OVERVIEW PLANT CLASSIFICATION

- **DATASET**
  - προβληματικά ως προς την κατανομή των κλάσεων. The long-tail problem.
  - μη σαφής διαχωρισμός train/val/test. Υπήρχε ένας φάκελος με τα train data (PlantCLEF2017Train1EoL) του EoL* και ένας φάκελος με τα test data (PlantCLEF2017Test), ο οποίος όμως είχε χύμα εικόνες και όχι σε φακέλους και μάλιστα οι εικόνες αυτές δεν ήταν από όλες τις κλάσεις. Υπήρχε δηλαδή πρόβλημα, καθώς το train set μπορεί να έχει X κλάσεις, αλλά το test set έχει λιγότερες.

★ **ΛΥΣΗ** Χρησιμοποιούμε τον ImageDataGenerator του TF2 για να διαχειριζόμαστε τις εικόνες και αυτό για να μπορεί να μπει σε μια ρουτίνα η διαδικασία. Διατηρούμε το PlantCLEF2017Train1EoL με τις 10000 κλάσεις του. Δημιουργούμε έναν δικό μας φάκελο PlantCLEF2017TestEoL, ο οποίος έχει test δεδομένα για ΟΛΕΣ τις κλάσεις. Προκειμένου να γίνει αυτό, ακολουθούμε την εξής διαδικασία: Πρώτα απ' όλα χρησιμοποιούμε το PlantCLEF2017OnlyTest.cs αρχείο για να βάλουμε σε ένα φάκελο όλες τις εικόνες που αντιστοιχούν σε μια κλάση και μάλιστα διατηρώντας το φορμαλισμό της ονοματοδοσίας, ονομάζουμε αυτό το φάκελο με το όνομα της ίδιας της κλάσης. Κάνουμε το ίδιο για όλες τις κλάσεις. Παρατηρούμε ότι έχουμε test δεδομένα για πολύ λίγες κλάσεις (το 1/5 από τις 10000 περίπου). Για να το αντιμετωπίσουμε αυτό, βρίσκουμε σε ποιες κλάσεις λείπουν test δεδομένα και προσθέτουμε από τις αντίστοιχες κλάσεις το 10% των train δεδομένων τους με κατώφλι την 1 φωτογραφία. Δηλαδή στην περίπτωση που κάποια κλάση στα train δεδομένα έχει λιγότερες από 10 φωτογραφίες (και αυτό θα σήμαινε ότι στο test δε θα περάσει καμία) έτσι ορίζουμε αυτή η κλάση να πάρει έστω 1 φωτογραφία στα test δεδομένα της. Ο κώδικας για αυτές τις διαδικασίες βρίσκεται στο αρχείο "create_test_dataset.py", το οποίο δημιουργεί τον φάκελο με τα ΕΛΛΙΠΗ test δεδομένα. Χρησιμοποιώντας το αρχείο .csv και "create_all_classes_test_dataset.py", το οποίο βρίσκει σε ποιες κλάσεις λείπουν δεδομένα και προσθέτει όπως είπαμε παραπάνω.

* ΛΥΣΗ (ΣΥΝΕΧΕΙΑ)

Το αρχείο "create_generators.py" δημιουργεί τους ImageDataGenerators που χρειαζόμαστε. Το validation set προκύπτει αυτόματα με split από το train set (10%-20% split οι καλύτερες επιλογές). Γενικά δοκιμάστηκαν και τα δύο ποσοστά split (0.10 - 0.20), με το 20% να επικρατεί. [redacted]

Όσον αφορά το πρόβλημα του unbalanced dataset (long-tail problem), η λύση που προτιμήθηκε είναι να δημιουργήσουμε fake data. Έγιναν δοκιμές σε διάφορους μετασχηματισμούς (shift, rotate, flip, crop etc.) και επιλέχθηκαν αυτοί που φαίνονται στο αρχείο "data_augmentation.py" καθώς και οι αντίστοιχες τιμές που φαίνονται. ΔΕΝ επιλέγουμε να εφαρμόσουμε έναν μετασχηματισμό/φιλτράρισμα τυχαία σε μία εικόνα κατά τη διαδικασία της εκπαίδευσης (on-the-fly), διατηρώντας έτσι τον αριθμό των εικόνων ίδιο, ΑΛΛΑ επιλέγουμε να ορίσουμε έναν αριθμό (1000 εικόνες) και να εφαρμόσουμε a priori τους μετασχηματισμούς ΤΟΣΟ ΩΣΠΟΥ η κάθε κλάση να έχει τον αριθμό των μετασχηματισμένων εικόνων [redacted] που ορίσαμε. Αυτό έγινε γιατί παρατηρήθηκε ΤΕΡΑΣΤΙΑ διαφορά στο accuracy.

Με λίγα λόγια, με αυτό τον τρόπο μπορούμε να ορίσουμε έναν αριθμό εικόνων και να εφαρμόσουμε τυχαίους μετασχηματισμούς μέχρι να φτάσουμε αυτό τον αριθμό per class. Καλό είναι ο αριθμός αυτός να μην είναι μεγάλος πολύ (το 1000 είναι καλό) γιατί μετά παρατηρείται το φαινόμενο του να έχουμε την ίδια εικόνα δύο φορές.

Επειδή οι 10000 κλάσεις του PlantCLEF EoL είναι πολλές και δεν είναι στην τάξη του μεγέθους του δικού μας προβλήματος *, επιλέγουμε 100 κλάσεις τυχαία και κάνουμε augment αυτές. Δημιουργούμε έτσι τον φάκελο 100ClassesPlantCLEF2017Train1EoL ο οποίος έχει μέσα τον φάκελο data με τα δεδομένα χωρίς το augment και τον φάκελο augmented data που έχει τα augmented data που χρησιμοποιούμε.

Αντίστοιχα δημιουργούμε το φάκελο 100ClassesPlantCLEF2017TestEoL που περιλαμβάνει τα test data για τις αντίστοιχες 100 κλάσεις.

OVERVIEW

- **ΔΙΚΤΥΑ**

- γενικά η πρώτη προτίμηση ήταν να δοκιμάσουμε τα ίδια δίκτυα που δοκίμασε στο paper του ο νικητής του διαγωνισμού. Το πρόβλημα είναι ότι όταν δοκιμάστηκαν αυτά τα δίκτυα, δοκιμάστηκαν με όλο το dataset των 10000 κλάσεων και αυτό εκτός του ότι ήταν χρονοβόρο, δεν επέτρεπε κιόλας να γίνουν οι κατάλληλες δοκιμές και τα απαραίτητα fine-tune στις hyperparameters. Επίσης, παρατηρήθηκε το φαινόμενο το [redacted] validation set accuracy να μένει στο ≅40%, ενώ το [redacted] training accuracy έφτανε το ≅100%. Υπήρχε, δηλαδή, πρόβλημα <u>OVERFIT</u>. Τα δίκτυα δοκιμάστηκαν να εκπαιδευτούν έχοντας ως αρχικοποίηση τα βάρη του Imagenet, κάτι που <u>ΑΚΟΜΑ</u> ισχύει στα τωρινά experiments. Έγινε προσπάθεια να διαπιστωθεί αν είναι καλύτερο τα βάρη να είναι freezed, semi-freezed ή non freezed. Σε γενικές γραμμές τα freezed βάρη δεν απέδωσαν και αυτό γιατί τα high level features του CNN για το πρόβλημα της ταξινόμησης των φυτών δε μπορούν να είναι ίδια με τα extracted/learned high level features του προβλήματος της ταξινόμησης των 10000 <u>ΓΕΝΙΚΩΝ</u> κλάσεων του dataset. Οπότε οδηγηθήκαμε σε μια semi-freezed λύση, όπου τα low level layers είναι freezed (αυτά που εξάγουν την πληροφορία των ακμών, γωνιών κλπ.), ενώ τα high level layers είναι unfreezed, για να μπορούν να προσαρμοστούν στο πρόβλημα της ταξινόμησης των φυτών. Το δίκτυο αφότου κάνει την εξαγωγή των χαρακτηριστικών καταλήγει σε ένα <u>Global Average Pooling</u>, έπειτα από το οποίο επιλέγεται να τοποθετηθεί για ταξινόμηση ένα <u>Dense/Fully Connected Layer</u> με <u>2048</u> νευρώνες και <u>0.5 Dropout</u> (για να εξασφαλίσει ότι οι νευρώνες θα μάθουν καλά) και τέλος ένα τελευταίο ακόμα Dense/Fully Connected με νευρώνες τόσους όσες και οι κλάσεις μας. Η αρχιτεκτονική αυτή γενικά κρατήθηκε/υιοθετήθηκε και στις μετέπειτα δοκιμές. Τα semi-freezed βάρη απέδωσαν καλύτερα από ότι πριν τα freezed. Τέλος, δοκιμάστηκε να γίνουν τελείως <u>unfreeze</u> τα βάρη όλα και να είναι <u>learnable</u> και αυτό απέδωσε καλύτερα από ό,τι άλλο.

- Έχοντας περίπου μια αρχιτεκτονική στο μυαλό, είχε παραμείνει το πρόβλημα του αριθμού των κλάσεων και του unbalanced dataset, το οποίο συνεπαγόταν να χάνεται πολύς χρόνος στις δοκιμές. Αποφασίστηκε να μειωθούν οι κλάσεις και μάλιστα με τον εξής τρόπο: να κρατηθούν μόνο οι κλάσεις που είχαν πάνω από 30 samples → ≅2000 κλάσεις. Έτσι και μειώθηκε ο αριθμός των κλάσεων και έγινε κάπως πιο balanced το dataset. Ακόμα δεν είχαμε κάνει augmentation και δεν είχαμε φτιάξει τα test δεδομένα (όπως αναφέρεται στις σελ. 1,2). Το πρόβλημα του <u>OVERFIT</u> συνέχιζε να υπάρχει. Το πρόβλημα αυτό γενικά μπορεί να οφείλεται/οφείλονταν στο γεγονός ότι τα δεδομένα για κάποιες κλάσεις (τις περισσότερες) ήταν λίγα και τα δίκτυα μάθαιναν σε βάθος, οπότε γινόταν memorize. Δηλαδή το δίκτυο έπεφτε στη λούπα του να μάθει όπως ήταν κάποιες εικόνες (σχεδόν δεν είχε πολλές) και έτσι έχανε γενικευτική ικανότητα. Δοκιμάστηκε να κρατηθούν οι κλάσεις που έχουν πάνω από 50 samples → ≅1000 κλάσεις, αλλά το ίδιο πρόβλημα παρέμενε. Έτσι λοιπόν, αποφασίστηκε να γίνει augmentation έτσι ώστε να έχουμε σταθερό αριθμό εικόνων για κάθε κλάση (όπως περιγράφεται στις σελίδες 1,2). Μεταβήκαμε λοιπόν από το Fig. 1 στο Fig. 2:

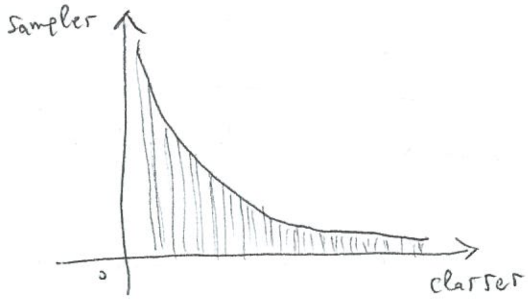

Fig. 1

→

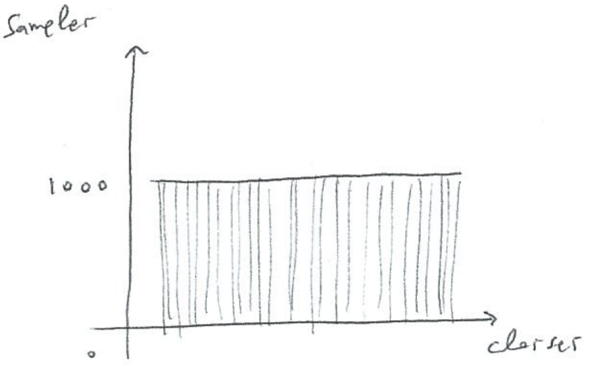

Fig. 2

Διευκρινίζεται πως (όπως φαίνεται στα Fig. 1 και Fig. 2) το augmentation εισάγει συγκεκριμένους τυχαίους μετασχηματισμούς, ώστε να δημιουργηθούν νέα fake data (και έτσι να αυξηθεί ο αριθμός των sampler για κάθε κλάση και να γίνει σταθερός). Όπως αναφέρεται και στις σελ. 1,2 το augmentation δε γίνεται on-the-fly (λόγω memory error) αλλά αποθηκεύονται οι εικόνες σε φάκελο ακολουθώντας το φορμαλισμό της ονοματοδοσίας. Augmentation έχει ως τώρα πραγματοποιηθεί σε 100 τυχαίες κλάσεις, καθώς το 100 σα νούμερο είναι στην τάξη μεγέθους του dataset που θα έχουμε από Μάρτιο και είναι και ένα καλό νούμερο υπολογιστικά για να γίνονται τα experiments. Έτσι λοιπόν, στον (D:) του λάπτοπ του Βασίλη έχουμε τους φακέλους:

100ClassesPlantCLEF2017Train1EOL → 100 τυχαίες κλάσεις με augmented sampler = 1000 όπως περιγράφηκε παραπάνω

100ClassesPlantCLEF2017TestEOL → για αυτές τις 100 κλάσεις ένα test dataset μειωμένο όπως περιγράφεται σελ. 1,2

αυτά χρησιμοποιούμε στο σημείο που βρισκόμαστε τώρα...

1000ClassesPlatCLEF2017Train1EOL → 1000 κλάσεις διαλεγμένες έτσι ώστε κάθε κλάση να έχει >50 sampler (χωρίς augmentation)

2000ClassesPlantCLEF2017Tran1EOL ← 2000 κλάσεις διαλεγμένες έτσι ώστε κάθε κλάση να έχει >30 sampler (χωρίς augmentation)

δε χρησιμοποιούνται πια...

PlantCLEF2017TestEOL → test data για τις 10000 κλάσεις του EOL (Βλ. σελ. 1,2)

PlantCLEF2017Train1EOL → 10000 κλάσεις του EOL χωρίς augmentation (για λόγους εξοικονόμησης σκληρού δίσκου)

μπορεί να χρησιμοποιηθούν στο μέλλον για να αποκόψουμε 500 κλάσεις αντί για 100

Να σημειωθεί εδώ πως αν θέλουμε να αποκόψουμε 500 κλάσεις αντί για 100 και να κάνουμε augment, τότε πρώτα απ' όλα πρέπει να τσεκάρουμε το σκληρό δίσκο και να εκτιμήσουμε το πόσο χώρο θα πιάσουν σε αυτόν και δεύτερον να χρησιμοποιήσουμε το αρχείο "data-augmentation.py", το οποίο παίρνει ως input έναν φάκελο με X κλάσεις και βγάζει X κλάσεις με augmented data.

Μια σημείωση ακόμα που πρέπει να γίνει και αφορά τη διαδικασία είναι ότι δημιουργώντας fake data δε λύνουμε ακριβώς το πρόβλημα. Ουσιαστικά δε δημιουργούμε νέα δεδομένα, αλλά εισάγουμε μια τυχαιότητα στα ήδη υπάρχοντα και έτσι αυξάνουμε τον αριθμό τους. Κάτι που σχετίζεται με αυτό, είναι το γεγονός ότι χρησιμοποιώντας τα augmented data το training και validation accuracy φτάνει μετά από ένα σημείο το ≅100%, αλλά το test accuracy είναι αρκετά μακριά. Για να γίνει περισσότερο κατανοητό αυτό, θα παραθέσουμε τις test set ακρίβειες του NASNetMobile δικτύου, στο οποίο έγιναν πολλές δουλειές, και αυτό γιατί είναι ένα δίκτυο που αξιοποιεί το NAS που είναι state of the art και επίσης είναι κατάλληλο για mobile, δηλαδή είναι μικρό σε μέγεθος και κάνει για την εφαρμογή που το θέλουμε.

Network Architecture Search

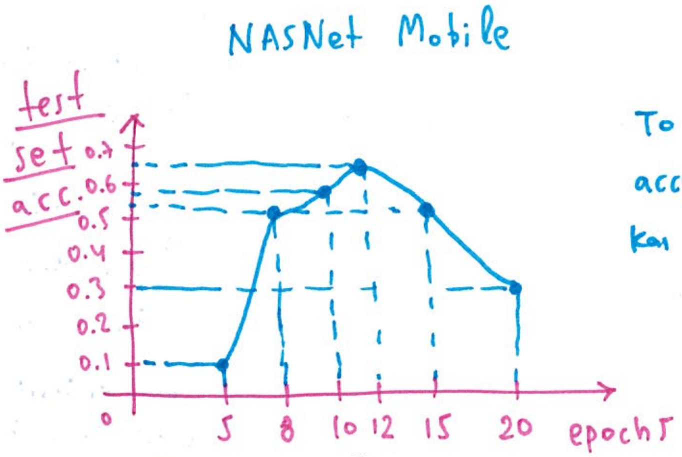

Fig. 3

Το NASNetMobile φτάνει test set accuracy 62% στις 12 Epochs και μετά φαίνεται να μεταβαίνει.

Και δεν είναι μόνο το NASNetMobile στο οποίο παρατηρήθηκε αυτό. Η εξήγηση που μπορεί να δοθεί είναι η εξής: πρώτον τα data που έχουμε δημιουργήσει είναι fake και προκύπτουν από συγκεκριμένες εικόνες με τυχαίους μετασχηματισμούς. Αυτό βοηθάει πολύ στο validation acc. γιατί ουσιαστικά το δίκτυο δε βλέπει εικόνες που δεν έχει ξαναδεί, αλλά βλέπει παραλλαγές εικόνων που έχει ξαναδεί. Έτσι εξηγείται το ότι το validation acc φτάνει και αυτό το ≅100%. Τώρα όσον αφορά το test set accuracy, στο Fig. 3 φαίνεται πως μετά από ένα σημείο το δίκτυο αρχίζει να μεταβαίνει. Σημαντικό είναι να σημειώσουμε πως οι εικόνες του test set είναι όντως εικόνες που δεν έχει ξαναδεί ποτέ. Το δίκτυο, λοιπόν, μετά από ένα σημείο φαίνεται να χάνει τη γενικευτική του ικανότητα, γεγονός που θέλει προσοχή!

Προκειμένου να γίνει εύκολα η σύγκριση των εκπαιδευμένων δικτύων (και η εν δυνάμει μελλοντική χρήση κάποιων σε ensembling), αυτά αποθηκεύονται στον φάκελο checkpoints.

Ο φορμαλισμός που ακολουθείται είναι ο εξής: ο κάθε φάκελος ονομάζεται με βάση το όνομα του δικτύου και μέσα περιέχει φάκελο που ονομάζεται με βάση τον αριθμό των κλάσεων που συμμετέχουν στην ταξινόμηση. Ο φάκελος αυτός (π.χ. 100_classes) περιέχει μέσα το μοντέλο σε .h5 αρχείο και ένα .log αρχείο που περιέχει το ιστορικό της εκπαίδευσης.

Ακόμα, ο φάκελος code περιέχει όλα τα αρχεία κώδικα που έχουν γραφτεί - παλιά και καινούρια.

Το PlantNet είναι μια προσπάθεια δημιουργίας ενός μικρού custom νευρωνικού, το οποίο μπορεί να αποδίδει ικανοποιητικά. Βέβαια, και το NASNetMobile που παρουσιάστηκε παραπάνω είναι δίκτυο κατάλληλο για χρήση σε κινητά. Στο φάκελο code υπάρχει το αρχείο "tflite_converter.py", το οποίο μετατρέπει τα .h5 αρχεία σε tflite αρχεία κατάλληλα για χρήση σε κινητό.

ΣΥΝΟΨΗ + ΠΟΥ ΒΡΙΣΚΟΜΑΣΤΕ + ΕΠΟΜΕΝΑ ΒΗΜΑΤΑ

→ Συνεχίζουμε πειράματα με τις 100 κλάσεις με τα augmented samples

100ClassesPlantCLEF2017Train
100ClassesPlantCLEF2017Test

→ Αποδεδειγμένα, ένα μεγάλο δίκτυο μπορεί να φτάσει κοντά στο 80% test set accuracy σε αυτό το 100 classes dataset. Αποφασίσαμε, όμως, να δοκιμάσουμε ένα πιο μικρό για λόγους που αναφέρθηκαν παραπάνω και εφόσον:

→ Το NASNetMobile έχει test set accuracy 62%, κρίνεται σκόπιμο να συνεχιστούν πειράματα με αυτό

Τι θα μπορούσε να αλλάξει

- (1) Dense με 2048 neurons + Dropout 0.5
  - 2048: experiment with
  - 0.5: experiment with
- (2) Learning Rate Scheduler (αρχείο "schedule.py")
  - "schedule.py": experiment with → μπορεί να βρεθεί ένα καλύτερο ελάχιστο για τον Adam
- (3) Number of samples of Dataset
  - → αντί το augmentation να δίνει 1000 samples, να δίνει 500 και το dataset το νέο να έχει 500 samples

Αυτό είναι πιθανό να οδηγήσει σε καλύτερη γενικευτική ικανότητα...

ΓΕΝΙΚΑ: Και οι (3) αλλαγές ((1),(2),(3)) που προτείνονται έχουν δοκιμαστεί στο παρελθόν και έχουμε καταλήξει ότι αυτό είναι το καλύτερο, ΑΛΛΑ χωράει ακόμα κάποιας περαιτέρω δοκιμής, γι' αυτό και προτείνονται!

→ Το PlantNet (δηλαδή το δικό μας custom CNN) θέλει πολλές δοκιμές για να φτάσει σε ακρίβεια του NASNetMobile, οπότε κρίνεται σκόπιμο να ΜΗΝ εστιάσουμε σε αυτό (αφού μάλιστα έχουν και κοντινό/παρόμοιο σκορ τα δίκτυα)

→ Λύση εναλλακτική είναι να χρησιμοποιηθεί ένα μεγάλο δίκτυο, όπως αυτό που έχει φτάσει ≅80% test set accuracy και να γίνει προσπάθεια μέσω της μετατροπής του .h5 αρχείου σε .tflite να μειωθεί ο όγκος του, ώστε να χωρέσει σε κινητό!

ΚΑΠΟΙΕΣ ΠΕΡΑΙΤΕΡΩ ΣΥΜΒΟΥΛΕΣ + TIPS

→ Τα μικρά δίκτυα τρέχουν σχετικά γρήγορα στο λάπτοπ του Βασίλη (δηλαδή τα 15 Epochs του NASNetMobile θέλουν περίπου 4 ώρες. Τα μεγάλα δίκτυα είναι καλύτερο να τρέξουν στον υπολογιστή στην Ξάνθη

↓

AnyDesk: 240656324
xanthi25410

στον υπολογιστή αυτό έχει διατηρηθεί ακριβώς ο ίδιος φορμαλισμός ονοματοδοσίας στους φακέλους, αλλά έχουν τρέξει εκεί τα μεγάλα CNN

→ Στο φάκελο D:\advent\code υπάρχουν πολλά έτοιμα δίκτυα τα οποία μπορούν να τρέξουν αλλά κάνοντας το αρχείο τους import στο "classification.py"

→ ΥΠΟΨΙΑΖΟΜΑΙ πως θα είναι σχεδόν αδύνατο να φτάσουμε παραπάνω test set accuracy ή αν φτάσουμε θα φτάσουμε ελάχιστα. Στην περίπτωση που χρειαζόμαστε παραπάνω πρέπει να σκεφτούμε τα εξής:

1. Το δικό μας dataset θα είναι πιο balanced και με καλύτερες (ελπίζω) εικόνες, το οποίο θα συνεπάγεται ↗.
2. Στο paper του αυτός ΔΕΝ ΚΑΤΑΦΕΡΕ να ξεπεράσει το 65% validation set accuracy ΧΩΡΙΣ ENSEMBLE (βέβαια εκείνος είχε 10000 κλάσεις)

όπως φαίνεται στο SubmissionModelsOverview.xslx. Ακόμα, δοκίμασε να χρησιμοποιήσει σαν αρχικοποίηση του τελικού του δικτύου βάρη από προηγούμενα, κάτι το οποίο μπορεί να δοκιμαστεί και σε συνδυασμό με ENSEMBLE να αποδώσει. ΑΥΤΗ ίσως να είναι η καλύτερη λύση και αυτή που θα αποδώσει το τελικό ΒΕΛΤΙΣΤΟ ΔΙΚΤΥΟ.