετύχετε αυτό μέσω `pandas.merge()` στα δύο προηγούμενα dataframe που δημιουργήσατε.

2.2 Μέθοδοι `__len__()` και `__getitem__()`

Για την υλοποίηση ενός `torch.utils.data.Dataset`, πρέπει να ορίσουμε τις μεθόδους `__len__()` και `__getitem__()`. Η πρώτη θα πρέπει απλά να επιστρέφει το μέγεθος του dataset. Η δεύτερη δέχεται ένα index `idx` και επιστρέφει την αντίστοιχη εικόνα και το label της. Αυτό είναι εύκολο αν έχετε δημιουργήσει το σχετικό `pandas.DataFrame` στην `__init__()` παραπάνω.

Για την ανάγνωση της εικόνας μπορείτε να χρησιμοποιήσετε τη μέθοδο `torchvision.io.read_image` του pytorch. Η μέθοδος επιστρέφει `torch.Tensor` με ακέραιες τιμές στο 0-255. Θα χρειαστεί να μετατρέψετε τις τιμές σε πραγματικούς αριθμούς (πχ με την `to()`) και να διαιρέσετε με 255 ώστε οι τιμές να μεταφερθούν στο [0-1]. Εναλλακτικά, μπορείτε να χρησιμοποιήσετε τη βιβλιοθήκη `PIL` της python και να δώσετε τον μετασχηματισμό `ToTensor()` στον `DataLoader` αργότερα.

Η `__getitem__()` πρέπει να επιστρέφει ένα ζεύγος `(X, y)` που αντιστοιχούν στην εικόνα και την ετικέτα της, αντίστοιχα.

2.3 Train / validation / test split και Data loaders

Χρησιμοποιήστε τη μέθοδο `torch.utils.data.random_split`, με έναν generator με σταθερό seed = 42 ώστε να δημιουργήσετε τα σύνολα εκπαίδευσης (training), επικύρωσης (validation) και δοκιμής (test) με ποσοστά 60%, 10% και 30% αντίστοιχα.

Σημείωση: Εδώ έχουμε την ιδιαιτερότητα ότι ορισμένες περιπτώσεις (πεδίο `lesion_id`) εμφανίζονται με πολλαπλές εικόνες στο σύνολο δεδομένων. Κανονικά θέλουμε οι εικόνες κάθε δείγματος είτε να είναι όλες στο σύνολο εκπαίδευσης, είτε στο σύνολο δοκιμής, αλλά όχι να μοιραστούν μεταξύ των δύο. Εδώ για λόγους απλούστευσης αγνοούμε αυτό το ζήτημα. Προαιρετικά, αν θέλετε μπορείτε να φροντίσετε ώστε εικόνες που αντιστοιχούν στο ίδιο `lesion_id` να είναι όλες στο ίδιο υποσύνολο δεδομένων.

2.4 Μετασχηματισμοί

Μπορείτε να χρησιμοποιήσετε μετασχηματισμούς του συνόλου δεδομένων για την αλλαγή μεγέθους των εικόνων (`torchvision.transforms.Resize()`) σε $m \times n$ (τα m και n ορίζονται σε κάθε ερώτημα). Έπειτα εφαρμόστε τυποποίηση των εικόνων με (`torchvision.transforms.Normalize()`) με μέση τιμή 0.5 και τυπική απόκλιση 0.5 (ορίσματα `(0.5, 0.5, 0.5), (0.5, 0.5, 0.5)`, καθώς χρειάζεται μία τιμή για κάθε κανάλι εισόδου).

Αν χρησιμοποιήσατε την PIL για τη φόρτωση των εικόνων, θα χρειαστεί να χρησιμοποιήσετε και τον μετασχηματισμό `ToTensor()` ώστε να μετατρέψετε την εικόνα σε `Tensor`.

3 Μέθοδοι εκπαίδευσης και δοκιμής

3.1 `train_net`

Υλοποιήστε μέθοδο `train_net`:

```
def train_net(model: nn.Module, trainloader: DataLoader, valloader: DataLoader = None,
              epochs: int = 10, optimizer: optim = None, loss: nn.modules.loss = None,
              device: str = 'cpu', print_period: int = 10) -> None:
```

Η μέθοδος θα πρέπει να εκπαιδεύει το μοντέλο `model`, όπως στα παραδείγματα που είδαμε στο μάθημα. Μετά από κάθε `print_period` επαναλήψεις η μέθοδος θα αξιολογεί το μοντέλο στο σύνολο επικύρωσης και θα τυπώνει, για την τρεχουσα περίοδο:

- το μέσο loss στο σύνολο εκπαίδευσης (μόνο για τα δεδομένα της περιόδου)
- το accuracy στο σύνολο εκπαίδευσης (μόνο για τα δεδομένα της περιόδου)
- το μέσο loss στο σύνολο επικύρωσης
- το accuracy στο σύνολο επικύρωσης

3.2 `test_net`

Υλοποιήστε μέθοδο `test_net`:

```
def test_net(model: nn.Module, testloader: DataLoader,
             loss: nn.modules.loss = None, device: str = 'cpu') -> None:
```

Η μέθοδος θα πρέπει να εφαρμόζει το μοντέλο `model` στα δεδομένα του `testloader` και να υπολογίζει και να επιστρέφει το μέσο loss και το accuracy.

4 Απλό ΣΝΔ (2 μονάδες)

Για το πρώτο ερώτημα μετατρέπουμε τις εικόνες σε μικρό μέγεθος, οπότε θέτουμε στη `Resize` $m = 50$ και $n = 62$ Το πρώτο μοντέλο είναι ένα ΣΝΔ με

- ένα συνελικτικό επίπεδο με 32 πυρήνες 3×3, ακολουθούμενο από ένα επίπεδο ReLU και Max Pooling 2×2
- ένα συνελικτικό επίπεδο με 64 πυρήνες 3×3, ακολουθούμενο από ένα επίπεδο ReLU και Max Pooling 2×2
- ένα συνελικτικό επίπεδο με 64 πυρήνες 3×3, ακολουθούμενο από ένα επίπεδο ReLU και Max Pooling 2×2
- μετατροπή σε διάνυσμα (με `nn.Flatten()`) και ένα γραμμικό επίπεδο για την έξοδο

Εκπαιδεύστε το μοντέλο με απώλεια διεντροπίας και στοχαστική κατάβαση κλίσης με ρυθμό εκμάθησης 0.1 για 20 εποχές. Κρατήστε τις ενδιάμεσες τιμές της απώλειας και της ευστοχίας και απεικονίστε τις σε ένα γράφημα. Στο τέλος, αξιολογήστε το μοντέλο σας στο σύνολο δοκιμής.

Τι επίδοση επιτυγχάνετε; Παρουσιάστε τον πίνακα σύγχυσης του μοντέλου στο σύνολο δοκιμής. Σχολιάστε επίσης την πορεία της εκπαίδευσης του μοντέλου.

Προαιρετικά: Μπορείτε να παρακολουθείτε και να εκτυπώνετε την πορεία της εκπαίδευσης του μοντέλου μέσω του `tensorboard`, με τη μέθοδο που παρουσιάζεται εδώ:
https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html

5 Σύνθετο ΣΝΔ (2 μονάδες)

Στο δεύτερο ερώτημα θα χρησιμοποιήσουμε ένα πιο σύνθετο μοντέλο, αξιοποιώντας επίσης πιο μεγάλη εικόνα εισόδου. Συγκεκριμένα το νέο μοντέλο θα έχει πέντε επίπεδα, όπου κάθε επίπεδο έχει ένα συνελικτικό επίπεδο με πυρήνες 3×3, ακολουθούμενο από ReLU συνάρτηση ενεργοποίησης, ένα επίπεδο Batch Normalization (που υλοποιείται από την κλάση `nn.BatchNorm2d`) και ένα επίπεδο Max Pooling 2×2. Το πλήθος των εξόδων θα είναι

1. 32
2. 64
3. 128

4. 256
5. 512

Μετά το τελευταίο επίπεδο εφαρμόζεται global average pooling, που στο pytorch υλοποιείται με τη συνάρτηση `torch.nn.functional.adaptive_avg_pool2d()` όπου θέτουμε τη διάσταση της εξόδου να είναι `(1, 1)`. Πρακτικά η συνάρτηση αυτή λαμβάνει το μέσο όρο των στοιχείων κάθε καναλιού εξόδου, δίνοντας έτσι 512 στοιχεία στην τελευταία έξοδο. Τα στοιχεία αυτά τα λαμβάνουμε ως διάνυσμα χρησιμοποιώντας την `nn.Flatten()` και τα περνάμε από ένα τελικό γραμμικό επίπεδο εξόδου.

Αυτή τη φορά εκπαιδεύουμε το μοντέλο με τη μέθοδο Adam και ρυθμό εκμάθησης 10^{-3} για 20 εποχές. Όπως προηγουμένως, κρατήστε τις ενδιάμεσες τιμές της απώλειας και της ευστοχίας και απεικονίστε τις σε ένα γράφημα και αξιολογήστε το μοντέλο σας στο σύνολο δοκιμής. Συγκρίνετε τα αποτελέσματα με την προηγούμενη περίπτωση

6 Μεταφορά μάθησης - transfer learning (2 μονάδες)

Στην περίπτωση αυτή θα αξιοποιήσουμε ένα προεκπαιδευμένο μοντέλο ResNet34, το οποίο έχει εκπαιδευτεί στο ImageNet (θέστε `weights='DEFAULT'`). Το pytorch μας δίνει το μοντέλο έτοιμο προς χρήση με την κλάση `torchvision.models.resnet34`. Θα χρειαστεί στις αρχικές εικόνες να εφαρμόσετε τους ακόλουθους μετασχηματισμούς:

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

Παρατηρήστε ότι εφαρμόζουμε και augmentation (`RandomHorizontalFlip()`). Μπορεί να χρειαστεί να προσθέσετε και τον μετασχηματισμό `ToTensor()` αν φορτώνετε τις εικόνες μέσω του PIL.

Εκπαιδεύουμε το μοντέλο με SGD με χαμηλότερο learning rate, 0.001 και momentum 0.9, ώστε να κάνουμε μια μικρή προσαρμογή στα βάρη, για 5 εποχές. Αναφέρετε τα αποτελέσματα και σχολιάστε, όπως προηγουμένως. Παρατηρείτε διαφορές σε σχέση με τις προηγούμενες περιπτώσεις; Εξηγήστε.

7 Bonus: Αξιοποίηση δημογραφικών μεταβλητών (3 μονάδες)

Η ερώτηση αυτή δίνει bonus 3 μονάδες. Αν ξεπεράσετε το 10 η εργασία λαμβάνει 10 και οι επιπλέον μονάδες προσμετρώνται προσθετικά στον τελικό βαθμό στο ποσοστό της εργασίας (ώστε αν έχετε χάσει μονάδες από αλλού, μπορείτε να συμπληρώσετε από εδώ).

Προσαρμόστε τον `DataLoader` ώστε εκτός από τις τιμές `X, y` των εικόνων και των labels, να επιστρέφει και ένα διάνυσμα με τα δημογραφικά χαρακτηριστικά των ασθενών του συνόλου δεδομένων. Έτσι, ο `DataLoader` θα επιστρέφει μία τριάδα `X, p, y` όπου η μεταβλητή `p` είναι ένα διάνυσμα χαρακτηριστικών που σχετίζονται με την ηλικία (διαιρέστε με το 100 ώστε να έχετε τιμές μικρότερες της μονάδας) και το φύλο των ασθενών, καθώς και το μέρος του σώματος από το οποίο έγινε η λήψη της εικόνας. Για τις κατηγορικές μεταβλητές χρησιμοποιήστε την `pandas.get_dummies()` ώστε να τις μετατρέψετε σε one-hot. Το μοντέλο σας θα πρέπει να δέχεται δύο εισόδους. Την πρώτη είσοδο (εικόνα) θα την επεξεργάζεται με κάποιο από τα προηγούμενα μοντέλα. Τη δεύτερη είσοδο (διάνυσμα) θα την επεξεργάζεται από ένα γραμμικό επίπεδο με ReLU με διάσταση εξόδου 128. Τα δύο διανύσματα που προκύπτουν θα τα συνενώνετε με τη μέθοδο `torch.cat` πριν περάσετε σε ένα τελικό γραμμικό επίπεδο εξόδου.

Τι παρατηρείτε; Βοηθάει η χρήση των δημογραφικών στοιχείων στη βελτίωση της επίδοσης του μοντέλου;

Αναφορές

[1] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

[2] Philipp Tschandl, Christoph Rinner, Zoe Apalla, Giuseppe Argenziano, Noel Codella, Allan Halpern, Monika Janda, Aimilios Lallas, Caterina Longo, Josep Malvehy, et al. Human–computer collaboration for skin cancer recognition. *Nature Medicine*, 26(8):1229–1234, 2020.