Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

Using Artificial Neural Networks for Zero-shot Learning

Χρήση Τεχνητών Νευρωνικών Δικτύων για Ταξινόμηση Προτύπων Άγνωστων Κατηγοριών με Μηδενική Υποστήριξη Δεδομένων

Διπλωματικη Εργασια

του

ΧΟΧΛΑΚΗ Δ. ΓΕΩΡΓΙΟΥ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής

Αθήνα, Νοέμβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

Using Artificial Neural Networks for Zero-shot Learning

Χρήση Τεχνητών Νευρωνικών Δικτύων για Ταξινόμηση Προτύπων Άγνωστων Κατηγοριών με Μηδενική Υποστήριξη Δεδομένων

Διπλωματικη Εργασια

του

ΧΟΧΛΑΚΗ Δ. ΓΕΩΡΓΙΟΥ

Επιβλέπων: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Νοεμβρίου 2020.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αλέξανδρος Ποταμιάνος	Πέτρος Μαραγκός	Κωνσταντίνος Τζαφέστας
Αναπληρωτής Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Αθήνα, Νοέμβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Σηματων, Ελεγχου και Ρομποτικης

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Γεώργιος Χοχλάκης

5η Νοεμβρίου 2020

Περίληψη

Σε αυτήν την διπλωματική εργασία, ασχολούμαστε με προβλήματα του τομέα της *Τεχνητής Νοημοσύνης*. Χρησιμοποιούμε Μηχανική Μάθηση και συγκεκριμένα *Τεχνητά Νευρωνικά Δίκτυα* για να λύσουμε το πρόβλημα της Ταξινόμησης Προτύπων *Άγνωστων Κατηγοριών* με *Μηδενική Υποστήριξη* Δεδομένων. Ο επίσημος όρος που χρησιμοποιείται στη βιβλιογραφία για το πρόβλημα, στα αγγλικά, είναι **Zero-shot Learning**. Ακόμη, αξίζει να αναφερθούμε στο **Few-shot Learning**, όπου οι κατηγορίες είναι μεν άγνωστες, μας δίδονται δε λίγα δείγματα αυτών των κατηγοριών, τα οποία ονομάζουμε support set.

Οι καλύτερες τεχνικές για Zero-shot Learning βασίζονται στα Παραγωγικά Δίκτυα (Generative Networks). Ο βασικός αλγόριθμος που ακολουθούνε είναι: Πρώτον, εκπαιδεύεται ένα Παραγωγικό Δίκτυο με βάση βοηθητικές περιγραφές των κατηγοριών και τα στιγμιότυπα τους. Αφού εκπαιδευτεί το Παραγωγικό Δίκτυο, τότε το χρησιμοποιούμε για να παράξουμε συνθετικά στιγμιότυπα των κατηγοριών που αντιμετωπίζουμε την ώρα της αποτίμησης, χρησιμοποιώντας τις αντίστοιχες περιγραφές. Τέλος, με αυτά τα δείγματα, εκπαιδεύουμε έναν απλό ταξινομητή.

Σε αυτήν την εργασία, προτείνουμε μια νέα μέθοδο για Zero-shot Learning, η οποία επαυξάνει υπάρχοντες αλγορίθμους, βασισμένους στο βασικό αλγόριθμο που παρουσιάσαμε, χρησιμοποιώντας τον ταξινομητή που χρησιμοποιούμε την ώρα της αποτίμησης και την ώρα της εκπαίδευσης του Παραγωγικού Δικτύου. Η χρησιμοποίηση του γίνεται συμπεριλαμβάνοντας το σφάλμα ταξινόμησης του στην εκπαίδευση του Παραγωγικού Δικτύου. Για να το κάνουμε αυτό, ο ταξινομητής θα πρέπει να μην εξαρτάται αποκλειστικά για την εκπαίδευση του από τα δείγματα του Παραγωγικού Δικτύου και να είναι ευέλικτος αναφορικά με τις κατηγορίες που μπορεί να ταξινομήσει. Τέτοιες ιδιότητες πρέπει να έχουν και ταξινομητές που χρησιμοποιούνται σε προβλήματα Few-shot Learning, συνεπώς, χρησιμοποιούμε έναν τέτοιον. Την ώρα της εκπαίδευσης και της αποτίμησης, θεωρούμε τα στιγμιότυπα που παράγουμε με το Παραγωγικό Δίκτυο ως το support set του ταξινομητή.

Παρατηρούμε πειραματικά βελτίωση στην ακρίβεια ταξινόμησης με βάση τον αλγόριθμο μας σε σύγκριση με τον απλό αλγόριθμο για Zero-shot Learning και πετυχαίνουμε state-of-the-art σε αρκετά προβλήματα. Επιπλέον, δείχνουμε πως η χρήση του ταξινομητή μόνο κατά την εκπαίδευσης ή μόνο την αποτίμησης βελτιώνει την ακρίβεια, ενώ τα προτερήματα της χρήσης του σε κάθε ξεχωριστή περίσταση σχεδόν αθροίζονται.

Λέξεις Κλειδιά

Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Βαθειά Μάθηση, Μηδενική Υποστήριξη Δεδομένων, Σύνολο Υποστήριξης, Παραγωγικά Δίκτυα

Abstract

In this diploma thesis, we are concerned with tasks in the domain of *Artificial Intelligence*. We utilize Machine Learning and, in particular, Artificial Neural Networks, to solve the problem of **Zero-shot Learning**, the task of evaluating our models on classification tasks where the patterns do not belong to any category seen during the model's training, and no supporting examples of these novel categories is provided. Moreover, **Few-shot Learning** is a similar task worth mentioning. In this setting, a small support set of samples from the test categories is provided in order for an algorithm to be able to adjust its parameters or extract the necessary knowledge.

Contemporary approaches to Zero-shot Learning are based on Generative Networks. The basic algorithm being used is as follows: First, a Generative Network is trained using the samples and auxiliary descriptions that are provided for training. After the Generative Network has been trained, we use it to generate synthetic examples of the categories we are to classify during testing, using the respective descriptions. Lastly, based on these samples, we train a simple classifier.

In this work, we propose a novel framework for Zero-shot Learning that augments already existing algorithms, based on the aforementioned basic one, by including the classifier used during testing in the training of the Generative Network. We do so by exploiting the classifier's classification loss for the training of the Generative Network. However, such a classifier must not depend on the Generative Network's samples for training and must be flexible w.r.t its label space. Such properties are also essential for Few-shot Learning, therefore we leverage such an algorithm. During training and testing, samples generated by the Generative Network are treated as the support set of the classifier, based on which it classifies real samples.

We empirically observe gains in performance compared to simple Zero-shot Learning algorithms. In addition, given that some of these algorithms achieved state-of-the-art performance in Zero-shot Learning benchmarks, we now achieve that in various cases. Also, we show that that the usage of the Few-shot learner only during training or only during testing still improves the accuracy of the Zero-shot learner, while the advantages of using it in either setting seem to be additive to one another.

Keywords

Artificial Intelligence, Machine Learning, Deep Learning, Zero-shot Learning, Few-shot Learning, Generative Networks

σε όσους οι διπλωματικές εργασίες είναι ένα άγνωστο άγνωστο

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον καθηγητή Αλέξανδρο Ποταμιάνο για την επίβλεψη αυτής της διπλωματικής εργασίας και για την καθοδήγηση. Επίσης ευχαριστώ ιδιαίτερα τον Ευθύμη Γεωργίου για την καθοδήγηση, την εξαιρετική συνεργασία που είχαμε και το χρόνο που μου διέθετε σε καθημερινή βάση. Τέλος θα ήθελα να ευχαριστήσω τους γονείς για τα γονίδιά μου και μέγιστη δυνατή μέριμνα για την εκπαιδευσή μου που μου επέτρεψαν να φτάσω μέχρι αυτό το σημείο.

Αθήνα, Νοέμβριος 2020

Γεώργιος Χοχλάκης

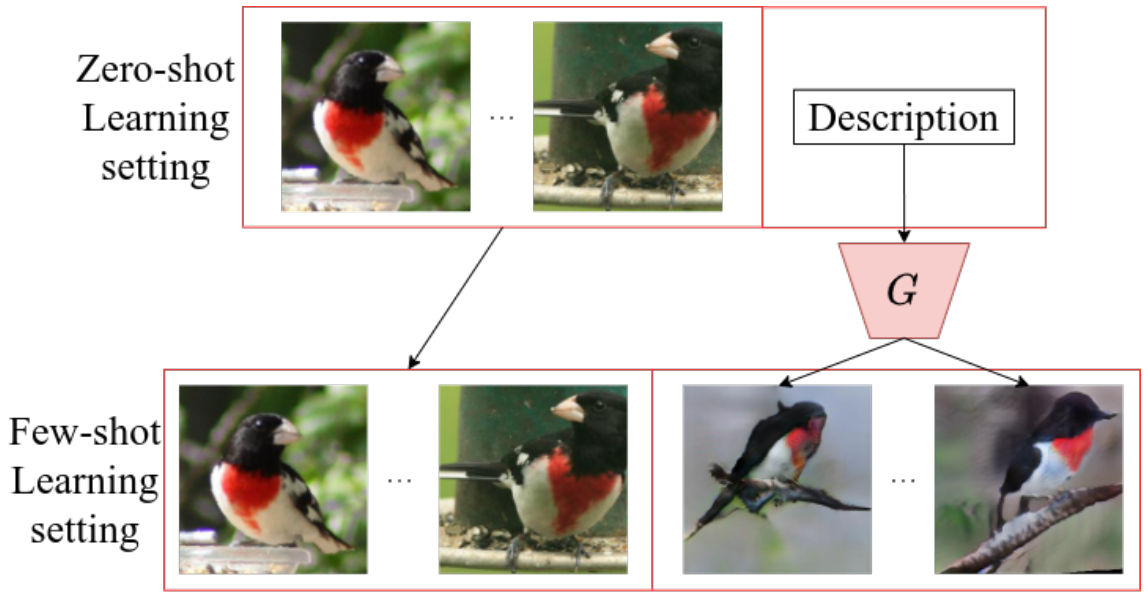
Contents

List of Figures

List of Tables

Κεφάλαιο 0

Εκτεταμένη Περίληψη



Σχήμα 1: *Αφαιρετική αναπαράσταση της αναγωγής του Zero-shot Learning σε Few-shot Learning.*

Σε αυτήν την διπλωματική εργασιά, ασχολούμαστε με το πρόβλημα της Ταξινόμησης Προτύπων *Άγνωστων Κατηγοριών* με *Μηδενική Υποστήριξη* Δεδομένων. Αυτό, συνοπτικά, σημαίνει ότι την ώρα που εξετάζουμε την απόδοση ενός μοντέλου, αυτό καλείται να ταξινομήσει πρότυπα (π.χ. εικόνες) τα οποία ανήκουν σε κατηγορίες στις οποίες το μοντέλο δεν είχε πρόσβαση κατά την εκπαίδευση των παραμέτρων του. Όμως, με δεδομένο ότι οι κατηγορίες είναι άγνωστες, για να διευκολυνθεί το έργο της κατηγοριοποίησής τους, δίνονται βοηθητικές περιγραφές των κλάσεων και την ώρα της εκπαίδευσης και, προπαντός, την ώρα της αποτίμησης. Ο επίσημος όρος που χρησιμοποιείται στη βιβλιογραφία για το πρόβλημα, στα αγγλικά, είναι **Zero-shot Learning**. Ακόμη, αξίζει να αναφερθούμε στην περίπτωση που οι κατηγορίες είναι μεν άγνωστες, μας δίδονται δε λίγα δείγματα αυτών των κατηγοριών, ουσιαστικά ένα υποστηρικτικό σύνολο (support set), για να προσαρμόσουμε τον αλγόριθμο μας σε αυτά ή να εξάγουμε την απαραίτητη γνώση. Αυτό το πρόβλημα λαμβάνει την ονομασία **Few-shot Learning**.

Η γενίκευση σε άγνωστες κατηγορίες έχει αναδειχθεί σε ένα πολύ σημαντικό και δύσκολο πρόβλημα. Τα τελευταία χρόνια, οι καλύτερες τεχνικές που χρησιμοποιούνται για την

επίλυση του Zero-shot Learning βασίζονται στα Παραγωγικά Δίκτυα. Ο βασικός αλγόριθμος που ακολουθούνε είναι: Πρώτον, εκπαιδεύεται ένα Παραγωγικό Δίκτυο με βάση τις περιγραφές και τα στιγμιότυπα των κλάσεων που δίδονται για την εκπαίδευση. Αφού εκπαιδευτεί το Παραγωγικό Δίκτυο, τότε το χρησιμοποιούμε για να παράξουμε συνθετικά στιγμιότυπα των κλάσεων που αντιμετωπίζουμε την ώρα της αποτίμησης, χρησιμοποιώντας τις αντίστοιχες περιγραφές. Τέλος, με αυτά τα δείγματα εκπαιδεύουμε έναν απλό ταξινομητή, ο οποίος ταξινομεί τα δείγματα εισόδου του στις συγκεκριμένες κατηγορίες που παράξαμε.

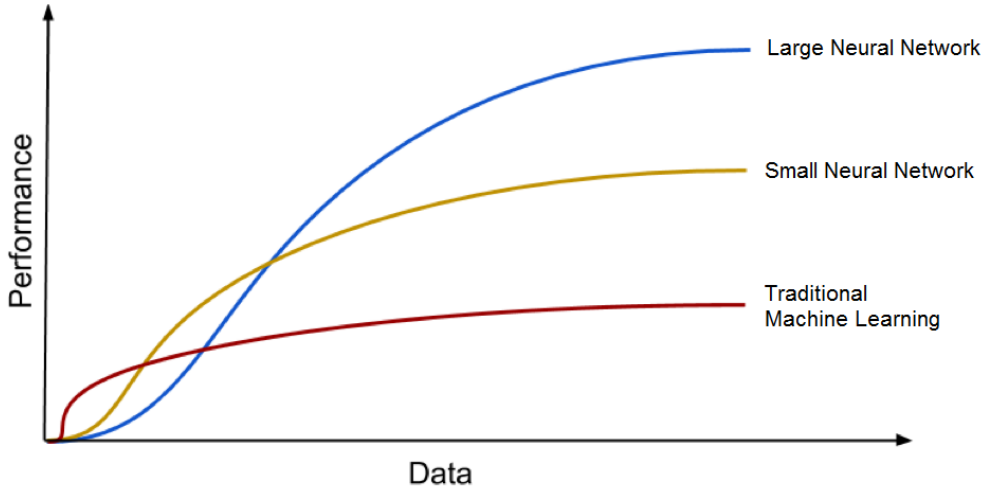
Σε αυτήν την εργασία, προτείνουμε μια νέα μέθοδο για Zero-shot Learning, η οποία επαυξάνει υπάρχοντες αλγορίθμους, βασισμένους στο βασικό αλγόριθμο που παρουσιάσαμε, χρησιμοποιώντας τον ταξινομητή που χρησιμοποιούμε την ώρα της αποτίμησης και την ώρα της εκπαίδευσης του Παραγωγικού Δικτύου. Η χρησιμοποίηση του γίνεται συμπεριλαμβάνοντας το σφάλμα ταξινόμησης του στην εκπαίδευση του Παραγωγικού Δικτύου, που είναι σφάλμα το οποίο προέρχεται από τον ταξινομητή που χρησιμοποιούμε την ώρα της αποτίμησης. Για να το κάνουμε αυτό, ο ταξινομητής θα πρέπει να μην εξαρτάται αποκλειστικά για την εκπαίδευση του από τα δείγματα του Παραγωγικού Δικτύου αλλά να βασίζεται κυρίως σε πραγματικά δείγματα, ώστε να μπορεί να χρησιμοποιηθεί παράλληλα με την εκπαίδευση του Παραγωγικού Δικτύου, και να είναι ευέλικτος αναφορικά με τις κατηγορίες που μπορεί να ταξινομήσει, αφού οι κατηγορίες της εκπαίδευσης και της αποτίμησης είναι διαφορετικές. Τέτοιες ιδιότητες πρέπει να έχουν και ταξινομητές που χρησιμοποιούνται σε προβλήματα Few-shot Learning, αφού κι εκεί οι κατηγορίες της αποτίμησης μπορεί να είναι άγνωστες, ενώ ένας τέτοιος ταξινομητής προφανώς χρειάζεται μόνο πραγματικά δεδομένα. Συνεπώς, εκμεταλλευόμαστε έναν τέτοιον αλγόριθμο για ταξινομητή. Την ώρα της εκπαίδευσης και της αποτίμησης, θεωρούμε τα στιγμιότυπα που παράγουμε με το Παραγωγικό Δίκτυο ως το υποστηρικτικό σύνολο του ταξινομητή και καλούμαστε με βάση αυτό να ταξινομήσουμε τα πραγματικά στιγμιότυπα. Μία αναπαράσταση της διαδικασίας αναγωγής της ταξινόμησης από πλαίσιο του Zero-shot Learning σε αυτό του Few-shot Learning με τη θεώρηση των παραγόμενων δειγμάτων ως support set φαίνεται στο Γράφημα 1.

Πειραματικά, παρατηρούμε βελτίωση στα αποτελέσματα συγκριτικά με τον απλό αλγόριθμο Zero-shot Learning και εξετάζουμε με περαιτέρω πειράματα τις πηγές της παρατηρούμενης βελτίωσης.

0.1 Εισαγωγή

Η Βαθιά Μάθηση έχει πολλές επιτυχίες σε πολλά προβλήματα, όπως είναι η Όραση Υπολογιστών [21, 22, 23], η Κατανόηση Γλώσσας και Ομιλίας [24, 25] ή τα Γραφικά Υπολογιστών [26, 27]. Όμως, έχει βοηθήσει και σε διάφορα επιστημονικά πεδία, όπως οι Νευροεπιστήμες [28, 29], η Τηλεπισκόπηση [30, 31] ή η Ιατρική [8, 32]. Παρά την ποικιλία στις εφαρμογές, υπάρχει ένας κοινός παρονομαστής: δεδομένα και πόροι. Η πλειονότητα των μοντέρνων εφαρμογών όχι απλά επωφελούνται αλλά χρειάζονται πάρα πολλά δεδομένα για να πετύχουν τις «εξωπραγματικές» επιδόσεις. Αυτό γίνεται εμφανές και από αυτές τις συχνά «υπεράνθρωπες» επιδόσεις (π.χ. [33, 34]) αλλά και από τις επιδόσεις των αλγορίθμων σε καταστάσεις όπου υπάρχουν λίγα δεδομένα, όπως φαίνεται και στο Γράφημα 2. Σε αυτό φαίνεται η πτώση της απόδοσης των Βαθιών Νευρωνικών Δικτύων όταν τα δεδομένα είναι

λίγα, απόδοση η οποία πέφτει και κάτω από Ρηχά Δίκτυα και κλασσικές μεθόδους.



Σχήμα 2: *Σύγκριση επιδόσεων συναρτήσει της ποσότητας διαθέσιμων δεδομένων. Πηγή: [1].*

Σε συνδυασμό με τις περιβαλλοντικές επιπτώσεις που έχουν οι αλγόριθμοι Βαθειάς Μάθησης, όπως αποφάνθηκαν στο [35], οι τεχνικές Βαθιάς Μάθησης γίνονται βιώσιμη εναλλακτική μόνο όταν υπάρχουν άπλετα δεδομένα.

Έχοντας υπόψιν αυτές τις δυσκολίες, πολλές προσπάθειες έχουν γίνει για να αντιμετωπιστούν οι δυσκολίες των μοντέλων σε περιβάλλοντα με λίγα δεδομένα. Τα τελευταία χρόνια, έχουν προταθεί πολλά τέτοια πλαίσια για να καθοδηγήσουν λελογισμένα τις προσπάθειες των ερευνητών, και πολλές δουλειές έχουν γίνει πάνω σε αυτά. Τέτοια πλαίσια είναι τα Zero-shot Learning και Few-shot Learning. Και τα δύο περιλαμβάνουν μάθηση με λίγα δεδομένα και γρήγορη προσαρμογή σε μεταβολές του περιβάλλοντος. Συγκεκριμένα, στο πρώτο δε μας δίνονται καθόλου *υποστηρικτικά* δεδομένα, δηλαδή παραδείγματα των κατηγοριών στα οποία μπορούμε να βασίσουμε την ταξινόμηση μας, επονομαζόμενα και support set, για τις κατηγορίες τις οποίες τελικά θα χρειαστεί να ταξινομήσουμε και η μόνη πηγή πληροφορίας για τις κατηγορίες αυτές είναι μία βοηθητική περιγραφή ανά κατηγορία. Στο δεύτερο πλαίσιο, μας δίνεται ένα *μικρό* σύνολο με υποστηρικτικά δεδομένα από τις κατηγορίες τις οποίες πρέπει να ταξινομήσουμε. Σε αυτή τη δουλειά, ασχολούμαστε με το πρόβλημα του Zero-shot Learning, αλλά χρησιμοποιούμε υποβοηθητικά το πρόβλημα του Few-shot Learning.

0.1.1 Σχετική Βιβλιογραφία

Αρχικές προσεγγίσεις [36, 37] για την επίλυση του Zero-shot Learning αντιμετώπισαν το πρόβλημα είτε μαθαίνοντας να προβλέπουν την περιγραφή μιας εικόνας από την ίδια την εικόνα, είτε άμεσα είτε έμμεσα έχοντας πρώτα υπολογίσει την πιθανότητα η εικόνα να ανήκει σε κάθε μία από τις κατηγορίες της εκπαίδευσης και χρησιμοποιώντας τις για να υπολογίσουν την περιγραφή. Η τελική ταξινόμηση γίνεται βρίσκοντας την πιο κοντινή από τις δεδομένες περιγραφές.

Σε ύστερες δουλειές, έγινε προσπαθεία να υπολογιστούν σκορ συμβατότητας μεταξύ μιας εικόνας και μιας περιγραφής. Παραδείγματα τέτοιων δουλειών που χρησιμοποιούν γραμμι-

κές συναρτήσεις συμβατότητας είναι οι [38, 39, 40, 41, 42, 43]. Λιγότερες δουλειές χρησιμοποιούν μη-γραμμικές συναρτήσεις, οι οποίες απλά χρησιμοποιούν πολλαπλές γραμμικές τέτοιες συναρτήσεις, π.χ. [15].

Ακόμα μια κλάση τεχνικών είναι οι προσεγγίσεις δημιουργίας πρωτοτύπων. Σε αυτές τις προσεγγίσεις, προσοχή δίνεται στη δημιουργία ενός πρωτότυπου για κάθε κατηγορία, τέτοιο ώστε να μπορούν να συγκριθούν, με κάποιον τρόπο, οι πραγματικές εικόνες με αυτά τα πρωτότυπα. Αυτό μπορεί να επιτευχθεί υπολογίζοντας κατευθείαν τα πρωτότυπα στο χώρο που βρίσκονται οι εικόνες (συνήθως χρησιμοποιούμε εξαγώμενα χαρακτηριστικά των εικόνων αντί για εικόνες), ή υπολογίζοντας το πρωτότυπο σε κάποιον ενδιάμεσο χώρο, στον οποίο μετά προβάλλουμε τις εικόνες. Χαρακτηριστικές τέτοιες προσεγγίσεις είναι οι [44, 45].

Παραγωγικές Προσεγγίσεις

Η εξέλιξη όλων αυτών των προσεγγίσεων έχει οδηγήσει στις προσεγγίσεις που πλέον δίνουν τις καλύτερες επιδόσεις, τις Παραγωγικές Προσεγγίσεις. Ο βασικός αλγόριθμος που ακολουθούν, ο οποίος παρουσιάστηκε στα [18, 11], είναι:

- Στάδιο 1 (Στάδιο Εκπαίδευσης Παραγωγικού δικτύου): Ένα Παραγωγικό δίκτυο εκπαιδεύεται με ένα ανάλογο πλαίσιο (π.χ. Generative Adversarial Network) για να παράγει στιγμιότυπα των κατηγοριών στις οποίες έχουμε πρόσβαση την ώρα της εκπαίδευσης, με είσοδο τις περιγραφές τους. Ένας προεκπαιδευμένος γραμμικός ταξινομητής μπορεί να χρησιμοποιηθεί για να προσθέσει ένα σφάλμα ταξινόμησης στην εκπαίδευση.

- Στάδιο 2 (Στάδιο Εκπαίδευσης Ταξινομητή): Με δεδομένες τις περιγραφές των κατηγοριών που θα αντιμετωπίσουμε την ώρα της αποτίμησης, το Παραγωγικό δίκτυο χρησιμοποιείται για να παράξει ένα συνθετικό σύνολο δεδομένων πάνω στο οποίο εκπαιδεύουμε ένα απλό ταξινομητή (π.χ. γραμμικό, SVM).

- Στάδιο 3 (Στάδιο Αποτίμησης): Ταξινομούμε τις πραγματικές εικόνες αποτίμησης με τον ταξινομητή του προηγούμενου σταδίου.

Τέτοιος αλγόριθμος είναι και ο [16]. Το βασικό αυτό πλάνο μπορεί να εμπλουτιστεί με έξτρα σφάλματα για να οδηγήσουν καλύτερα την εκπαίδευση, π.χ. [46, 13], επιπλέον δίκτυα [12], ή διαφορετικές μορφές εκπαίδευσης [47].

0.2 Μέθοδος

Σε αυτή τη δουλειά, χτίσαμε πάνω σε αυτόν τον αλγόριθμο, τον κορμό των Παραγωγικών Προσεγγίσεων, και τον βελτιώσαμε χρησιμοποιώντας ένα ταξινομητή Few-shot Learning, στον οποίο ανάγουμε τις ταξινόμησεις και την ώρα της εκπαίδευσης και την ώρα της αποτίμησης. Συγκεκριμένα, ο αλγόριθμος που παρουσιάσαμε παραπάνω μεταβάλλεται ως εξής:

- Στάδιο Εκπαίδευσης Παραγωγικού Δικτύου: Χρησιμοποιούμε ένα ταξινομητή Few-shot Learning σαν τον ταξινομητή από τον οποίο παίρνουμε σφάλμα ταξινόμησης για το Παραγωγικό δίκτυο. Ο ταξινομητής αυτός είναι προεκπαιδευμένος και προαιρετικά μπορούμε να τον προσαρμόσουμε περαιτέρω.

- Στάδιο Εκπαίδευσης Ταξινομητή: Μπορούμε, προαιρετικά, να προσαρμόσουμε τον ταξινομητή Few-shot Learning του 1ου σταδίου χρησιμοποιώντας το Παραγωγικό δίκτυο για να πάρουμε και τα υποστηρικτά δεδομένα και τα δεδομένα αποτίμησης με βάση τις περιγραφές των κλάσεων του ελέγχου.

- Στάδιο Αποτίμησης: Χρησιμοποιούμε τον προαναφερθεντα ταξινομητή Few-shot Learning. Τα υποστηρικτικά δεδομένα παράγονται από το Παραγωγικό Δίκτυο.

Η χρήση πρακτικά του ίδιου ταξινομητή σε όλα τα στάδια μπορεί πλέον να γίνει γιατί οι ταξινομητές αυτοί συνήθως ταξινομούν ταιριάζοντας υποστηρικτικά δεδομένα με αυτά του ελέγχου και η εκπαίδευση του, και μιας και αποτελεί ταξινομητής αυτοτελούς προβλήματος, μπορεί να εκπαιδευτεί σε πραγματικά δεδομένα και δεν είναι εξαρτημένος από την εκπαίδευση του Παραγωγικού δικτύου.

Η επιλογή χρησιμοποίησης του, όμως, έγινε ώστε να δώσουμε στο Παραγωγικό δίκτυο πρόσβαση, την ώρα της εκπαίδευσης του, στον ταξινομητή που θα χρησιμοποιηθεί και την ώρα του ελέγχου, ο ταξινομητής μπορεί να εκπαιδευτεί σε πραγματικά δεδομένα που αναμένεται να βοηθήσει στη γενίκευση και το Few-shot Learning είναι διαισθητικά πιο κοντά στο Zero-shot Learning.

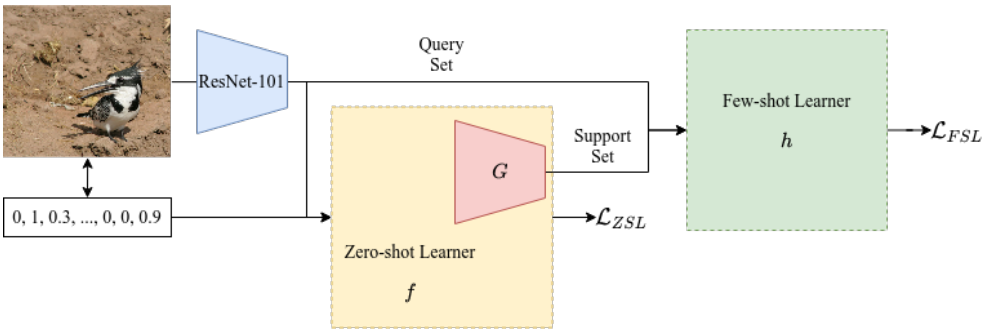
Επιπλέον, να σημειώσουμε πως η μέθοδος μας είναι Plug and Play, δηλαδή μπορεί να χρησιμοποιηθεί οποισδήποτε αλγόριθμος Zero-shot Learning, αρκεί να ακολουθεί το σκελετό που παρουσιάσαμε παραπάνω, και οποιοσδήποτε αλγόριθμος Few-shot Learning, αρκεί να ταξινομεί ταιριάζοντας support με query και η διαδικασία που το κάνει να είναι διαφορίσιμη, που είναι εύλογες υποθέσεις.

Η σύγκριση του προηγούμενου αλγορίθμου με τον προτεινόμενο φαίνεται συγκεντρωτικά στον Πίνακα 1.

Το τελικό σφάλμα που καλείται να ελαχιστοποιήσει το Παραγωγικό δίκτυο ορίζεται ως

$$\mathcal{L}_{Z2FSL} = \mathcal{L}_{ZSL} + \gamma \mathcal{L}_{FSL} \tag{1}$$

όπου γ είναι υπερπαράμετρος, $\mathcal{L}_{ZSL}$ το σφάλμα του αλγορίθμου για Zero-shot Learning και το $\mathcal{L}_{FSL}$ το σφάλμα του ταξινομητή Few-shot Learning με υποστηρικτικά δεδομένα από το Παραγωγικό δίκτυο. Το πλαίσιο φαίνεται και στο Σχήμα 3.



Σχήμα 3: *Γραφική αναπαράσταση του πλαίσιου που προτείνουμε.*

Στάδιο	**Βασική Παραγωγικά Προσέγγιση**	**Προτεινόμενη Παραγωγική Προσέγγιση**
Στάδιο Εκπαίδευσης Παραγωγικού δικτύου	Εκπαίδευση Παραγωγικού δικτύου, προαιρετική χρήση γραμμικού ταξινομητή.	Εκπαίδευση Παραγωγικού δικτύου, χρήση ταξινομητή Few-shot Learning για σφάλμα ταξινόμησης. Προαιρετική προσαρμογή ταξινομητή.
Στάδιο Εκπαίδευσης Ταξινομητή	Εκπαίδευση απλού ταξινομητή με βάση τα συνθετικά στιγμιότυπα του Παραγωγικού δικτύου.	Προαιρετική επανεκπαίδευση ταξινομητή Few-shot learning σε συνθετικά στιγμιότυπα άγνωστων κατηγοριών.
Στάδιο Αποτίμησης	Ταξινόμηση εικόνων αποτίμησης με απλό ταξινομητή.	Ταξινόμηση εικόνων αποτίμησης με ταξινομητή Few-shot Learning. Το Παραγωγικό δίκτυο δίνει τα υποστηρικτικά δεδομένα.

Πίνακας 1: *Σύγκριση μεταξύ βασικής Παραγωγικής Προσέγγισης και προτεινόμενης μεθόδου.*

0.3 Αποτελέσματα

0.3.1 Διαισθητικός Ορισμός Προβλημάτων

Zero-shot Learning: Εδώ μας δίνονται κλάσεις με τις εικόνες τους για εκπαίδευση, μαζί με τις περιγραφές τους. Στόχος μας είναι να βρούμε πως τα χαρακτηριστικά των εικόνων απορρέουν από τις περιγραφές. Την ώρα της αποτίμησης, μας δίνονται στιγμιότυπα από κλάσεις τις οποίες δεν είχαμε πρόσβαση την ώρα της εκπαίδευσης, μαζί με τις περιγραφές τους, κι ουσιαστικά καλούμαστε να ταιριάξουμε τις εικόνες με τις περιγραφές τους για να τις κατηγοριοποιήσουμε.

Generalized Zero-shot Learning: Αποτελεί γενίκευση του Zero-shot Learning. Ουσιαστικά παρακρατάμε κάποιες από τις εικόνες την ώρα της εκπαίδευσης για να τις χρησιμοποιήσουμε την ώρα του ελέγχου, δηλαδή ελεγχόμαστε στο πόσο καλά ταξινομούμε και τις κλάσεις της εκπαίδευσης και τις άγνωστες κλάσεις, πάντα με δεδομένες όλες τις περιγραφές. Εδώ υπάρχει το ζήτημα της προκατάληψης υπέρ των κλάσεων της εκπαίδευσης.

Οι μετρικές που χρησιμοποιούνται είναι, για το Zero-shot Learning, η μεσή ανά κατηγορία ακρίβεια, και για το Generalized Zero-shot Learning, είναι ο αρμονικός μέσος της μέσης ανά κατηγορία ακρίβειας για τις άγνωστες κατηγορίες και τις κατηγορίες της εκπαίδευσης.

0.3.2 Dataset

Χρησιμοποιούμε το Caltech UCSD Bird (CUB) [19], που περιέχει 11788 εικόνες από 200 κλάσεις και οι περιγραφές των κατηγοριών έχουν μορφή 312-διάστατου διανύσματος. Χρησιμοποιούμε επίσης το Animals with Attributes 2 (AwA2) [14], με 37322 εικόνες 50 κλάσεων και 85-διάστατες περιγραφές, και, τέλος, το SUN Scene Classification (SUN), με 14340 εικόνες 717 κλάσεων με 102-διάστατες περιγραφές.

Αντί για εικόνες, χρησιμοποιούμε αναπαραστάσεις που μας δίνει το προεκπαιδευμένο στο ImageNet [48] ResNet-101 [49] στο προτελευταίο επίπεδο του, που είναι 2048-διάστατα διανύσματα. Κάνουμε L2-normalization στις περιγραφές και χρησιμοποιούμε αυτές με τις συνεχής τιμές αντί για αυτές με τις δυαδικές. Κάνουμε επαύξηση δεδομένων χρησιμοποι-

	Zero-shot Learning			**Generalized Zero-shot Learning**								
	CUB	AwA2	SUN	CUB			AwA2			SUN		
Μέθοδος	**aT1**	**aT1**	**aT1**	**u**	**s**	**H**	**u**	**s**	**H**	**u**	**s**	**H**
CONSE [50]	34.3	44.5	38.8	1.6	72.2	3.1	0.5	90.6	1.0	6.8	39.9	11.6
SAE [43]	33.3	54.1	40.3	7.8	54.0	13.6	1.1	82.2	2.2	8.8	18.0	11.8
LATEM [15]	49.3	55.8	55.3	15.2	57.3	24.0	11.5	77.3	20.0	14.7	28.8	19.5
SJE [40]	53.9	61.9	53.7	23.5	59.2	33.6	8.0	73.9	14.4	14.7	30.5	19.8
ESZSL [41]	53.9	58.6	54.5	12.8	63.8	21.0	5.9	77.8	11.0	11.0	27.9	15.8
SYNC [44]	55.6	46.6	56.3	11.5	70.9	19.8	10.0	90.5	18.0	7.9	43.3	13.4
DEVISE [39]	52.0	59.7	56.5	23.8	53.0	32.8	17.1	74.7	27.8	16.9	27.4	20.9
ALE [38]	54.9	62.5	58.1	23.7	62.8	34.4	14.0	81.8	23.9	21.8	33.1	26.3
CVCZSL [45]	54.4	71.1	62.6	47.4	47.6	47.5	56.4	81.4	**66.7**	36.3	42.8	39.3
f-CLSWGAN [18]	57.3	-	60.8	43.7	57.7	49.7	-	-	-	42.6	36.6	39.4
LisGAN [46]	58.8	-	61.7	46.5	57.9	51.6	-	-	-	42.9	37.8	40.2
f-VAEGAN [16]	61.0	-	64.7	**48.4**	60.1	**53.6**	-	-	-	**45.1**	38.0	41.3
GDAN [12]	-	-	-	39.3	**66.7**	49.5	32.1	67.5	43.5	38.1	**89.9**	**53.4**
OCD [51]	60.3	**71.3**	63.5	44.8	59.9	51.3	**59.5**	73.4	65.7	44.8	42.9	43.8
Z2FSL(VAEGAN, PN)	**62.0**	63.2	**66.5**	42.6	58.7	49.4	50.5	**81.8**	62.4	42.1	33.3	37.2

Πίνακας 2: *Σύγκριση της μεθόδου μας (τελευταία στήλη, με αλγόριθμο Zero-shot Learning του [18] και Few-shot Learning του [17]) με άλλες ανταγωνιστικές μεθόδους. Το **aT1** συμβολίζει τη μέση ανα κατηγορία ακρίβεια, το **u** τη μέση ανά κατηγορία ακρίβεια των άγνωστων κατηγοριών, το **s** την ίδια μετρική για τις κατηγορίες της εκπαίδευσης και το **H** είναι ο αρμονικός μέσος των δύο τελευταίων. Για state-of-the-art σε Generalized Zero-shot Learning εξετάζουμε μόνο τις μεθόδους που πετυχαίνουν καλό **u**, που φαίνονται κάτων από την μπάρα.*

ώντας 10 κομμάτια των εικόνων αντί για απλά την ίδια την εικόνα (1 από αυτά παραμένει όμως η αρχική εικόνα, την οποία και χρησιμοποιούμε την ώρα της αποτίμησης). Χρησιμοποιούμε τις διαιρέσεις εκπαίδευσης-αποτίμησης που δίνονται στο [14].

0.3.3 Σύγκριση με State-of-the-art

Παρουσιάζουμε τα αποτελέσματα μας σε σύγκριση με άλλους ανταγωνιστικούς αλγορίθμους στον Πίνακα 2. Βλέπουμε ότι πετυχαίνουμε καλύτερα αποτελέσματα σε 2 από τα 6 πλαίσια και το βέλτιστο **s** στο AwA2. Εξετάζουμε περαιτέρω τα αποτελεσματά μας με επιπλέον πειράματα.

0.3.4 Σύγκριση Βασικού με Προτεινόμενο

Στους πίνακες 3 και 4 βλέπουμε ότι η μέθοδος μας πράγματι βελτιώνει τα αποτελέσματα του βασικού Παραγωγικού αλγορίθμου, ενώ εντοπίζουμε πως η αύξηση αυτή οφείλεται και στη χρήση του καινούργιου ταξινομητή στην εκπαίδευση, αλλά και στην αποτίμηση, χωρίς σχεδόν να χάνουμε καθόλου από τα ξεχωριστά κέρδη όταν αυτός χρησιμοποιείται μόνο σε ένα από αυτά τα στάδια. Παρατηρούμε επίσης ότι τα αποτελέσματα από την υλοποίηση μας είναι χαμηλότερα του αναμενόμενου, που δικαιολογεί σε μεγάλο βαθμό την κάτω των προσδοκιών απόδοση στο Zero-shot Learning.

	SUN
Μέθοδος	**aT1**
f-VAEGAN [16] (original)	64.7
f-VAEGAN [16] (our impl.)	60.1
f-VAEGAN [16] (our impl., w/o aug.)	61.1
Z2FSL(f-VAEGAN, PN) (w/ softmax)	62.8
Z2FSL(f-VAEGAN, PN) ($\gamma = 0$, no finetuning)	64.3
Z2FSL(f-VAEGAN, PN)	66.5

Πίνακας 3: *Σύγκριση μεταξύ του βασικού αλγορίθμου (αναφερόμενα στα [16] και δικά μας αποτελέσματα) και του δικού μας αλγορίθμου (με αλγόριθμο Zero-shot Learning τον [16] και Few-shot Learning τον [17]) όταν χρησιμοποιείται όπως τον ορίσαμε και όταν ο ταξινομητής χρησιμοποιείται μόνο την ώρα της εκπαίδευσης ή μόνο την ώρα της αποτίμησης.*

	ΎΥΒ	ΑωΑ2	ΣΥΝ
Μέθοδος	**aT1**	**aT1**	**aT1**
f-VAEGAN [16] (original)	61.0	-	64.7
f-VAEGAN [16] (our impl.)	53.4	61.6	60.1
Z2FSL(f-VAEGAN, PN)	62.0	63.2	66.5
Absolute Performance Gain (original)	+1.0	-	+1.8
Relative Performance Gain (original)	+1.6	-	+2.8
Absolute Performance Gain (our impl.)	+8.6	+1.6	+6.4
Relative Performance Gain (our impl.)	+16.1	+2.6	+10.6

Πίνακας 4: *Σύγκριση της μεθόδου μας (με αλγόριθμο Zero-shot Learning του [18] και Few-shot Learning του [17]) με τα αποτελέσματα του f-VAEGAN [16] όπως αναφέρονται στη δημοσίευση και δική μας υλοποίηση.*

0.4 Συμπεράσματα

Πρώτο το κύριο, παρουσιάζουμε ένα πλαίσιο για Zero-shot Learning το οποίο εκμεταλλεύεται τον ίδιο ταξινομητή και την ώρα της εκπαίδευσης και την ώρα της αποτίμησης. Δεύτερον, εφόσον χρησιμοποιούμε ταξινομητή Few-shot Learning, το πλαίσιο παντρεύει δύο προβλήματα με έλλειψη δεδομένων, το Zero-shot Learning και το Few-shot Learning, και το οποίο μπορεί να χρησιμοποιηθεί άμεσα σε κάποιο περιβάλλον που μας δίνονται και υποστηρικτικά δεδομένα ή/και βοηθητικές περιγραφές των κατηγοριών. Τρίτον, κάνουμε ένα βήμα πιο κοντά στη Βαθιά Μάθηση με την αντικατάσταση απλών ταξινομητών με τον ταξινομητή για Few-shot Learning (π.χ. για το AwA2 στο Zero-shot Learning, το μοντέλο που χρησιμοποιήσαμε έχει 410 φορές περίπου περισσότερες εκπαιδεύσιμες παραμέτρους συγκριτικά με έναν απλό γραμμικό ταξινομητή). Τέταρτον, παρουσιάζουμε ένα απλό Plug and Play πλαίσιο για την άμεση βελτίωση των αποτελεσμάτων ενός αλγορίθμου για Zero-shot Learning. Πέμπτον, δείξαμε ότι οι αλλαγές που κάναμε κατά την εκπαίδευση και κατά τον έλεγχο είναι ουσιώδεις.

Chapter 1

Introduction

1.1 Artificial Intelligence

Artificial Intelligence (AI), also referred to as *machine* or *computational* intelligence, describes intelligence demonstrated by (inorganic) machines. The term "Artificial" is used to contrast AI with what is described as *natural intelligence*, displayed by humans and animals alike. The field of AI is broadly described as the study of "intelligent agents" (similarly: "rational agents"): any device that perceives its environment and takes actions to maximize its chance of successfully achieving its goals [52]. AI is also widely considered as the attempt to mimic cognitive functions associated with the human mind, and has drawn inspiration from the human brain.

The traditional challenges of AI research, formally initiated in 1955 as an academic discipline, include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects [52]. However, as pivotal advances are made in all these fields, *General Intelligence* has become the overarching goal, where a machine is considered intelligent if it can successfully and practically adapt to novel, incongruous tasks.

1.1.1 Brief History

AI first engendered interest on the assumption that human intelligence can be completely simulated by a machine, namely a computer, when computers started to become more widespread. Nonetheless, ethical and philosophical issues that arise from bestowing intelligence to otherwise inanimate machines have been explored through myths and fiction since antiquity. One such machine is the bronze giant *Talos* (Greek: Τάλως), depicted in an ancient Greek coin in Figure 1.1, which (or who; the relative pronoun appropriate for such a machine is among, or rather a corollary of, the ethical issues of AI) was forged by the gods to protect its (or his/her) land and its master. A more recent example is the famous Mary Shelley's *Frankenstein*. The characters' fates raised and continue to raise many of the issues now discussed formally in the ethics of artificial intelligence.

Alan Turing's theory of computation, which suggests that a machine manipulating 0s and 1s can simulate any conceivable act of mathematical deduction and its "corollary", the Church-Turing thesis, which suggests that digital computers can simulate any process of formal reasoning, laid the foundation of AI. Simultaneous advances in the disciplines

of neurobiology, information theory and cybernetics essentially kick-started the research on digital brains. Moreover, the question of whether a machine can be intelligent was substituted by the one of plausibility of intelligent behavior judged by us, humans, as, for the time being, we do not have any way to discern between the two.

The initial estimations of the difficulty, the sophistication and the complexity of the task were greatly underestimated. The unexpected slow progress resulted multiple times in retrieval of funds. However, commercial successes and the exponential increase in computational power (i.e. Moore's law) allowed for research to progress, which culminated into further integration in the business world and other important landmarks, like the victory over the world chess champion, Garry Kasparov, in 1997, by the computer named "Deep Blue".

Figure 1.1: *The mythical automaton Talos depicted in an ancient Greek coin from Crete. Source: [2].*

1.2 Machine Learning

Machine Learning (ML) is a class of algorithms and techniques that are a subset of AI, as seen in Figure 1.2. A ML algorithm is an algorithm that is able to learn through data. This can be succinctly described formally as "A computer program is said to learn from experience E w.r.t. some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E". T, E and P are left vague because they can be defined in various manners. For examples, tasks T may include classification, language translation, etc. Performance P can be classification accuracy and translation accuracy respectively. Regarding experience E, ML algorithms can be said to belong in two broad classes, **supervised** and *unsupervised*, depending on the experience they are allowed to observe. The former usually receives (training) datasets along with labels or targets for each sample, while the latter is most likely only provided

with a dataset and tasked with extraction of useful properties [53].

1.2.1 Supervised Learning

Supervised Learning is the ML task of learning a function that maps an input to an output based on example input-output pairs, labeled training data that is. In supervised learning, each input datapoint is accompanied by its desired output value. Ideally, the learning algorithm will learn to extrapolate to new datapoints, i.e. be able to generalize, and correctly predict their labels.

A plethora of ML algorithms are available. Because of the "No free lunch" theorem, we know, in some sense, that no algorithm can be said to be better universally than another [54]. Four major considerations are to be made when selecting an algorithm, two of which are integral to understanding the problem we are trying to address.

Bias-Variance trade-off

The bias of a point predictor (alternatively, *estimator*) g of a variable ϑ, $\hat{\vartheta}_m = g(x^{(1)}, \ldots, x^{(m)})$, can be defined as $\text{bias}(\hat{\vartheta}_m) = \mathbb{E}(\hat{\vartheta}_m) - \vartheta$, with the expectation being over the data. An unbiased estimator has $\text{bias}(\hat{\vartheta}_m) = 0$, which can also be asymptotic. The variance of the estimator is simply the variance $V(\hat{\vartheta})$ where the random variable is the training set. This indicates the deviation of the prediction when independently resampling the dataset. Now, if we measure the mean square error of the predictions, we get $MSE = \text{bias}(\hat{\vartheta}_m)^2 + V(\hat{\vartheta}_m)$. Predictors must achieve a low bias whilst retaining a low variance. The relationship between bias and variance is linked with capacity, underfitting and overfitting. Underfitting describes the situation when the model is not able to achieve a sufficiently small error on the training data. Overfitting occurs, on the other hand, when the gap between the training error and the error on unseen samples is large. Informally, a model's capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit training data, models with high capacity can overfit by memorizing properties of the training set that do not benefit generalization. Usually, increasing capacity leads to an increase in variance and a reduction of bias.

Function complexity and cardinality of training set

If a function is simple, even an inflexible classifier with high bias will be able to learn it using a small training set. However, when the function models complex interactions between its inputs and behaves differently in different areas of its input space, then we need a flexible algorithm, with low bias but high variance, and a lot of examples.

Dimensionality of the input space

The dimensionality of the input space is integral because of both the combinatorial explosion and the potential irrelevance (or small correlation) of many values in the input vectors to the output value, which essentially constitute noise.

Noise in the output values

Noise in the desired output of the training examples, either due to measurement or human error, can lead to overfitting.

We further elaborate on the bias-variance trade-off and the cardinality of the dataset, the two issues we are trying to alleviate, in the next two sections while discussing **Deep Learning** (DL) and the setting devised to test generalization given a bigger shift in the data structure of the test data compared to normal Supervised ML, **Zero-shot Learning** (ZSL).

1.2.2 Big Data & Deep Learning

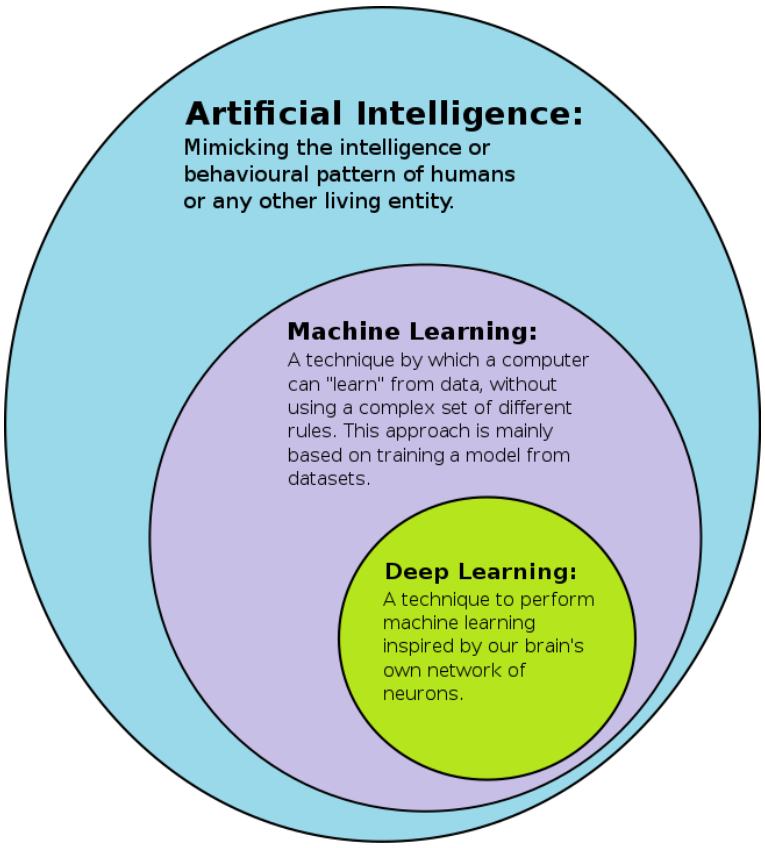


Figure 1.2: *Abstract Venn diagram of AI, ML and DL algorithms. Source: [3].*

DL differs from other ML concepts in the sense that it has allowed an understanding of the world in terms of a hierarchy of concepts, with each concept defined through its connections to simpler ones. Similarly to ML, this approach circumvents the need for humans to formally specify all the knowledge the computer needs. If we draw a directed graph to represent the hierarchy of knowledge, we end up with a deep graph who displays more intricate representations the deeper we are in a path, like Figure 1.3. This is why this family of approaches are called *Deep* Learning. DL algorithms usually include deep networks, which is also why the age of DL is accompanied by the age of *Big Data*. Datasets

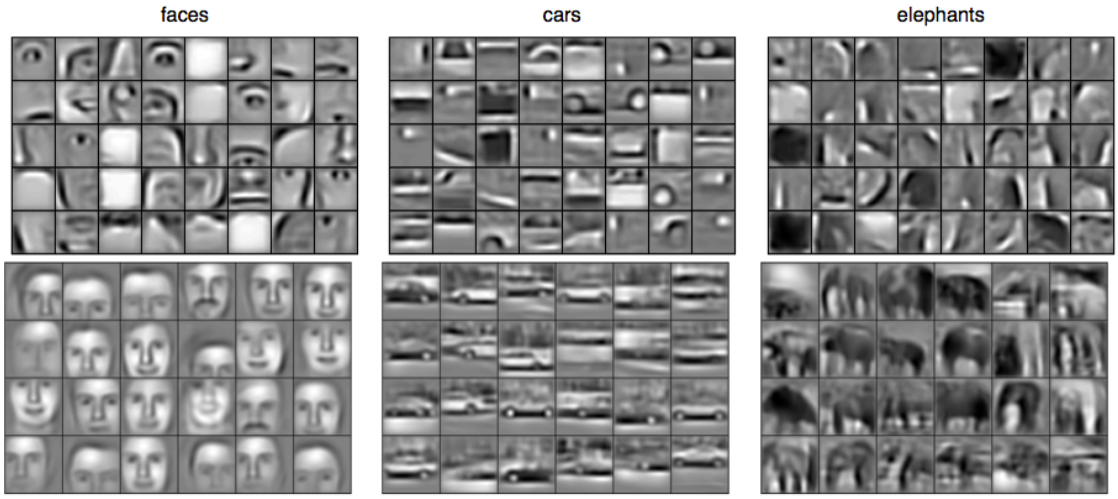


Figure 1.3: *Graphical representation of representations learned from deep NNs. Top row displays representations in the initial layers (low-level representations), while bottom row displays representations in the last layers (high-level representations).*

have become several Gigabytes long and companies that operate in the AI space or strive for further automation and improvement of their operations continuously build immense private datasets to train such algorithms. This has led user data and personal information to become a sort of currency, with companies allowing free usage of their services and applications in return for data, and a point of controversy, with authorities introducing policies[1] to restrict their acquisition and inform the public about their rights.

This abundance of data has promptly resulted in superb performance by DL algorithms. DL has seen great success in various settings, like CV, by surpassing the performance of classical algorithms [21], by reconstructing images to 3D environments [22], landscapes to 3D elevation models [30] and video to 3D [23]. Another success story is NLP with very deep models that can produce even coherent articles and almost perfectly understand human language [24, 25] and Computer Graphics [26, 27]. In addition, a plethora of different disciplines deploy DL algorithms, such as Neuroscience [28, 29] or Remote Sensing [30, 31], and it has achieved superhuman performance in many narrow tasks, e.g. [33, 34].

Nonetheless, not all applications can afford to collect sufficient data to utilize *existing* DL algorithms and refrain from using them owning to overfitting. The phenomenon can be succinctly summarized by the graph in Figure 1.4, where it becomes obvious that DL algorithms, deep NN that is, actually underperform in low data regimes, despite their superhuman performance on gigantic datasets. Combined with the demand of DL for resources, whose environmental toll is only beginning to be scrutinized, with initial results revealing a disheartening reality [35], DL techniques become a worthwhile endeavor only

[1] A pertinent example of the latter would have to be the General Data Protection Regulation (GDPR), which is a regulation on data protection and privacy in the European Union. It also addresses the transfer of personal data outside it. The GDPR's principal aim is to allow individuals to control their personal data and to simplify the regulatory environment for international business by consolidating the regulation within the European Union.

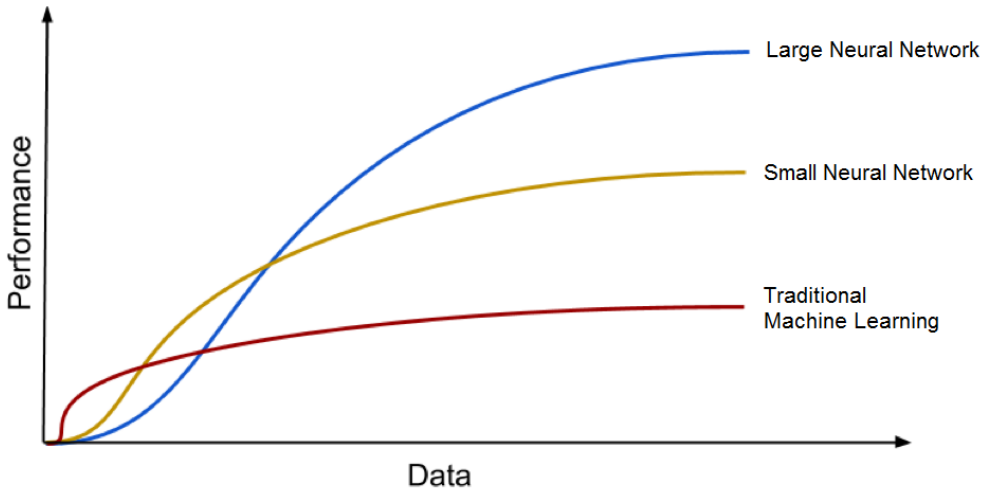


Figure 1.4: *Performance comparison w.r.t. the amount of available data. Source: [1].*

when large datasets are available.

1.2.3 Zero-shot Learning

As a means to study the performance of CV systems, researchers devised tasks that include small or medium datasets and substantial shifts in the distribution of the data between training and testing, so as to "attack" DL techniques from multiple angles and essentially induce overfitting in naive approaches. One such task is ZSL, where the current, at least, benchmarks involve small datasets, and the learner is tasked with classifying images from classes that the algorithm has not seen during training and without any supporting data! It can be easily discerned why tuning the system to training noise will cause a huge degradation in performance. Of course, to ease the task, a description of the "quintessential" entity/object of each category is given, meaning that one description per novel class is provided, and the task, thereafter, becomes to match the description of the class to its images.

Theses descriptions have to provide important, high-level information to the learner about the traits of a category in order to facilitate the training and, ultimately, to make the problem worthy of scrutiny. The learner can be thought of as being tasked to figure the causal relation between the high-level semantic descriptions and the low-level pixel information in order to extrapolate to novel classes. An additional difficulty is that the descriptions are provided without the exact semantics becoming available to the learner, and therefore the system does not solely have to learn how these descriptions translate to entities, but also the exact semantics of the description, or else is in peril of overfitting due to the co-occurrence of descriptive atoms resulting in misinterpretation of their semantics, etc. An enlightening example is the following. Suppose we have to categorize species of birds, and it just so happens that every bird in the training set with a specific body color has the same wing color. Then it becomes *impossible* for the learner to extrapolate to species with different colors of body and wings, as a bird with blue body and red wings

might as well be the exact same to the system as one with the colors reversed!

Naturally, it is worth pondering whether a human, ignorant to bird species and their characteristics, could make that distinction. The answer in this edge case is also negative, but humans, with all their peripheral knowledge, i.e. what a wing and what a body is, which, remember, is not part of the input to the algorithm, will be able to imagine non-existent species of bird after seeing just a single specimen of any species. As a consequence, even though humans are able to deal with unseen classes and learn from few examples, we have to acknowledge that this is feasible in large part due to their overall knowledge of the world. For this reason, generalizing to unseen categories becomes even harder when only using the few available training data.

Having established the difficulty of the task, we discuss the implications for DL algorithms on the task. Currently, the only DL algorithm being used is for the extraction of high-level features/representations from the images using a *very deep* extractor trained on a large image dataset. This can indeed bring knowledge not available in the actual training set to the table. Afterwards, modern techniques deploy shallow NNs and linear classifiers or traditional supervised classifiers, like Support Vector Machines (which belong to either the yellow or the red curve in Figure 1.4). Even though the progress and the effort are commendable, DL has failed to be incorporated further in the solution of ZSL.

1.2.4 Few-shot Learning

The task of generalizing to unseen categories becomes *somewhat* easier if some instances of each novel class are provided. That is to say, instead of a description of a category we have to match its instance to, *supporting examples* are provided. That is the basic premise of Few-shot Learning (FSL). We can use these examples to find the basic characteristics displayed by each category and use these characteristics to differentiate between the categories we are tasked with classifying. Notice that such tasks are easier to devise (for research purposes) because all we need are labeled examples and no extra data, and therefore datasets are somewhat larger and richer compared to the ZSL ones. This usually allows for deeper networks.

FSL is not only intuitively easier for computer algorithms, it is of course easier for humans as well. Humans display superb performance in such tasks when the supporting examples are few (e.g. one or two). Their performance declines steeply when the classes they have to differentiate between become numerous w.r.t. their working memory, e.g. even 15-20 different categories [55]. These experiments do not test, however, the ability of humans to classify when the supporting examples are readily available for reference, making the task with that many classes tedious and demotivating.

FSL and ZSL are related in that generalization is required to unseen classes, and whatever supporting data (supporting examples and descriptions respectively) are provided are usually not sufficient for or compatible to standard approaches to supervised classification tasks.

1.3 Contributions

In this work, we introduce a novel and effective yet simple framework for Zero-shot Learning that utilizes the same classifier during training and testing. Our framework couples an existing Zero-shot learner that utilizes a generative approach with also an existing Few-shot learner to improve upon the results of the plain Zero-shot learner. In contrast to contemporary approaches, we are able to make use of the the Few-shot learner both during training and testing. Therefore, our contribution are:

- We present a Zero-shot-learner agnostic framework, which we experimentally show increases the performance of the Zero-shot learner without any modifications to the framework itself.
- We demonstrate that a simple pretrained Few-shot learner is better that a linear or a simple non-parametric classifier in classifying the test examples based on synthesized ones, even if the Few-shot learner is not utilized in any way during training. In effect, we empirically demonstrate that the Few-shot Learning setting is more appropriate than a standard classification setting.
- We also show that the Few-shot learner results in performance boosts even when only used during training (can be though of as a regularizer).
- We illustrate that the gains of using the Few-shot learner only during training and only during testing are almost orthogonal, providing a gain almost equal to their sum when used in both settings.
- We demonstrate that the Few-shot learner on its own is enough to constitute a competitive Zero-shot learning framework, by guiding a generator only through a classification loss, achieving state-of-the-art results if we exclude the framework introduced in this work.
- We link together two different low-data regimes, Few-shot Learning and Zero-shot Learning.

1.4 Thesis Outline

The organization of this thesis is as follows. In Chapter 2, we introduce many notions and notations that make reading the rest of the thesis feasible. Namely, we discuss basic Machine Learning concepts in Section 2.1, along with introducing Neural Networks architectures that we use or discuss. Later, in Section 2.2 we discuss maybe the most important concept in this work, Generative Networks. To end the Section, we qualitatively introduce the classification settings we tackle. Chapter 3 is the essential part of this thesis. We start by introducing the rationale behind Zero-shot Learning and discuss past work. In Section 3.2, we discuss what we believe can be improved in previous work and elaborate on how we try to improve them. Next, in Section 3.3, we formally present the

task we are trying to solve, the necessary background and describe at a medium level our framework. We conclude the Chapter in Section 3.4 by presenting the technical details of our implementation and our results on major benchmarks. Moreover, we present several ablation studies that we believe illustrate the merits of our framework. Finally, we briefly discuss the conclusions we drew from our work in Section 3.5 and more elaborately in Chapter 4, as well as future extensions and philosophical discussion.

Chapter 2

Background

2.1 Machine Learning

Machine Learning (ML) is a subfield of AI. It enables computers to learn from data and even improve themselves without being explicitly programmed. The basic premise of ML is to build algorithms that can receive input data and use statistical analysis to improve themselves on predicting an output. It differs in this regard from other computational approaches within Computer Science, where algorithms are given explicit instructions on how to solve problems and everything has to be accounted for by the programmer. In ML, learning from data results in the algorithms learning by themselves what to account for and how to deal with every hurdle.

In recent years, AI has experienced a resurgence due to a subfield of ML, Deep Learning (DL). DL utilizes copious amounts of data and models with the ability to memorize a plethora of rules and high-level concepts, all encoded in their parameters. DL models achieve state-of-the-art and even superhuman performance on many well-defined settings [33, 34].

ML tasks are generally classified into two broad categories concerning how learning is achieved, namely how feedback is provided to the ML model. The first one, **Supervised Learning**, in which algorithms are trained based on example input and output data, usually labeled by humans. The second and last one is **Unsupervised Learning**, where unlabeled data is available and the task is to extract statistical structure from them. In this thesis, we are concerned with Supervised Learning, and in particular, image classification.

2.1.1 Supervised Learning

In Supervised Learning, the model is provided with example inputs that are coupled with their desired output values. The purpose of an algorithm in this domain is to be able to "learn" by comparing its output to the desired one, locating where the errors occurred within its internal computations and modifying its parameters. Thereafter, the patterns learned in the training data are used to predict the output of unlabeled data.

For instance, a model may be fed data with images of "cats" labels as such, and images of "dogs" likewise. After assimilating this data, the supervised model is tasked with discriminating between unlabeled images of cats and dogs. Notice that, in Figure

2.1, a naive approach will classify the input dog as a cat, given its size and the variation of face characteristics between dog races.

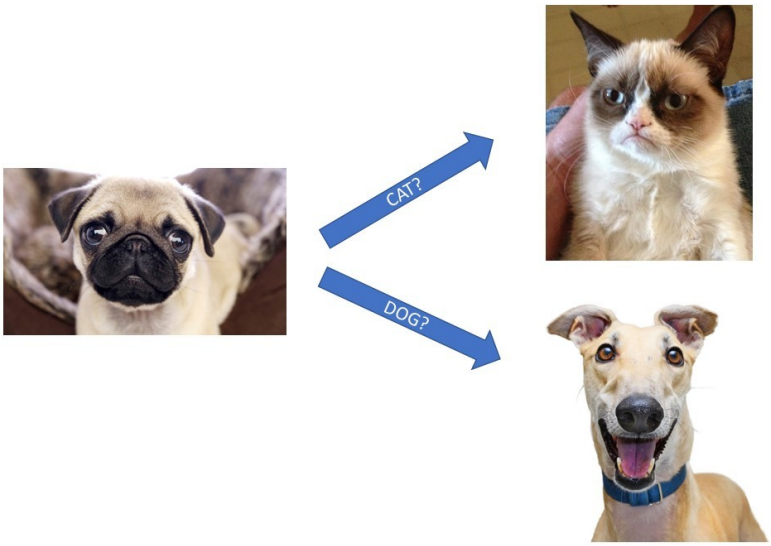


Figure 2.1: *The model is tasked with classifying an unlabeled image of a small dog. It has to do so going from the images it has seen during its training. We can understand that, even though it is cognitively easy for us to make the distinction, when trying to express the our rationale behind our classification we may fail to provide a comprehensive answer. In fact, in this case, a naive classification policy will result in an error.*

A common use case for supervised algorithms is the use of historical data to predict statistically likely events. The task may be stock market prediction, spam filtering or tumor detection.

Formally, given a set of N training examples $\{(x_i, y_i)|i = 1, \ldots, N\}$ such that x_i is the representation of the input (e.g. a digital image) and y_i is its label, a supervised algorithm tries to learn a function $f : X \to Y$, where X is the input space and Y the output space. Usually, the function f belongs to a hypothesis space F. It is, also, convenient to represent f using a compatibility function $g : X \times Y \to \mathbb{R}$ such that f can be defined as returning the label with the maximum score $f(x) = arg\max_y g(x, y)$. There are two basic approaches to choosing f or g: **empirical risk minimization** and **structural risk minimization**. The former seeks the function that best fits the data. The latter includes includes a penalty function that controls the bias-variance trade-off. In both cases, we assume the training set consists of i.i.d. samples.

In order to measure the error of a prediction we define a loss function, $L : Y \times Y \to \mathbb{R}^{\geq 0}$. However, we do not optimize f directly with the loss function. Rather, we define the *risk* (a functional) $R(f)$ of function f as the expected loss. Since the expectation requires knowledge of the true distribution of the data, we approximate that with the histogram of the data and, therefore, the risk with the *empirical risk*

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)). \tag{2.1}$$

We do this instead of iterating over each individual sample because the ultimate goal is to optimize f w.r.t all the data collectively, not individual examples. To calculate R_{emp}, we ussually sample some examples from the training set instead of fetching and processing the whole set at each step. This randomly sampled set is usually called a **batch**.

2.1.2 Perceptron & Feedforward Neural Networks

Perceptron

Let the training set contain samples of only two categories (classes/labels). We will examine linear classification functions, i.e. $f(x; w) = w^T x + w_0$, where x is an input vector, w and w_0 are the coefficients, referred to as *weights*. The classification is achieved by checking the sign of $f(x)$, so one class is predicted for x if $f(x; w) > 0$, the other if $f(x; w) < 0$. This model is called *Perceptron* [56, 57]. Figure 2.2 is emblematic of the process.

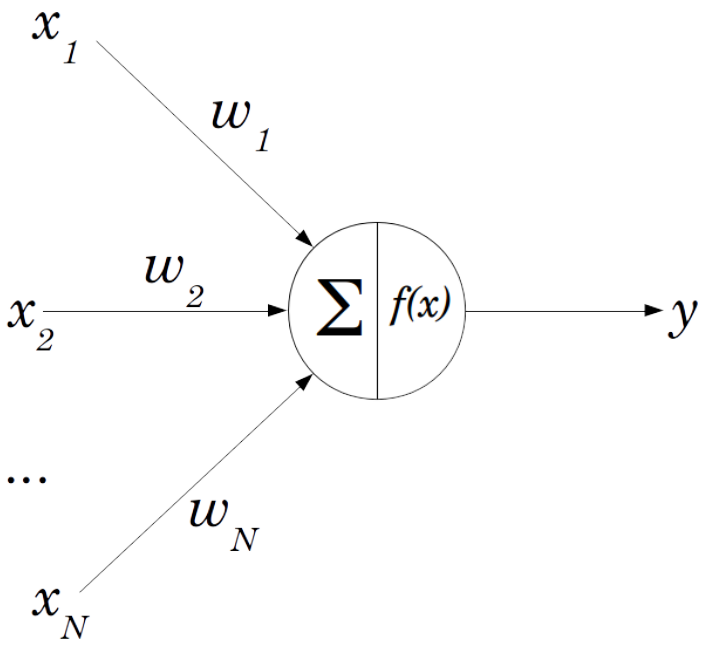


Figure 2.2: *The Perceptron. Each value in the input vector is multiplied by its coefficient, everything is summed and, given the sign of the result, a label is predicted.*

It becomes evident why w's coefficients are called weights. Note also the similarity with (human) brain neurons and synapses, presented in Figure 2.3. They both receive multiple inputs that are adjusted by the weight of the connection, compute their output by an internal function before making it available. In fact, the neuron provided the motivation for the introduction of the perceptron and its evolution. We further elaborate on the similarity and its evolution later. Moreover, to simplify notation, we consider x to be the input vector plus an extra 1 the end, intuitively $x \leftarrow [x; 1]$, so we can include w_0 within w and, finally, $f(x; w) = wx$. Additionally, we consider the labels to be $y = \pm 1$ and $y = 1$ corresponds to a positive sign. Using this notation, we can tell whether a prediction

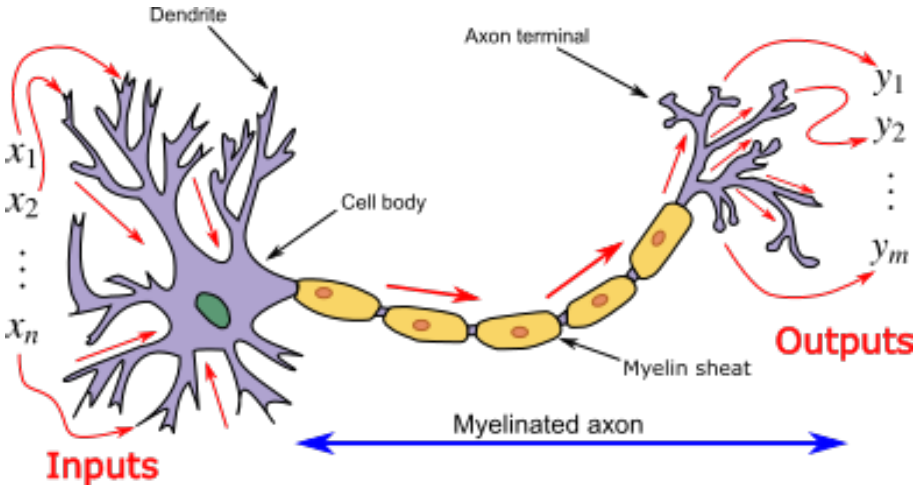


Figure 2.3: *A human brain neuron. It receives the Action Potential from other neurons through its synapses. Using the Action Potentials, it computes and sends its Action Potential to the neurons it is connected to through the Synaptic terminals. Source: [4].*

is correct if

$$f(x; w) \cdot y > 0 \tag{2.2}$$

However, we need to find an appropriate value for the weights w. To do this, we use the **Perceptron algorithm** that guarantees convergence within a finite amount of steps. The algorithm proceeds as follows: We start with a randomly initialized weight vector $w[0]$. Thereafter, for every (x_i, y_i) in the dataset, if:

- $f(x_i; w[t]) \cdot y_i > 0$, then $w[t+1] = w[t]$
- $f(x_i; w[t]) \cdot y_i < 0$, then $w[t+1] = w[t] + \rho \cdot y_i \cdot x_i$, ρ is a positive parameter which we can set arbitrarily. This is chosen because $w[t+1] \cdot x_i = w[t] \cdot x_i + \rho \cdot y_i \cdot \|x_i\|^2$, which pushes the inner product towards the desired sign.

We iterate over the dataset until all examples are classified correctly, when the sequence converges that is, and we naturally set $w = w[t_{final}]$. Note that if the examples in the dataset do not amend themselves to such a solution, the algorithm will never converge, as it shouldn't.

There is another, more intuitive interpretation of the decision the perceptron, that of the *Decision surface*. We first note that $f(x; w) = 0$ is a line in N-dimensional space, where $N - 1$ is the dimension of the input examples (+1 for the w_0 term). It is easy to see that two points in the same half-hyperplane have the same sign w.r.t $f(x; w)$. Therefore, $f(x; w) = 0$ separates the N-dimensional space into two half-hyperplanes, each of which contain points that are predicted to belong to the same class. This is illustrated in Figure 2.4 for a certain representation of cats and dogs, which also demonstrates the movement of the surface as the weights are modified by the Perceptron algorithm.

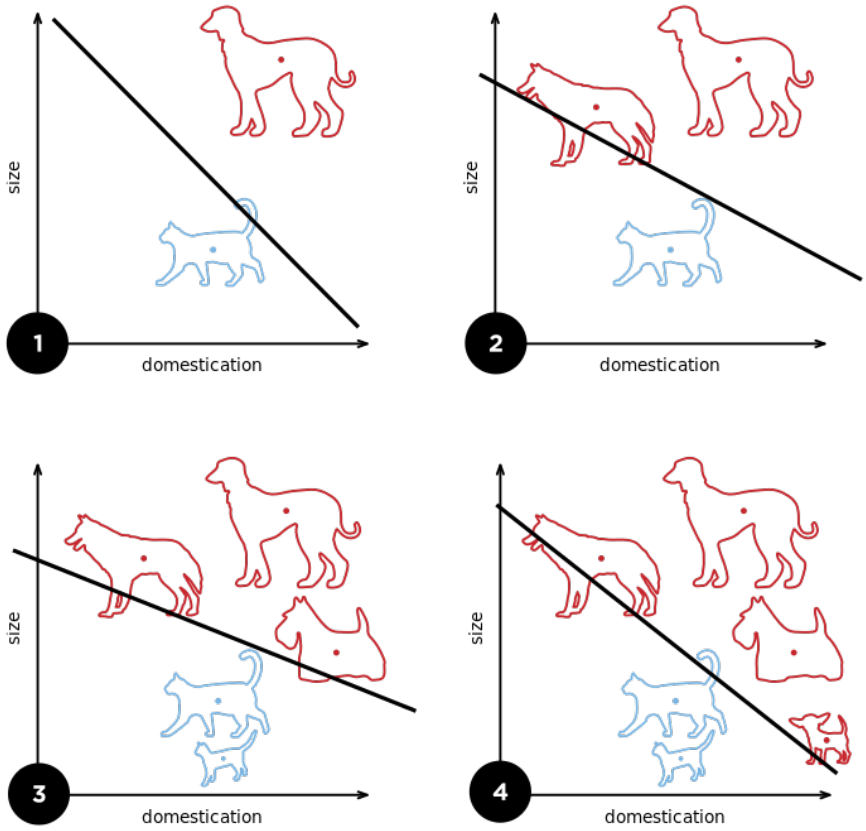


Figure 2.4: *The Perceptron as a decision surface. Source: [5].*

Feedforward Neural Networks

Although the perceptron mimics the process of a single neuron, intelligence arises from the collaboration of numerous neurons constantly exchanging messages, which is something the plain Perceptron lacks. The incorporation of multiple perceptron-like models with a single computational unit led to Multilayer Perceptrons (MLP), or simply Feedforward Neural Networks (FFNN).

Notice that the perceptron can be thought of as having a Step function as its **output activation function**, i.e. the function:

$$h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{2.3}$$

is applied after the inner product between the input and the weights is computed. It turns out that using continuous and for the most part differentiable functions can lead to effective training of larger collections of perceptron-like computational units. A unit that performs the same computation as the perceptron but use that kind of function is called (artificial) neuron.

MLPs are the quintessential DL models and every other DL models is more or less a derivative of them. The goal of an MLP is to approximate some unknown function f^*. A

classifier, for example, implements a function that maps an input to a category, similar to the perceptron. A FFNN is essentially a mapping $y = f(x; \theta)$ that tries to learn the parameters θ (the equivalent of w for the perceptron) that result in the best possible approximation of the real function.

The "Feedforward" part of their name denote the fact that the information flows forward; from the input to the hidden computations made using parameters θ and finally the output value y. There are no feedbacks, i.e. no parameter is used multiple times per input. Otherwise, such networks are called Recurrent Neural Networks (RNN), which are discussed later.

The "Network" part of their name, on the other hand, is derived from the structure of the computation, where it is represented as a collection of different functions in the form of an directed acyclic graph. That is to say, if a MLP represents a function f, an alternative representation can be a chain of functions $f(x) = f^{(n)}(f^{(n-1)}(\cdots(f^{(1)}(x))))$, where $f^{(i)},\ i \in \{1, \ldots, n\}$ are generally different functions. $f^{(1)}$ is called the **first** or **input layer**, $f^{(2)}$ the **second layer**, ..., and $f^{(n)}$ is the **output layer**. n is the **depth** of the network, and the **Deep** Learning terminology arose from increasing the depth of such networks when data and computational power became ample. When training the FFNN to learn to predict y_i based on the input example x_i, the only value the network computes whose correctness is dictated by the training set is the output y. Therefore, all layers but the output layer (that outputs that y) are called **hidden layers**. A FFNN can be seen in Figure 2.5, presented as layers of neurons. Each layer computes a multivariate function $f^{(i)}$. It becomes evident that parameters θ are split between layers. The visualization also makes apparent that, other than the depth of the network, we have to select the *width* of each layer, meaning the number of neurons in each layer.

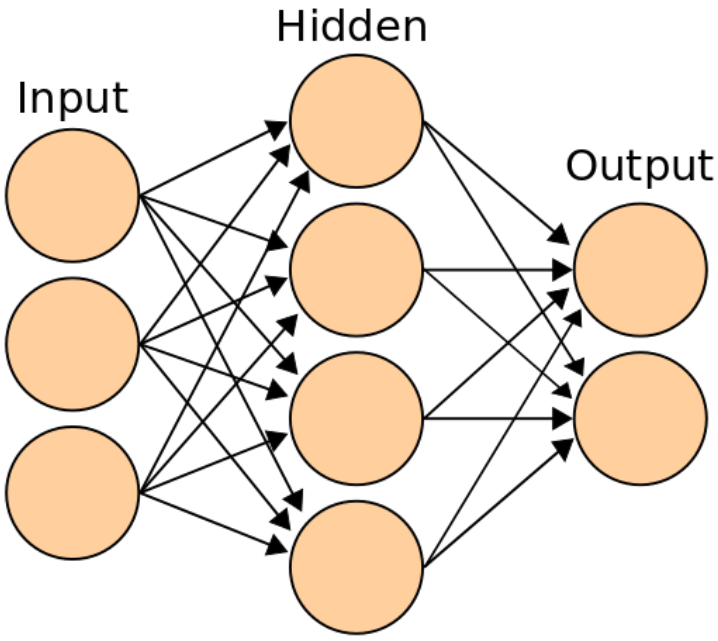


Figure 2.5: *A Multilayer Perceptron / Feedforward Neural Network presented as a chain of layers of artificial neurons.*

Finally, the "Neural" part of the name refers to the fact we noted earlier, that of the computation of a single unit (like the perceptron) being similar to that of a single neuron. In the case on a network, this also incorporates the cooperation of multiple units, which are generally thought to give rise to intelligence. This analogue to natural cognitive functions is the rationale behind naming the networks "neural". Note, nevertheless, that the parallel between artificial and natural neural networks is mostly qualitative and less quantitative. Actual neurons perform more complex computations and do not need any synchronizing in the sense that there is no mechanism for a neuron to wait all of the neurons it receives input from to commence its function, in contrast to the chain of functions we presented earlier.

We briefly discussed activation functions earlier. Let's elaborate a little further. Each layer i in the network can be represented by the following operations:

- $z_i = W_i \cdot a_{i-1}$, where a_{i-1} is the column-vector output of the i-th layer (or the input if this is the first layer, i.e. $a_0 = x$), W_i is the weight matrix of the layer, where each row contains the weights of a single neuron in the layer, and z_i is an intermediate value within the layer. Note that this implicitly includes a **bias** term b_i, since we consider x to include an extra dimension as we noted earlier. That makes b_i the last column of W_i.
- $a_i = f_a^{(i)}(z_i)$, where $f_a^{(i)}$ is the activation function, an element-wise *non-linear* function.

We emphasize the non-linearity of the activation function because, as it can be easily seen, a network with linear activation functions is equivalent to a single layer with a linear activation function. Therefore, non-linearities are integral to approximating more complex functions, as proven for sigmoids for example in [58]. Some common activation function are illustrated in Figure 2.6.

The sigmoid function is almost mandatory in settings where the output value corresponds to a Bernoulli distribution, namely classification between two categories. For more categories, a Multinoulli distribution must be outputted and the *softmax* is utilized. Given the intermediate output vector z of the output layer, which in the case of the softmax are referred to as *logits*, the softmax's j-th value is computed as:

$$softmax(z)_j = \frac{\exp z_j}{\sum_i \exp z_i}. \tag{2.4}$$

Although it is not strictly an element-wise function, it remains differentiable. We can easily see that the sum of all the output values is equal to 1, hence the Multinoulli distribution.

Gradient Descent

The question of how training of arbitrarily configured network (we have to set depth, widths, activation functions) remains. The answer is the **Gradient Descent**. Gradient Descent is a simple optimization algorithm that provides a means to iteratively find close

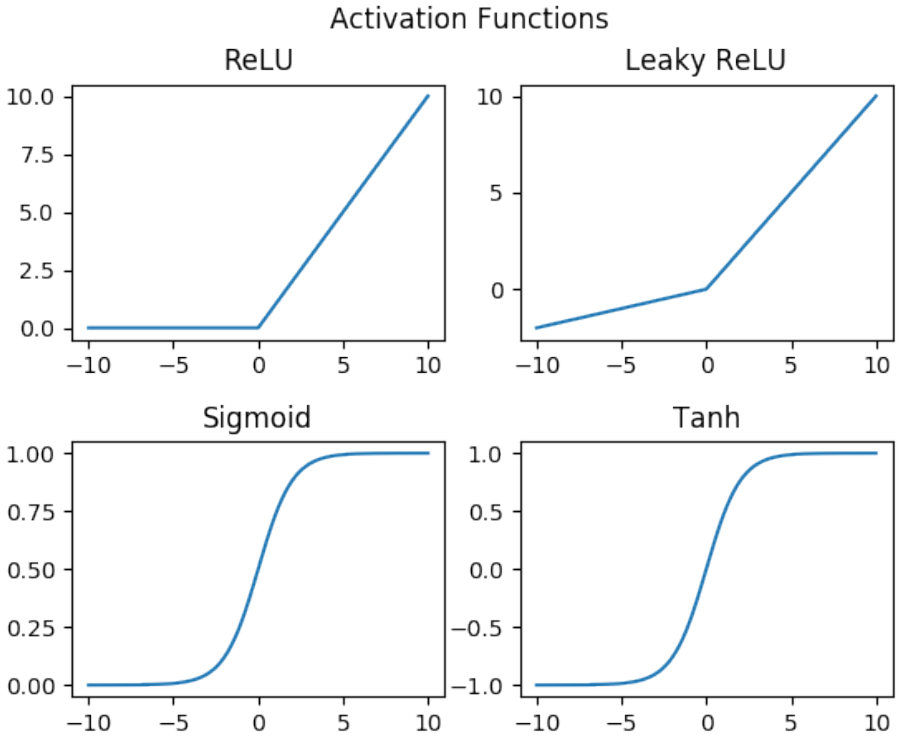


Figure 2.6: *Multiple activation functions.*

to optimal arguments / parameters of a differentiable function with access only to the derivative at the current configuration of parameters. In detail, given a differentiable function R, we can express the function by its Taylor series approximation:

$$R(w[t] + \Delta w) \simeq R(w[t]) + \nabla R(w)|_{w=w[t]} \Delta w \tag{2.5}$$

So, given parameters $w[t]$, if we are looking for the value of Δw to modify it so as to lower the value of R, then a sensible choice is to select a Δw in the opposite direction of $\nabla R(w)|_{w=w[t]}$, i.e. $\Delta w = -a\nabla R(w)|_{w=w[t]}$, $a > 0$. Then, we end up with $w[t+1] = w[t] - a\nabla R(x)|_{x=w[t]}$ and

$$\begin{aligned} R(w[t+1]) = R(w[t] + \Delta w) &\simeq R(w[t]) - a\|\nabla R(w)|_{w=w[t]}\|^2 \\ &\leq R(w[t]). \end{aligned} \tag{2.6}$$

The process is repeated until the desired convergence criterion is met. The process is illustrated graphically in Figure 2.7. It is important, however, to note that the Taylor series approximation holds for small Δw, so a careful choice of a, the **learning rate**, must be made.

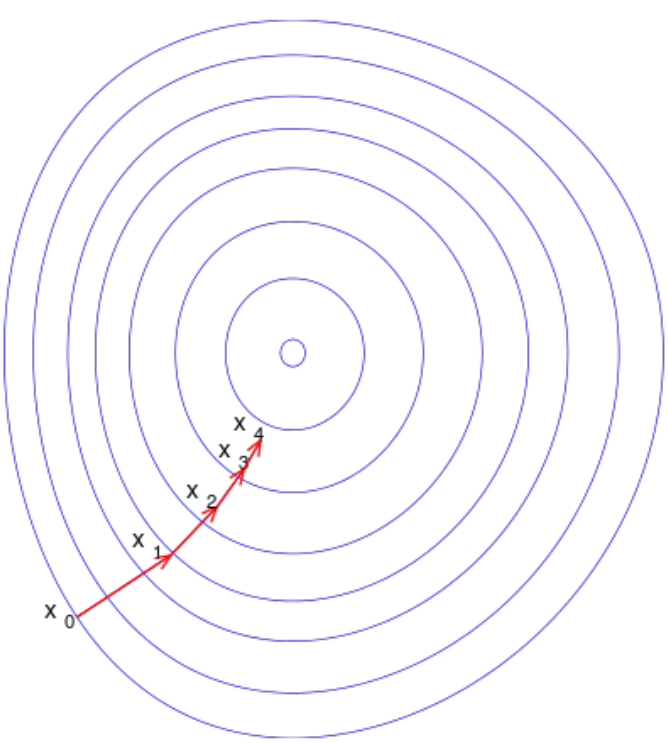


Figure 2.7: *Illustration of Gradient Descent on a 2-dimensional parameter space. Source: [6].*

Backpropagation

So, we see that in order to use the Gradient Descent algorithm we only need the gradient of the function we are trying to minimize w.r.t the parameters of the model. To get that, we basically need the chain rule of calculus, $\frac{df}{dx} = \frac{df}{du}\frac{du}{dx}$. This is how the **Backpropagation** algorithm operates, which is simply an efficient algorithm to compute the gradient of the loss function w.r.t all the model's parameters. It is succinctly summarized in the following equations:

$$\begin{aligned}
&1. \quad \delta^L = \nabla_a R \odot f_a^{(L)\prime}(z_L) \\
&2. \quad \delta^l = ((W_{l+1}^T \delta^{l+1})) \odot f_a^{(l)\prime}(z_l) \\
&3. \quad \frac{\partial R}{\partial b_{l,j}} = \delta_j^l \\
&4. \quad \frac{\partial R}{\partial W_{l,jk}} = a_{l-1,k}\delta_j^l
\end{aligned} \tag{2.7}$$

where $\nabla_a R$ is the gradient of the risk w.r.t the output of the last layer, $f_a^{(l)}$ is the activation function of the l-th layer, L is the depth of the network, a and z remain as we defined them aka the output after and before the activation function of a layer, the first subscript and the exponent denote the layer while the second denotes the index with the matrix /

vector, W is considered to be a weight matrix without its last column, which we referred to as bias and is represented by b, and $\odot$ is the Hadamard product.

The first two equations give us a way to compute some intermediate values δ and the last two utilize these values to compute the gradient of the risk w.r.t the desired parameter. We repeat this process until Gradient Descent converges.

To conclude this section, we also discuss some useful loss / risk functions. The first is the *binary cross-entropy* (BCE) loss, which is suitable for binary classification tasks. It is defined as:

$$BCE(y, \hat{y}) = -\left[y \log \hat{y} + (1 - y) \log (1 - \hat{y})\right], \tag{2.8}$$

where y is the *groundtruth* label, equal, in this case, to either 0 or 1, and $\hat{y}$ is the sigmoid output of a NN. It is easy to see that BCE is minimized when $\hat{y} = y$. Another important loss function for classification tasks in general and the generalization of the BCE is the *cross-entropy loss*, defined as:

$$\mathcal{L}_{XE}(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i \tag{2.9}$$

where y a probability vector, usually with only one element equal to 1 and all others 0, and $\hat{y}$ is the vector output of a softmax output function. Again, the loss is minimized when the vectors are equal (here we can actually deviate where $y_i = 0$ but due to the softmax we would subtract from the category we really care about).

2.1.3 Convolutional Neural Networks

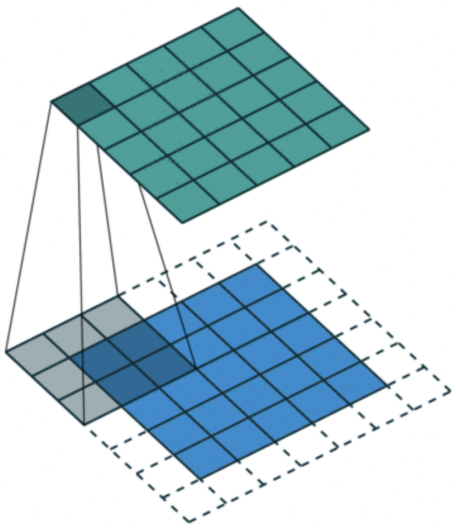


Figure 2.8: *Convolutional neurons have a limited receptive field. Source: [7].*

Another success story of neuroscientific influence on ML are Convolutional Neural Networks (CNN). CNNs are based on the fact that natural visual systems have a limited receptive field, i.e. a receptive unit focuses on a small part of the picture being perceived by

an eye as a whole. This units are generally represented as in Figure 2.9 in Neuroscience. In ML, the way that is translated to NNs is by arranging the neurons of a layer in grids (multiple 1D grids per layer for sequential data, multiple 2D grids per layer for images, which could also be spectrograms) called channels. Moreover, each neuron in the grid usually receives input from a small neighborhood around its "corresponding" neuron in the previous layer, as illustrated in Figure 2.8. All other connections are zeroed out. Finally, in a CNN, there is extensive **parameter sharing**, which we discuss after presenting the mathematical formulation.

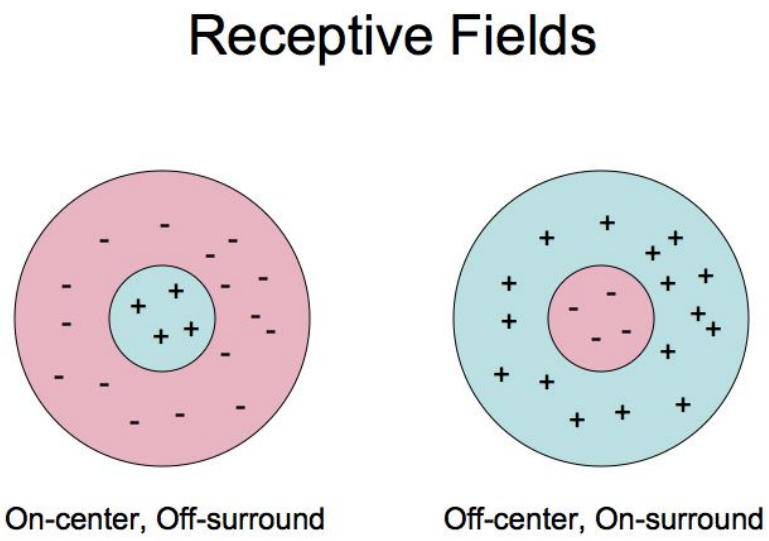


Figure 2.9: *Receptive fields of nerves in the retina.*

Mathematically, convolution is an operation between two matrices, I and K, that results in a third matrix. For historical reasons, we consider I to be a 2D matrix, aka an image, similar to the output. K is called the **kernel** of the convolution. The operation is defined as

$$conv(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.10}$$

To get a firm grasp of the above equation, consider the kernel to be infinite, with its non-zero elements being restricted to a rectangle, which m and n iterate over. So, we see that the (i,j)-th element of the resulting matrix results from the element-wise product between the offset kernel and the image around the (i,j)-th element, followed by their summation. The only parts of the image that actually affect the result are the ones that happen to coincide with the non-zero elements of the kernel.

That is the definition of the convolution that is useful for ML. In reality, that is the definition of *cross-correlation*. Nevertheless, given we can control the kernel, the definitions are equivalent. In the ML setting, $conv(i,j)$ is the intermediate value $z_{i,j}$ of the layer,

I are the activations of the previous layer and K the connection weights. The kernel, in practical applications is usually much "smaller" than the image (except maybe from when they are used in the last layers of a deep CNN). Moreover, the kernel can be considered to be centered at $(0,0)$ and usually square, therefore m, n can be thought to iterate over $\{-w, -w+1, \ldots, w-1, w\}$, thus giving us the neighborhood around the "corresponding" element of the previous "image", as shown in Figure 2.8. Additionally, note that we use the same kernel, i.e. the same weights, to compute the intermediate values of the layer. That is the *parameter sharing* we eluded to earlier. This is one of the strengths of CNNs, not a limitation, as it is able to detect features in an image wherever they are on it, as the kernel scans the image in its entirety. For instance, a cat identifier should be able to tell a cat is in an image no matter its position in it.

Another important consideration is that the "images" in a CNN are not two-dimensional, but rather three-dimensional. The convolution is expanded to include another dimension to both the image and the kernel, with a length of the new dimension equal to the channels of the input image, which essentially results in performing convolutions with each channel of the input based on a dedicated 2D kernel for each and, finally, summing the results over.

CNNs have become really prominent in DL [21, 22, 49] due to the zeroing out of the vast majority of the parameters coupled with parameter sharing for all the others, both of which result in a huge reduction in trainable parameters and, in turn, allow for very deep networks. The reason the restricted receptive field is effective follows by the structure of natural images which, by virtue of natural selection, led to our own visual systems resembling CNNs, i.e. the locality of information a visual system needs to correlate to grasp a scene.

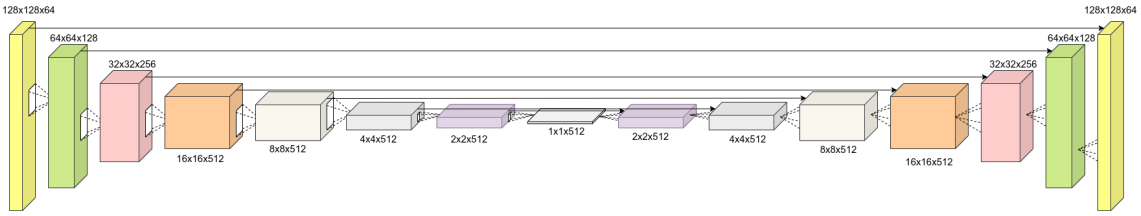


Figure 2.10: *U-net architecture [8]. The numbers attached to each layer denote its channels and the size of the kernel.*

2.1.4 Recurrent Neural Networks

So far, we have seen networks that can handle input of fixed size. That means we have to either crop the input or zero-pad it. For this reason, the Recurrent Neural Networks (RNN) were introduced. RNNs are specialized for processing a sequence of values $x_1, \ldots, x_\tau$, where (for most RNNs) τ is unconstrained. To achieve this, we again use parameter sharing, as shown in Figure 2.11. This allows the generalization to sequence lengths not seen during training and the sharing of statistical strength across sequences of different length or across time. That is to say, as we noted for CNNs, a piece of information can occur in many different time steps, e.g. "You can find the cat in the garden"

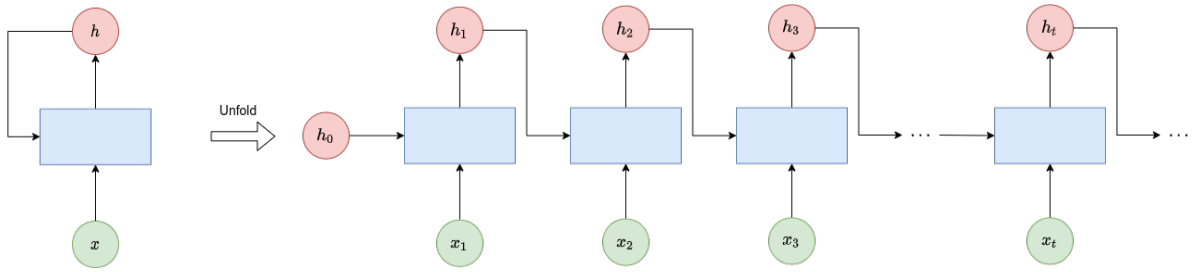


Figure 2.11: *Recurrent Neural Networks. On the left, we can see its circuit diagram and, on the right, the same network as an unfolded computational graph.*

and "The cat can be found in the garden" have exactly the same meaning, however the entities that convey the meaning in the sequence are in reverse order.

Mathematically, the classical form of a dynamical system parameterized by θ is:

$$s_t = f(s_{t-1}; \theta) \tag{2.11}$$

Extending the equation to include external input, we get

$$s_t = f(s_{t-1}, x_t; \theta) \tag{2.12}$$

Using the established notation for the activations of layers whose output is not dictated by the training examples as *hidden*, the final formulation commonly used is

$$h_t = f(h_{t-1}, x_t; \theta) \tag{2.13}$$

In order to perform training using a sequence of finite length τ, we perform an **unfolding** of the network. For example:

$$\begin{aligned} s_3 &= f(s_2, x_3; \theta) \\ &= f(f(s_1, x_2; \theta), x_3; \theta) \end{aligned} \tag{2.14}$$

which is simply a FFNN with parameters shared between layers, so the training process can proceed as-is. The only consideration is the aggregation of the gradients accumulated for each layer of the network, since in reality it concerns the same parameters. The contributions are normally summed up.

There are many alternatives of this basic model. First and foremost, we have to consider what we are trying to model. Do we simply want a single output, e.g. a Multinoulli distribution, or are we expecting a sequence as a result. In the first case, a linear layer can be applied at the end of the input sequence on the hidden representation of the RNN, followed by a softmax, to get the probabilities. In the latter case, we can build a FFNN on top of h to model the output based on the hidden representation of the RNN at each timestep of the output sequence. However, we also have multiple ways of doing that. One alternative is to model the output sequence concurrently with the input one. Another, shown in Figure 2.12, is to first read through all the input sequence, in which case the

final hidden output h is expected to summarize all the input sequence for the rest of the pipeline, where a RNN, with no external influence, outputs the desired sequence.

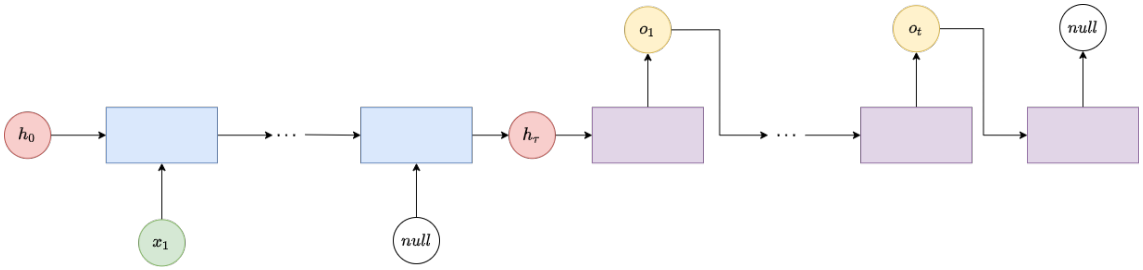


Figure 2.12: *Recurrent Neural Networks used to model sequence-to-sequence processes.*

We have yet to discuss how the parameters θ are allocated. There is no need for the FFNN used in each time step, represented by $f(.;\theta)$, to be a single layer, nor for only the final hidden representation to be passed to the next time step. There are no restrictions to the configuration of the inner network and the effectiveness of each can, more or less, be tested only empirically.

Long Short-term Memory

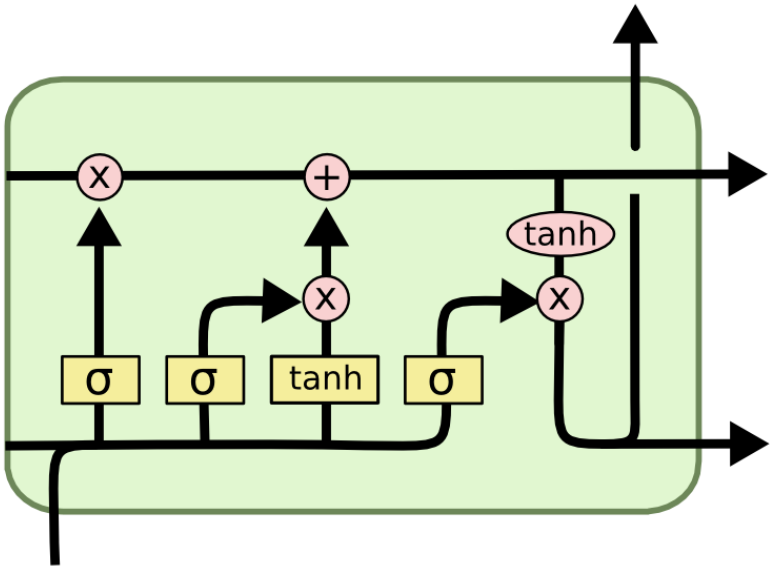


Figure 2.13: *LSTM cell. Source: [9].*

Even though the theoretical foundation for RNNs is solid, in practise their training can be very arduous. To see why this happens, we have to give a second look to the equations used to compute the gradients of the risk w.r.t the parameters. There, we see that in order to backpropagate the information from a layer to its predecessor, we have to multiply by the weight matrix of the said layer (Equation 2 in 2.7). In the RNN setting, this is translated

to repeatedly multiplying with the same weight matrix, rendering the relation between the length of the sequence and the elements of the matrix approximately exponential. If the sequence is long enough, then we get the **Vanishing/Exploding gradient** problem. This is important because it can lead to bad convergence properties at best, divergence otherwise. In the case of vanishing gradients, we also have to consider that information extracted in distant time steps in a sequence cannot be communicated and their relationship cannot be modeled, even if their coexistence alters the meaning of the sequence.

To improve training, [59] introduce Long Short-Term Memory (LSTM). This work essentially introduced the Constant Error Carousel units, which enable the backpropagation of the gradients unscathed. Further developments resulted in its current formulation, namely

$$\begin{aligned}
&1. \quad f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
&2. \quad i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
&3. \quad o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
&4. \quad \tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
&5. \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
&6. \quad h_t = o_t \odot \sigma_h(c_t)
\end{aligned} \tag{2.15}$$

where f_t is the output of the **forget gate** of the LSTM, i_t is the output of the **input/update gate**, o_t is the output of the **output gate**, c_t is the state of the cell, $\tilde{c}_t$ is the cell input activation, σ_g is the sigmoid function, σ_c is the hyperbolic tangent function, σ_h is the hyperbolic tangent function (or the identity function as suggested in [60]).

2.1.5 Representation Learning

Representation Learning or Feature Learning consists of learning representations of the data that render extracting essential information easier when building classifiers or other predictors. This replaces manual feature engineering or automatic but not-data driven feature extraction, e.g. Scale Invariant Feature Transform [61], and allows a machine to both learn the features and use them to perform a specific task.

As demonstrated in Figure 1.3, DL models learn representations as part of their training [62]. To access this internal representations, all we really need to do is remove the last layers of a NN, usually catered towards the specific task the NN was trying to solve, e.g. ImageNet classification [48]. If the training set is diverse enough, then general representations, useful to many tasks, can be learned by the NN.

CNNs trained on ImageNet are routinely utilized as feature extractors, like the ResNet [49] for example in [16, 18, 46], or GoogleNet [63] in [64]. The merits of using a pretrained CNN to extract features are mostly two-fold: first, the network is usually trained on a large database, bestowing external knowledge to the features. Second, it can reduce the dimensionality of the input vectors, a form of compression. For instance, for a 256×256

image, a common size for DL, a ResNet can reduce the dimension of the input vector from ~ 200000 dimensions to only ~ 2000, i.e. 100 times less input data to parse and handle. Feature extraction is, of course, not lossless, but it reduces the number of necessary parameters whilst retaining a significant portion of the original information.

2.2 Generative Networks

Generative models represent the probability distribution function over multiple variables in some way. Some models allow the probability distribution function to be evaluated explicitly. Other Generative models support operations that only implicitly make use of the probability distribution, such as drawing samples from the distribution. **Generative networks** belong in the latter category. In this subsection, we discuss these Generative Networks, which allows sampling from the posterior of the desired space given a latent variable. We refer to these networks as "generators" as well.

2.2.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) [65] are more easily understood as a framework to train a generator efficiently. They achieve this by creating a minimax game between said generator and a discriminator, whose job is to discriminate between real data and data generated by the generator. The generator is thus trained to fool the discriminator. Of course, both the generator and the discriminator are NNs. The loss function of the minimax game is defined as

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{z\sim P_Z}[\log(1 - D(G(z)))] \tag{2.16}$$

while the minimax game is simply

$$G = arg\min_G \max_D \mathcal{L}_{GAN}(G, D), \tag{2.17}$$

where G is the generator, D the discriminator, P_r the distribution of real data and P_Z a noise distribution to draw latent variables from. It is proven in [65] that, given enough capacity, the generator can learn the true distribution of the data.

To implement this scheme, the generator and the discriminator are trained interchangeably. The discriminator is trained multiple times in a row before the generator can be trained once, as suggested by [65].

Wasserstein Generative Adversarial Networks

The minimax game of GANs can be shown to minimize a distribution divergence metric [66]. Therefore, the GAN framework is extended to be able to minimize other distribution distance metrics. Wasserstein distance is chosen in [66, 67] because it is continuous and differentiable almost everywhere, and allows more sequences (the weights of the generator can be thought of as a sequence when they are updated one step at a time) to converge. The Wasserstein GAN (WGAN) utilizes the following loss function:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{z\sim P_Z}[D(G(z))] \\ - \lambda \cdot \mathbb{E}_{(\hat{x},a)\sim P_{\hat{X}}|P_Z}[(\|\nabla_{\hat{x}} D(\hat{x}, a)\|_2 - 1)^2]\end{aligned} \tag{2.18}$$

The third, new term is included by [67] to enforce a Lipschitz constraint on the discriminator, which is essential for the Wasserstein distance formulation.

Both losses can be expanded to include conditioning variables, i.e. an extra variable next to the random latent variable that dictates the essense of the output, e.g. the label of the category we want to get a sample from.

2.2.2 Autoencoders

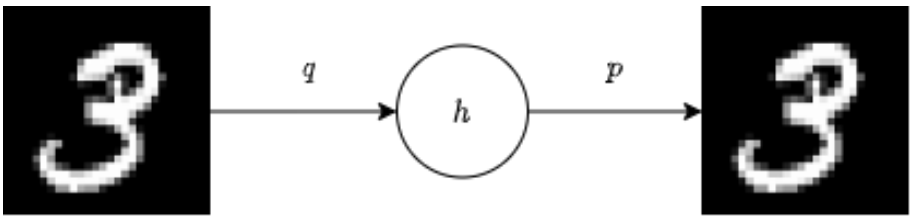


Figure 2.14: *An Autoencoder used to model black and white digits.*

An Autoencoder (AE) is a NN that is trained to copy its input to its output. Internally, it has a hidden layer whose output h is referred to as **code**, used to represent the input in a different space. The whole network can be thought of as consisting of an **encoder** $h = E(x)$ that encodes the input, and a **decoder** $\hat{x} = p(h)$ that decodes the code. It is presented graphically in Figure 2.14.

If the NNs p and q have enough neurons to simply act as identity matrices, then the AE is of no use to us. Therefore, the encoder is designed to perform compression of the input or regularization is applied with a penalty term to avoid lossless compression. In this manner, AEs become useful for denoising their input and/or manifold learning [68], among others. Although AEs are not generative networks per se, they include the Variational Autoencoder (VAE) in their class, which can be used as such.

Variational Autoencoders

The VAE is a particular kind of AE that results from trying to perform inference in a directed probabilistic model [69]. While the details of the derivation and the jargon are of no importance to us, there are several desirable properties of the VAE that make it useful as a NN:

1. The autoencoding arises naturally from its theoretical basis, i.e. it provides a mathematically principled way to derive an AE.

2. In contrast to most AEs, the decoder can be used on its own as a generator. This happens because we can control the distribution that the encoder outputs by selecting a prior distribution for it to match. Therefore, we can sample from the prior

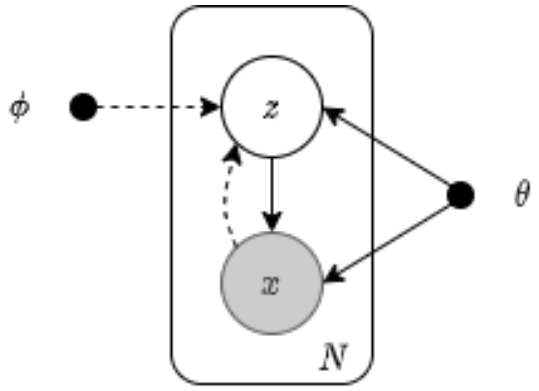


Figure 2.15: *The directed probabilistic model that characterizes the VAE.*

distribution and reasonably expect a plausible output that belongs to the input space.

We can achieve such feats by the particular choice of network formulation and risk function. The loss is defined as

$$\begin{aligned} \mathcal{L}_{VAE}(\partial, \phi; x) &= D_{KL}(q_\phi(h|x)||p_\partial(h)) - \mathbb{E}_{q_\phi(h|x)}\left[\log p_\partial(x|h)\right] \\ &= D_{KL}(q_\phi(h|x)||p_\partial(h)) + \mathcal{L}_{REC}(\partial, \phi; x). \end{aligned} \tag{2.19}$$

Equation 2.19 is the inverse of the Variational Lower Bound of the likelihood. $p_\partial(z)$ is the prior distribution of the code, $D_{KL}(p(x)||q(x)) = \int p(x) \log \frac{p(x)}{q(x)} dx$ is the Kullback-Leibler (KL) divergence, $\mathbb{E}_q$ is the expectation w.r.t the PDF q, ∂ the parameters of the decoder and ϕ the ones of the encoder. However, we see that KL divergence requires integration and p_∂ is the probability of the input x arising from the code rather than its reconstruction.

To eschew the inconvenience, we introduce the notion of MLPs as probabilistic encoders and decoders. The decoder is defined as a Bernoulli MLP which outputs a multivariate Bernoulli distribution and, therefore,

$$\log p(x|z) = \sum_i x_i \log \hat{x}_i + (1 - x_i) \log (1 - \hat{x}_i) \tag{2.20}$$

Instead of treating the resulting $\hat{x}$ as probabilities however, we consider it the reconstruction, given that $\mathcal{L}_{REC}$ is now minimized at $\hat{x} = x$. Note that to use the Bernoulli MLP, we need to normalize the input to the $[0, 1]$ range.

The encoder is defined as a Gaussian MLP. It outputs the mean and the diagonal elements of the, by our definition, diagonal covariance matrix (in reality, we output the logarithm of those to avoid having a zero variance) of the code instead of the code itself. Then, we sample the code from that distribution using the reparameterization trick, according to which we actually get the code using $h = m(x) + \epsilon\sigma(x),\ \epsilon \sim \mathcal{N}(\epsilon; 0, I)$ instead of $h \sim \mathcal{N}(h; m(x), o^2(x)I)$, which allows for backpropagation. We choose the prior of the code to be standard multivariate normal, because the KL divergence can be computed analytically for two normal distributions:

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = 0.5\left\{tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - k + \ln\frac{|\Sigma_1|}{|\Sigma_0|}\right\}. \tag{2.21}$$

2.3 Few-shot Learning

Few-shot Learning (FSL) is a classification task that departs from the "standard" supervised setting. It does so in two ways. First, the samples during testing do not belong to the categories that appeared in the training set. Second, and a consequence of the first difference, since we do not have any information on the number of test classes or their characteristics, we are provided with a small set of examples of the new classes, named **support set**.

The naive approach would be to train a "standard" classifier on the support set, like a FFNN. While this is possible, it is discouraged for many obvious reasons. One, we do not take advantage of the training set at all. Two, NNs require substantial amounts of data to be trained, while the support set is, by definition, a small set.

Many approaches have been suggested for FSL tasks. An integral component of them is the manner in which the network is trained, not the architecture itself. Frameworks are not simply designed for learning, but for "learning to learn". That is to say, we teach the network how to extract knowledge from fake support sets we design during training in order to prepare the network for the test setting. We do so by training it in an *episodic manner*. In particular, instead of batched, we sample "episodes". An episode is characterized by the existence of a support set and a *query set*, which is simply the set with examples that need classifying. Episodes mirror the test setting, which is the reason training is carried out as such in an effort to better cater the network to the actual setting it is going to be used in.

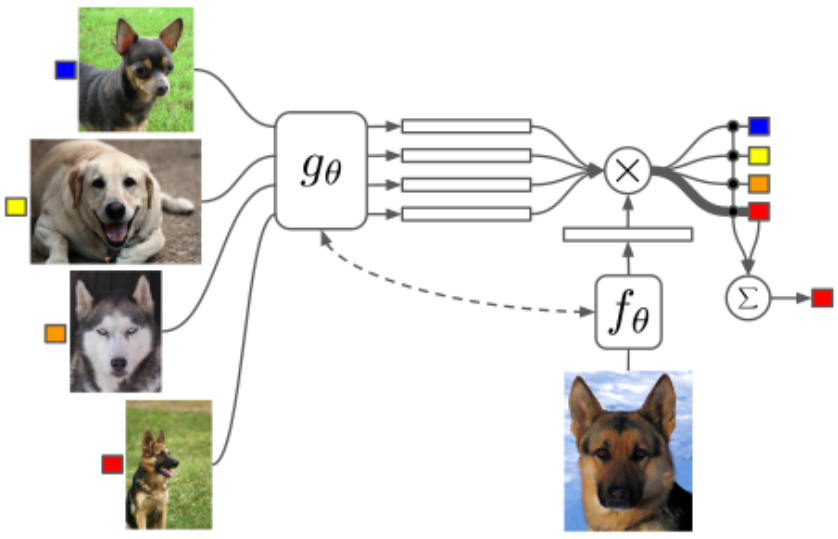


Figure 2.16: *Graphical representation of a Matching Network. Source: [10].*

Some typical approaches are Prototypical Networks [17] and Matching Networks [10]. The former uses a NN to map the support set examples to a metric space, where they are aggregated and used as prototypes for each class. Queries are also mapped to the same space and categorized to the closest prototype. The former uses LSTMs to sequentially compute embeddings with the knowledge of where previous samples were mapped. However, the support is limited to only one example per class. A graphical representation of the framework can be seen in Figure 2.16. The two frameworks are fused into one in [70], where a Prototypical Network is used to compute a prototype per class, the collection of which are in turn treated as a support set with only one example per class so a Matching Network can be used.

2.4 Zero-shot Learning

Zero-shot Learning (ZSL) takes a step further into the direction FSL compared to standard settings. Instead of providing a small support of the new classes for the learner to adapt to them, it instead provides auxiliary descriptions of the new classes. This means we are not given any supporting examples during training nor during testing for the classes we are tasked with classifying!

ZSL is an extreme scenario for ML and DL research as it has developed, but it can prove realistic nonetheless. Not all phenomena, processes or tasks that people are interested in automating have ample labeled data available, and if they do it is not a guarantee that they are widely available to the public. From an ethical standpoint, developing techniques that can respond and provide insightful information on never-before-seen (for the algorithm) circumstances and objects can help level the playing field between big corporations with the resources to amass big datasets and small businesses and startups that have to compete against them. The advantages from a scientific and theoretical perspective of solving such tasks are significant but have to play a second fiddle for the time being.

Most current benchmarks concern image classification given vectors of semantic attributes. Each value in that kind of vector signifies the frequency with which people (usually Amazon Turkers) responded that a corresponding feature (e.g. are there any people in the picture) characterizes instances of that class. To do so, they were shown a few randomly sampled images of that class, so as to reduce the variance of the result as much as possible without making the task tedious for the annotators. Example datasets are the Caltech UCSD Birds 200 (CUB) [19], which consists of 200 bird species located in North America, with 57 images per species give or take, Animals with Attributes 2 (AwA2) [14] with 50 animal species and a huge variety of number of samples per class, and SUN Scene Classification [20], with 717 categories of scenes ranging from cathedrals to submarines. All include class-level attributes, of which CUB contains 312 semantic features, SUN 102 and AwA2 85. Some works [64] have also collected per image text descriptions and used 1D CNNs and RNNs to perform Representation Learning on these descriptions. These are routinely used in Text-to-Image synthesis works [71, 72, 73, 74] because they are richer and, more importantly perhaps, are available multiple times per image and not once per class. They have also been used in ZSL tasks [18, 47], but we refrain from

testing our methods using these descriptions due to the sheer volume of attribute usage.

Over the years, several techniques have emerged. Initial attempts tried to estimate a compatibility function between the objects and their descriptions of a (sometimes piece-wise) bilinear form [38, 39, 40, 41, 42, 43]. Other essayed to estimate the posteriors of one group given the other, performing the final classification based on the similarity of the most likely choice and the available data [36, 37].

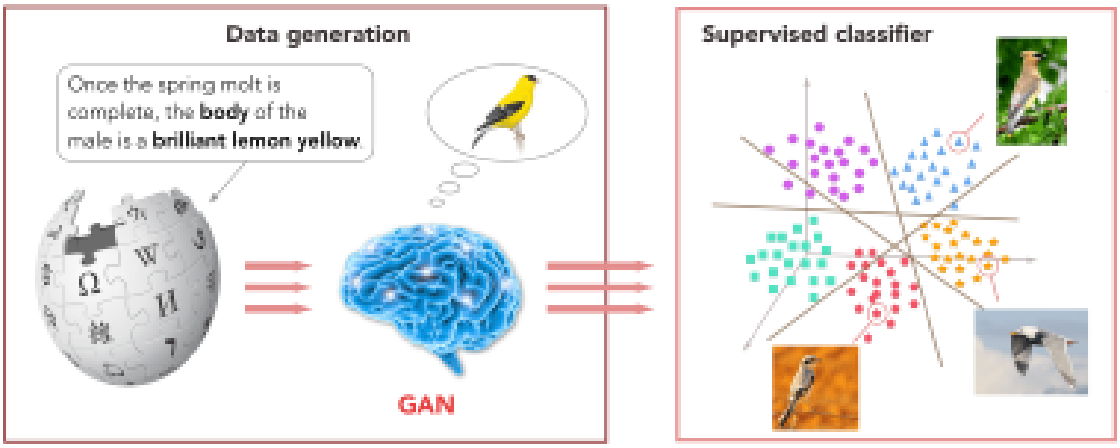


Figure 2.17: *Generative approaches presented as an analogue of imagining/hallucinating never-before-seen birds given their Wikipedia article. Source: [11].*

In this work, we are concerned with the current state-of-the-art, *generative approaches* to ZSL. The basic algorithm was introduced concurrently in [18, 11] (see also Figure 2.17). It uses a generator, which, given a description, can generate infinite amounts of images (implementations use a pretrained feature extractor instead of the image itself) that correspond to that description. A high-level summary of the approach is the following:

1. This is the essential training stage. A generative network is trained to generate instances of seen classes conditioned on the provided descriptions, usually with a WGAN or a VAE framework. A classifier can also be integrated to drive discriminative generation of features, i.e. it is not enough to fool a discriminator, a classifier must also classify them correctly.

2. Given the description of every class that is present during testing, the generator is used to generate synthetic features of the test classes, transforming the problem into a supervised one. A linear or a non-parametric classifier is trained on them.

3. This classifier is used for the final classification.

This basic approach has been modified and adjusted to improve its performance. Several works apply regularization techniques to avoid overfitting on seen classes and provide some sort of guarantee for the generalization of the features [46, 13]. Others perform more elaborate modifications [12, 16], but the process nonetheless remains the same. Some of these techniques are presented graphically in Figures 2.18 and 2.19.

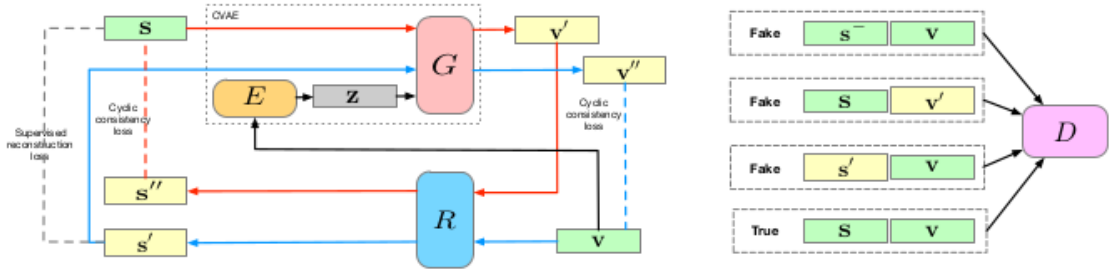


Figure 2.18: *GDAN uses a network that maps descriptions back to features to enforce cycle-consistency. Source: [12].*

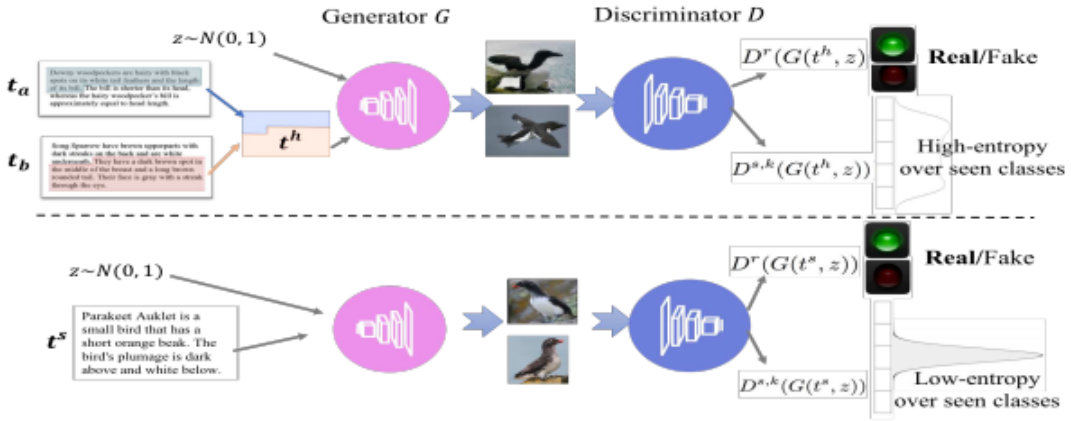


Figure 2.19: *CIZSL interpolates descriptions to create fake ones and tasks the generator with generating features that are plausible but hard to classify in one of the seen classes. Source: [13].*

Chapter 3

Reducing Zero-shot to Few-shot Learning

3.1 Introduction

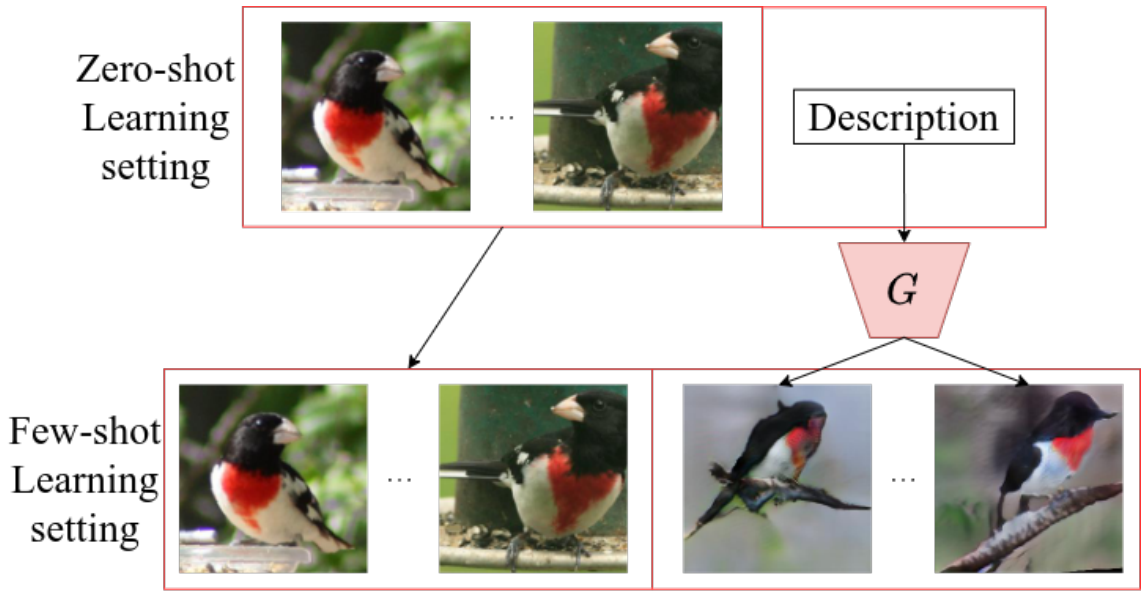


Figure 3.1: *Abstract representation of the conceptual reduction from Zero-shot Learning to Few-shot Learning.*

Deep Learning has seen great success in various settings, like Computer Vision [21, 22, 23], Speech and Language Processing [24, 25] or Computer Graphics [26, 27], and a plethora of different disciplines, such as Neuroscience [28, 29], Remote Sensing [30, 31] or Medical Science [8, 32]. Despite the variation of applications, there is a common denominator: resources. The overwhelming majority of contemporary applications not only benefit but necessitate voluminous data to realize their true potential, along with the associated hardware resources. This is evident both from their unprecedented, superhuman performance in many narrow tasks compared to traditional algorithms, e.g. [33, 34], and their surprising deficiencies in low-data regimes, as illustrated by Figure 1.4, that clearly demonstrates a large drop in performance where it even becomes inferior to traditional Machine Learning techniques. Combined with their demand for resources, whose environmental toll is only beginning to be scrutinized, with initial results revealing a disheartening reality [35], Deep Learning techniques become a worthwhile endeavor

only when large datasets are available.

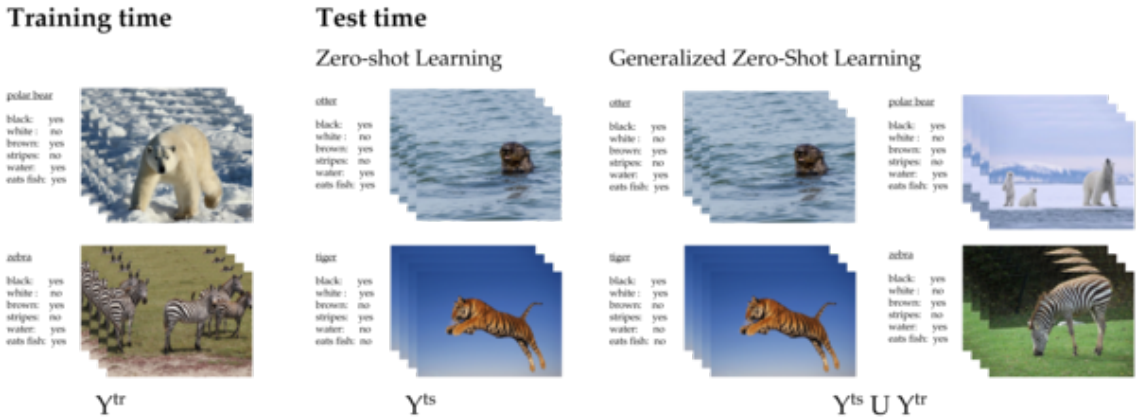


Figure 3.2: *Illustration of Zero-shot Learning, both regular and Generalized. During training, we are given images of several classes and their description. During testing, we are given descriptions and are tasked with matching them to images. In the Zero-shot Learning setting, test classes are all novel. In the Generalized Zero-shot Learning setting, test classes are mixed. Source: [14].*

Having identified this performance issue, in an effort to extend the merits of Deep Learning to the low-data regime, researchers have proposed and developed settings that task deep learning algorithms with learning from few data or at least adapting to new tasks or categories with a relatively small support. Moreover, over the last several years, the number of works addressing such problems has significantly increased. **Zero-shot** and **Few-shot** Learning tasks are such settings, which require adapting to novel classes. In the former, *no actual support* of these classes is given and *descriptions* are provided instead. In the latter, a *handful of supporting examples* per novel class are available. A possible scenario of the setting can be seen graphically in Figure 3.2. Note that "standard" Zero-shot Learning datasets and benchmarks exclusively consist of image classification tasks and the descriptions are provided in the form of class-level attribute vectors (each value in the vector denotes how much a property exists in the instances of the class. An example property for a dataset of bird species could be whether the birds of a species have a red body). Additionally, extracted image features are usually chosen over the images themselves for the sake of using low(er)-dimensional input data.

3.1.1 Related Work

To address Zero-shot Learning, early works split inference in two stages, usually inferring the attributes of an image and then assigning the image to the closest match from the given attributes. For example, DAP [36] estimates the posterior of every attribute of an image by learning probabilistic attribute classifiers and then predicts the label using the MAP estimate of the class posteriors. The algorithm in [37] operates similarly. On the other hand, IAP [36] predicts the class posterior of seen classes and then the class probabilities are used to calculate the attribute posteriors of any image. Instead of attributes, other methods opt for Word2Vec [75] as auxiliary data, such as CONSE [50].

More recent research focuses on a learning a mapping, mostly linear ones, from image feature space to a semantic space. ALE [38] attempts to match attributes and image features by learning a compatibility function that is a bilinear form with a pairwise ranking objective. DEVISE [39] also classifies images based on compatibility scores by learning a linear mapping between image features and text-label embeddings and also uses a pairwise ranking objective. SJE [40] utilizes a bilinear form as a compatibility function too, optimizing a structural SVM loss. ESZSL [41] too uses a bilinear form as a compatibility function. Using noisy text instead of attributes, [42] introduce a $l_{2,1}$ regularization term to the learning of the compatibility function. SAE [43] takes a step further and deploys an autoencoder to learn the mapping, though that still remains a linear one.

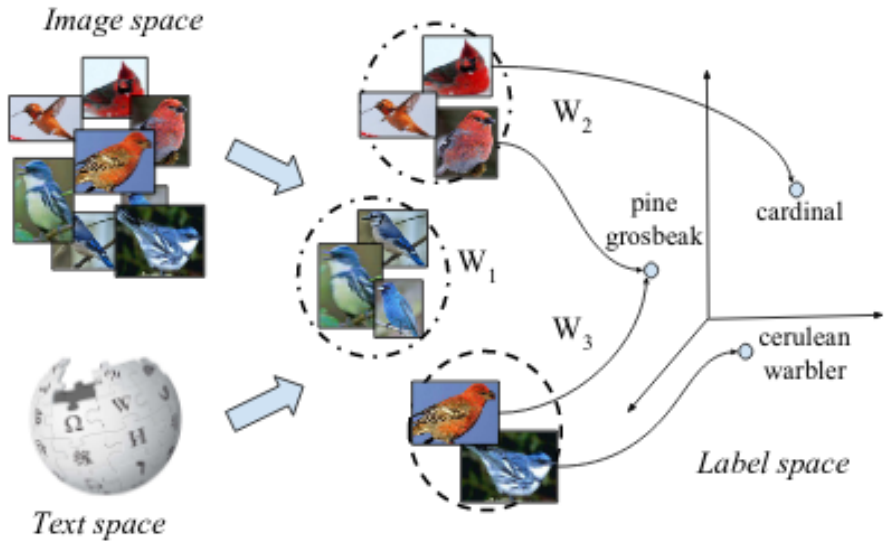


Figure 3.3: *LATEM learns a compatibility function between the input space (image and text) and the output space (labels). Source: [15].*

Fewer are the methods that learn non-linear mappings. The typical example for such an approach is LATEM [15], illustrated in Figure 3.3, which simply transforms the bilinear compatibility function of SJE [40] to be piece-wise linear by learning multiple linear mappings.

Another branch of recent works can be described as prototypical approaches, in the sense that, even though they still map to an intermediate space or from one principal space to the other, they focus on producing a prototype sample (an exemplar) per class. SYNC [44] and CVCZSL [45] are representative such methods. SYNC learns edge weights of a class graph in the semantic (description) space and postulates that this graph should be aligned with the one in the image feature space, calculating feature prototypes for every class in this manner. CVCZSL, on the other hand, learns a feedforward neural net that directly maps vector representations of the descriptions to image features, mapping the class description to a class prototype. Moreover, the network is trained in an episodic manner, i.e. the train setting is made to resemble the test setting (we provide more details in Section 3.3.2).

Generative Approaches

The culmination and evolution of the aforementioned efforts have promptly led to the current state-of-the-art approaches, generative approaches. The foundations of these approaches were largely laid out concurrently in [18, 11]. The basis of these approaches can be broken down as follows:

- *Stage I* (aka the *Generator Training Stage)*: This is the essential training stage. A generative network is trained to generate instances of seen classes conditioned on the provided descriptions. A classifier can also be integrated to drive discriminative generation.
- *Stage II* (aka the *Classifier Training Stage*): Given the description of every class that is present during testing, the generator is used to generate synthetic features of the test classes, transforming the problem into a supervised one. A linear or a non-parametric classifier is trained on them.
- *Stage III* (aka the *Test Stage*): The classifier is tasked with classifying the actual test images.

Note that the fact that the support is both synthetic and derived from novel classes acts as a deterrent to the usage of more complex classifiers, as learning the subtle peculiarities of the dataset is not only of no use but expected to deteriorate performance due to overfitting.

This basic setting has been extended to cope with the fact that plain generative training, like a mere GAN or VAE framework, does not offer any form of guarantee for the generalization of the generator, although results are sufficiently accurate. In fact, there are many works that present not only their framework, but also their results as an update compared to previous works instead of as their own method. One such work is CIZSL [13]. The authors use creative generation for recognition, i.e. generations based on synthetic descriptions are allowed to deviate just enough from seen classes so as to be both plausible and able to transfer knowledge from seen classes to novel ones. LisGAN [46] borrows from the prototypical approaches and uses prototypes (called exemplars in the paper), both for real and generated features, and forces the generated exemplars and the generated features as a whole to be close to the real ones. f-VAEGAN [16] imposes weight sharing between the decoder of a VAE and the generator of a WGAN in an attempt to leverage the best aspects of VAEs and GANs and create a more robust generator from their amalgamation.

As theses methods have become more mature and robust, larger deviations from the blueprint have started to emerge. GDAN [12] uses cycle consistency along with the authors addressing the potential of the discriminator being used as a compatibility function. E-PGN [76] trains the generator (note: generator is deterministic, i.e. it can be thought of as a prototypical approach using generative networks) in an episodic manner by constructing Zero-shot Learning tasks from the train splits. ZSML [47] uses Meta-Learning to train the generator along with the discriminator in the WGAN setting and the auxiliary

classifier. OCD [51] trains a VAE with an Over-complete distribution (interpolation of means produced from the VAE) to create more hard examples, which in turn render the synthetic dataset more informative.

To conclude, **the current state-of-the-art for Zero-shot Learning tasks utilizes generative networks to generate image features conditioned on auxiliary class descriptions. This way, the problem during evaluation is transformed into a supervised problem and simple classifiers are tasked with the final classification**.

3.2 Beyond Linear Classifiers

3.2.1 Intuition

After the introduction of a generative model in the process of learning, it was only natural to leverage the generator to produce a synthetic dataset to transform the problem during the actual inference into a supervised one. However, after that transformation, it is **not self-evident why a standard supervised setting is the most suitable setting** and, thus, why a classifier appropriate for that setting is the proper delegate for the final classification of the test images. Therefore, every work that follows the blueprint presented in [18, 11] has implicitly made an assumption, that of the appropriateness of the setting, which we intuitively and empirically challenge.

Simultaneously, as we note emphatically in previous sections, the ultimate reason Zero-shot Learning is now researched is to force the disentanglement of the enhanced performance of standard *Deep* Learning and large amounts of data. Nevertheless, contemporary Zero-shot Learning techniques fall back on traditional, simple Machine Learning techniques for the final classification. Although WGANs and/or VAEs are generally involved in the proposed algorithms, the aforementioned fact partly defeats the purpose of challenging Deep Learning algorithms with dealing with generalizing to novel classes with zero support.

3.2.2 Approach

In this thesis, we build and improve on generative solutions to Zero-shot Learning in a way that addresses both pivotal issues we have identified with previous work, the reliance on traditional Machine Learning models and the reduction to a standard supervised setting.

Namely, we use a **Few-shot Learner as a delegate during inference instead** of the aforementioned classifiers. More concretely, we alter the basic algorithm as follows.

- *Generator Training Stage*: We use a Few-shot learner as a classifier to train the generator alongside the generative framework.
- *Classifier Training Stage*: We (optionally) finetune the same Few-shot learner used in Generator Training Stage instead of training a new classifier.
- *Test Stage*: We use the aforementioned Few-shot learner for the final classification.

Stage	**Basic Generative Approach**	**Our Generative Approach**
Generator Training Stage	Train a Generator in a WGAN and/or VAE framework, optionally use linear classifier to backpropagate a classification loss	Train a Generator in a WGAN and/or a VAE framework while using a Few-shot learner to backpropagate a classification loss. Concurrently train that Few-shot learner.
Classifier Training Stage	Train a linear classifier on generated features based on necessary descriptions.	Finetune Few-shot learner on generated features of unseen classes.
Test Stage	Classify test image features with linear classifier	Classify test image features with Few-shot learner. Generator provides support set.

Table 3.1: *Comparison between the basic generative approach and our framework.*

We compare, stage-by-stage, our framework with the aforementioned basic generative approach in Table 3.1.

Our rationale for this modification is that a Few-shot learner:

- learns to classify at a setting that is intuitively more similar to Zero-shot Learning than standard classification setting is. That is because Few-shot learners are tasked with learning to learn instead of simply learning. Indicatively, some of them are also *Meta-learners*. This allows adaptation to novel tasks or the making of appropriate adjustments to accommodate the current task.

- can be categorized more convincingly as a Deep Learning technique (although that depends on the specific learner), thus not eschewing the main hindrance of a Zero-shot Learning task.

We can understand why this method checks all the qualitative boxes we discussed. By virtue of the experimental results, we demonstrate that it quantitatively checks them as well. The gains in accuracy can be intuitively attributed to the fact that the Few-shot learner, as a classifier, does not have a set label space, i.e. is tasked with matching queries to support rather than a specific label. Consequently, it can be trained during training, along with the generator, enabling it to increase generalization through their interaction, in contrast to the classifiers deployed in other methods which are introduced after the training of the generator is done, in the Classifier Training Stage. This fact gives one access to a sophisticated classifier come test time, which, almost by definition, has been trained not to overfit but to adapt. In addition, in a realistic setting, previous methods would necessitate training from scratch with each new arrival, whereas, as we demonstrate, the Few-shot learner can be used as-is without any finetuning. Next, we formally formulate our simple framework.

3.3 Framework Formulation

3.3.1 Problem Definition

Let $\mathcal{X}_S \subseteq \mathbb{R}^{d_f}$ be the train image feature space and $\mathcal{X}_U \subseteq \mathbb{R}^{d_f}$ be the test image feature space, where d_f is the dimensionality of the features. Let $\mathcal{Y}_S$ be the seen label space and $\mathcal{Y}_U$ be the unseen label space, subject to $\mathcal{Y}_S \cap \mathcal{Y}_U = \emptyset$. Finally, let $\mathcal{A}_S \subseteq \mathbb{R}^{d_a}$ and $\mathcal{A}_U \subseteq \mathbb{R}^{d_a}$ be the train and test vector representation space of the descriptions respectively, where d_a is the dimension of the description vectors. The problems we examine contain class-level descriptions, so we can alternatively use the mappings $a_X : \mathcal{Y}_X \to \mathcal{A}_X,\ X \in \{S, U\}$, which is a bijection. Having defined the necessary spaces, we denote the seen-classes (train) dataset as the tuples in $D_S = \{(x_i, y_i) \mid x_i \in \mathcal{X}_S, y_i \in \mathcal{Y}_S\}$. The unseen-classes (test) dataset, although a definition is unnecessary for our purposes, can be defined similarly as $D_U = \{(x_i, y_i) \mid x_i \in \mathcal{X}_U, y_i \in \mathcal{Y}_U\}$.

Definition 3.1 (Zero-shot Learning)**.** *In this setting, we are given D_S in conjunction with a_S, and $\mathcal{A}_U$ training and, during testing, we must classify every test feature in D_U based on descriptions in $\mathcal{A}_U$.*

Definition 3.2 (Generalized Zero-shot Learning)**.** *In this setting, we are given a set $D'_S \subset D_S$ in conjunction with a_S, and $\mathcal{A}_U$ for training. For every class in D_S, there is at least one sample of the class in D'_S and at least one in $D_S \setminus D'_S$ (the latter restriction is not mandatory for the setting to be coherent, however it is so in every examined benchmark). During testing, we must classify every test feature (from unseen classes) in D_U and every test feature (from seen classes) in $D_S \setminus D'_S$ given the descriptions of every class, both $\mathcal{A}_U$ and $\mathcal{A}_S$.*

Note that the aforementioned tasks solely describe the *inductive* setting of Zero-shot Learning and Generalized Zero-shot Learning. An extension of it is the *transductive* setting, which involves having access to all images, meaning both $\mathcal{X}_S$ and $\mathcal{X}_U$, during training, with the labels and descriptions available remaining precisely as described above. We do not examine this setting because it is both unrealistic and counter to the goals of Zero-shot Learning. Additionally, we refer to the jargon as if we were discussing Zero-shot Learning, but the Generalized Zero-shot Learning formulation can be easily inferred.

Definition 3.3 (Few-shot Learning)**.** *In this setting, we are given D_S for training and, during testing, we are iteratively given a small subset $\mathcal{S}$ of D_U, named support set, which includes image features from n_{way} classes with n_{shot} features each. S_k denotes the set of samples of class k. Based on this set, we are tasked with classifying images of these n_{way} classes, again sampled from D_U, which belong to another set called query set. Depending on the values of n_{way} and n_{shot}, the particular Few-shot Learning setting is called "n_{way}-way n_{shot}-shot" (e.g. 10-way 4-shot). Of course, we can substitute D_S with D'_S and D_U with $D_U \cup (D_S \setminus D'_S)$ in the definition.*

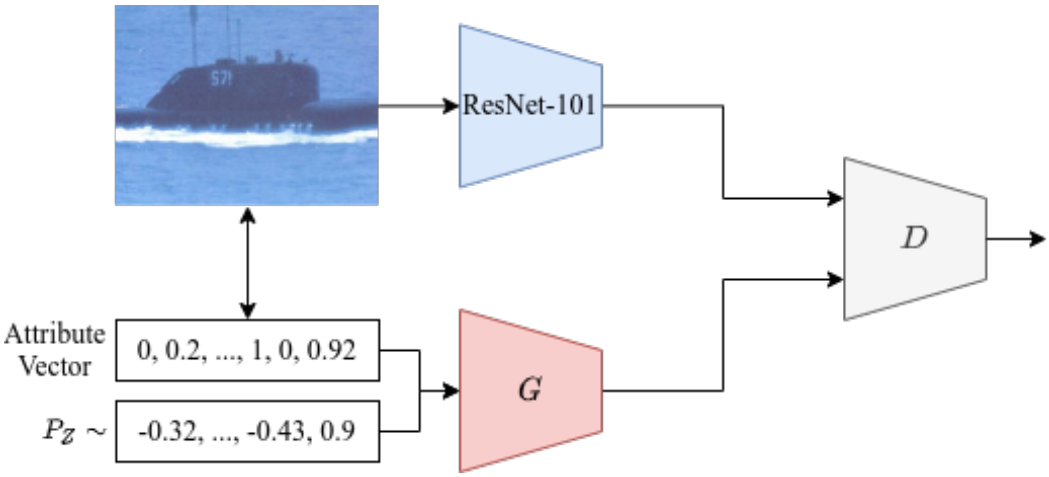


Figure 3.4: *Graphical representation of conditional WGAN for image features.*

3.3.2 Background

WGAN

Generative Adversarial Networks (GAN) were introduced in [65]. In short, it is a framework to train a stochastic generator G, with a noise vector as input, where the generator is tasked with confusing a discriminator, which is a neural network tasked with differentiating between the real distribution and the one produced by the generator. The networks are trained in a minimax fashion. In [66], the authors introduce the Wasserstein GAN (WGAN), which improves the training of the GANs by changing the function of the minimax game into the Wasserstein distance that, however, constraints the discriminator (referred to as critic in the paper) to be a Lipschitz function, which is initially achieved by weight clipping. The weight clipping is swapped for a regularization term in [67], which is the final formulation we utilize in this work. Note that we augment the formulation in [67] to include the description as an extra input to the noise, also called conditioning variable. Namely, we define

$$\begin{aligned}\mathcal{L}_{WGAN}(G, D; P_R, P_G, P_Z) = \mathbb{E}_{(x,a)\sim P_R}[D(x,a)] - \mathbb{E}_{(\tilde{x},a)\sim P_G|P_Z}[D(\tilde{x},a)] \\ - \lambda \cdot \mathbb{E}_{(\hat{x},a)\sim P_{\hat{X}}|P_Z}[(\|\nabla_{\hat{x}} D(\hat{x},a)\|_2 - 1)^2],\end{aligned} \tag{3.1}$$

where G is the generator, D the discriminator, P_R the distribution of the real data (e.g. random sampling from D_S), P_Z is a noise distribution, usually chosen to be the standard normal or uniform distribution, $P_G|P_Z$ is the distribution of the generator and the input descriptions where the noise input is sampled from P_Z, $P_{\hat{X}}|P_Z$ is the joint distribution of input description used for the generator output for convenience and the uniform distribution on the line between $x \sim P_R$ and $\tilde{x} \sim P_G|P_Z$ (intuitively $\hat{x} = ux + (1-u)\tilde{x}, u \sim U(0,1)$), and λ is a hyperparameter. We use the distributions both as joint distributions and the marginals interchangeably to avoid further mathematical jargon. The minimax "game" is defined as:

$$\min_G \max_D \mathcal{L}_{WGAN}(G, D; P_R, P_G, P_Z). \tag{3.2}$$

VAE

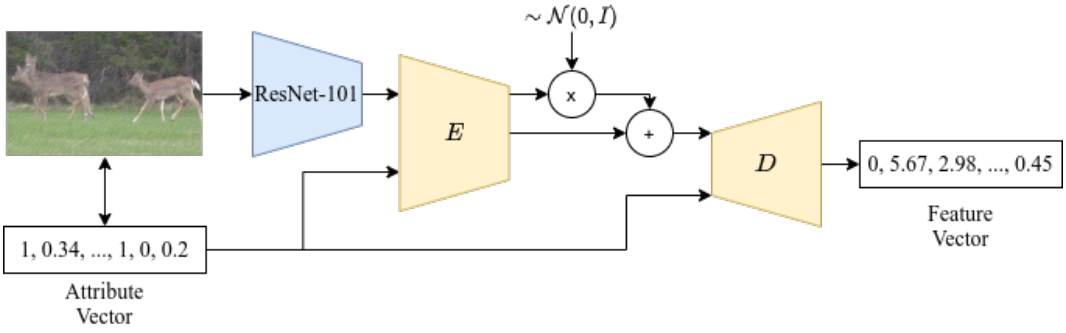


Figure 3.5: *Graphical representation of conditional VAE for image features.*

The Variational Autoencoder (VAE), introduced in [69], is an autoencoder that maximizes the variational lower bound of the marginal likelihood. An autoencoder consists of an encoder network, which compresses its input to a "latent" variable, in a low-dimensional space, and a decoder network that receives the output of the encoder and essays to reconstruct the original input of the encoder. We present the mathematical formulation of the Conditional VAE (CVAE), namely

$$\mathcal{L}_{VAE}(E, D; P_R, P_\partial) = \mathcal{L}_{REC}(D, E; P_R) + \mathbb{E}_{(x,a)\sim P_R}[D_{KL}(p_E(z|x, a)||p_\partial(z))], \tag{3.3}$$

where E is the encoder of the VAE, D its decoder, D_{KL} is the Kullback-Leibler divergence, $\mathcal{L}_{REC}$ is the reconstruction loss and, as in [69], it is usually the binary cross-entropy loss (BCE)

$$\mathcal{L}_{REC}(D, E; P_R) = -\mathbb{E}_{(x,a)\sim P_R}[x \cdot \log D(E(x, a), a) + (1 - x) \log (1 - D(E(x, a), a))],$$

P_R the distribution of the real data, p_E is to be understood as the output distribution of the encoder, which is a Gaussian MLP, i.e. outputs the mean and the variance of a Gaussian distribution so the Kullback-Leibler divergence can be computed analytically, and p_∂ is the prior distribution of the latent variable. For practical purposes, this prior is set to the standard normal distribution and the reparameterization trick [77] is used, meaning we do not sample directly from the distribution the encoder produces, but we compute the latent variable as $\mu(x) + \epsilon\sigma(x)$, $\epsilon \sim \mathcal{N}(0, I)$.

f-VAEGAN

f-VAEGAN [16] is a generative approach to Zero-shot Learning that deploys both a WGAN and a VAE to train the generator. This is achieved by means of sharing the

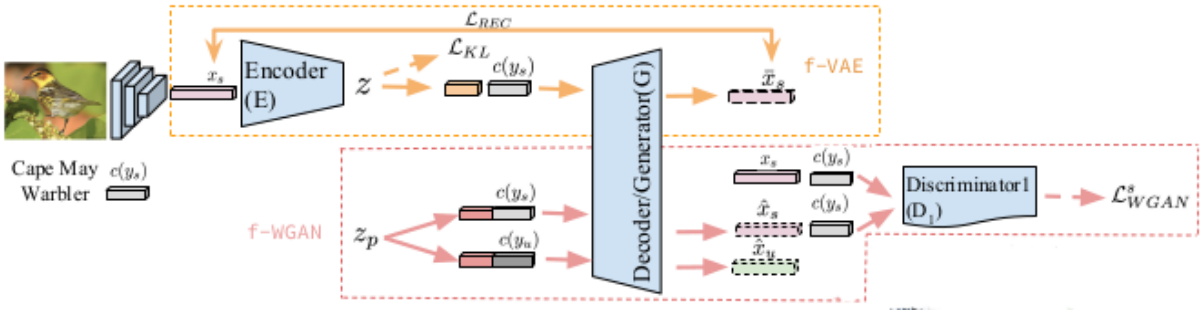


Figure 3.6: *Graphical representation of conditional f-VAEGAN. Source: [16].*

weights of the VAE's decoder and the WGAN's generator. The overall loss function of the approach is

$$\mathcal{L}_{VAEGAN}(G, E, D; P_R, P_G, P_Z, \beta) = \mathcal{L}_{VAE}(E, G; P_R, P_Z) + \beta \cdot \mathcal{L}_{WGAN}(G, D; P_R, P_G, P_Z), \quad (3.4)$$

where E is the encoder of the VAE, D the discriminator of the WGAN, G the generator of the WGAN and the decoder of the VAE as described above, and β a hyperparameter controlling the relative importance of the the WGAN loss. The authors implement $\mathcal{L}_{REC}$ as the BCE. Note that the framework can be extended for transductive Zero-shot Learning, whose formulation we skip since it is of no use to us.

Prototypical Networks

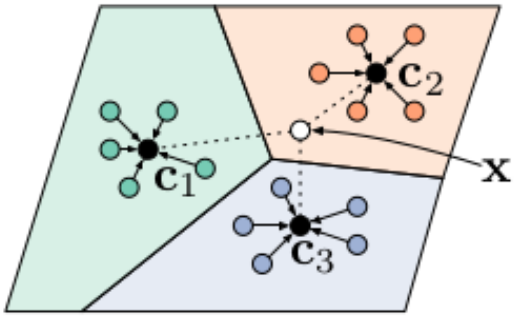


Figure 3.7: *Graphical representation of the output space of a Prototypical Network. Source: [17].*

Prototypical Networks (PN) [17] present a simple framework for Few-shot Learning. A neural network, f_ϕ, is used to map the input image features to a metric space. Their methodology is to map the support set and average their embeddings per class so as to get a prototype for each class. Thereafter, each sample in the query set is mapped to the metric space and classified to the nearest prototype based on a distance metric d. The formulation is

$$
\begin{aligned}
c_k &= \frac{1}{|S_k|} \sum_{x_i \in S_k} f_\phi(x_i), \\
p_\phi(y = k|x) &= \frac{\exp\left(-d(f_\phi(x), c_k)\right)}{\sum_{k'} \exp\left(-d(f_\phi(x), c_{k'})\right)}, \\
\mathcal{L}_{PN}(f_\phi; S, Q) &= \frac{1}{|Q|} \sum_{(x_i, y_i) \in Q} \log p_\phi(y = y_i|x),
\end{aligned}
\tag{3.5}
$$

where $p_\phi(y = k|x)$ if the probability of feature x belonging to class k, and $\mathcal{L}_{PN}$ is the final cross-entropy loss. The authors use the Euclidean distance, as it renders the framework a Mixture Density Model. Instead of using the standard supervised setting during training, the network is trained in an *episodic* manner. That is to say, the train setting is made to resemble the test setting. In other words, instead of sampling batches from D_S, we sample episodes, also referred to as minibatches, i.e. we sample $n_{shot} + n_{query}$ features from each of n_{way} random classes, n_{shot} of course for the support set and the rest for the query set.

3.3.3 Z2FSL Framework

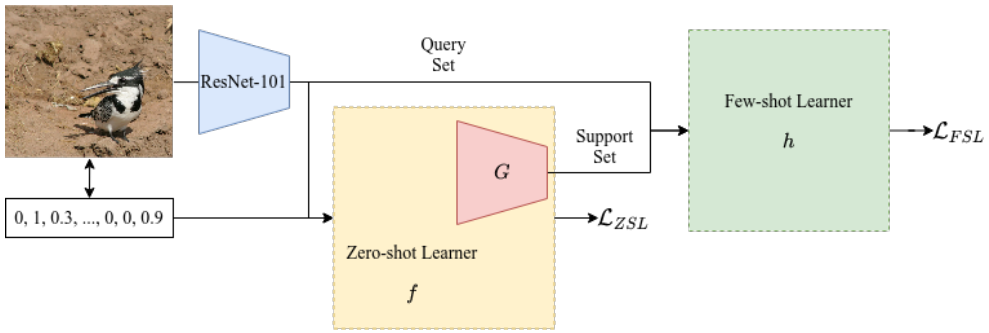


Figure 3.8: *Graphical representation of our Z2FSL framework.*

In this work, we leverage Few-shot Learning as a delegate for the final classification of test images after the synthetic dataset has been created, essentially reducing Zero-shot Learning to Few-shot Learning. As a result, for convenience, we refer to our method as **Z2FSL**. Since the classifier can be available during training, we also make use of it in that setting by generating its support, while the queries naturally come from the train images. In the interest on presenting a simple yet effective framework that can be integrated with any Zero-shot Learning method, we formulate our approach as multitasking for the generator of an already existing Zero-shot Learning approach f in conjunction with also an existing Few-shot Learning approach h, as presented graphically in Figure 3.8. Since our framework is agnostic to the Zero-shot and the Few-shot learner, we skip the technicalities that depend on these choices, and present the framework in a very high-level manner. We denote the loss of the Zero-shot learner as $\mathcal{L}_{ZSL}$ and the loss of the Few-shot learner as $\mathcal{L}_{FSL}$. Therefore, the overall loss of our framework is simply

ALGORITHM 3.1: *Z2FSL (f-VAEGAN, Prototypical Network) training procedure.*

Input: a_f: Zero-shot learner's learning rate, ∂_h: pretrained Few-shot learner, a_h: Few-shot learner's learning rate, β: WGAN loss' coefficient, γ: Few-shot learner loss' coefficient, n_{way}: training episode's way, n_{query}: training episode's queries per class, n_{shot}: training episode's generations per class, N: number of training episodes, D_S: training dataset, n_{critic}: number of discriminator steps per generator step

Output: The generator's parameters, ∂_g

```
Initialize VAE's encoder parameters ∂_e
Initialize generator's parameters ∂_g
Initialize discriminator's parameters ∂_d
P_Z ← N(0, I)
for i=1,...,N do
    for j=1,...,n_critic + 2 do
        Q, A ← episode(D_S, n_way, n_query)
        if j ≤ n_critic then
            # Discriminator training
            S ← generate(∂_g, A, P_Z, n_query)
            L_WGAN ← −L_WGAN(∂_g, ∂_d; (Q, A), (S, A), P_Z)
            ∂_d ← Adam(∂_d, a_f, L_WGAN)
        else if j == n_critic + 1 then
            # Few-shot learner training
            S ← generate(∂_g, A, P_Z, n_shot)
            L_PN ← L_PN(∂_h; S, Q)
            ∂_h ← Adam(∂_h, a_h, L_PN)
        else
            # Generator training
            S ← generate(∂_g, A, P_Z, n_shot)
            L_ZSL ← L_VAEGAN(∂_g, ∂_e, ∂_d; (Q, A), (S, A), P_Z, β)
            L_FSL ← L_PN(∂_h; S, Q)
            L_Z2FSL ← L_ZSL + γ · L_FSL
            ∂_g ← Adam(∂_g, a_f, L_Z2FSL)
        end if
    end for
end for
```

$$\mathcal{L}_{Z2FSL} = \mathcal{L}_{ZSL} + \gamma \cdot \mathcal{L}_{FSL}. \tag{3.6}$$

where γ is a hyperparameter. We skip the arguments in the losses as there is room for deviation depending on the implemented methods.

During the original Classifier Training stage, we instead perform an almost naive finetuning of the Few-shot learner, by simply generating both the support and the queries based on unseen descriptions.

To demonstrate the efficacy of our approach, we implement f as the f-VAEGAN [16] and h as a Prototypical Network [17]. For convenience, we refer to our method by a macro which we simply define as *Z2FSL*(f, h): f is used as the Zero-shot learner and h as the

ALGORITHM 3.2: *Z2FSL test finetuning procedure.*

Input: ∂_h: Few-shot learner, a_h: Few-shot learner's learning rate, n_{query}: training episode's queries per class, n_{shot}: training episode's generations per class, N_h: number of training episodes, D_U: test dataset of unseen classes
Output: The Few-shot learner's parameters, ∂_h

$P_Z \leftarrow \mathcal{N}(0, I)$
for i=1,. . . ,N_h **do**
  $A \leftarrow episode(D_U, all, none)$
  $\mathcal{Q} \leftarrow generate(\partial_g, A, P_Z, n_{query})$
  $S \leftarrow generate(\partial_g, A, P_Z, n_{shot})$
  $L_{PN} \leftarrow \mathcal{L}_{PN}(\partial_h; S, \mathcal{Q})$
  $\partial_h \leftarrow Adam(\partial_h, a_h, L_{PN})$
end for

Few-shot Learner. We use the approach's name and its citation interchangeably in the arguments.

The training procedure is summed up in Algorithm 3.1. While the finetuning is straightforward, we also present the procedure in Algorithm 3.2 to introduce important notation.

First, we pretrain a Few-shot learner on the training dataset D_S (can be considered as the zeroth stage). Afterwards, we iteratively sample episodes from D_S for training in the Generator Training Stage, which other than the image features also include the corresponding descriptions. We follow [65, 66, 67] in training the discriminator multiple steps, denoted n_{critic} in the algorithm, before training the generator once. The episode count is measured relative to the generator steps, i.e. we consider the multiple steps part of one episode, although we sample from D_S multiple times for the discriminator. We also finetune the Few-shot learner during the generator training, where its support is generated with the generator. We elect for the training step of the Few-shot learner to take place before the one of the generator instead of training them concurrently to avoid overfitting between the networks. Although the Few-shot learner step is independent from the one of the discriminator, it affects the training of the generator (it is used to compute L_{FSL}) and vice versa. Nevertheless, the order of training between the generator and the Few-shot learner was selected arbitrarily. Note that it is not imperative to train / finetune the Few-shot learner (can be achieved by setting the learning rate $a_h = 0$).

During testing, the generator is used to provide the support set based on the test descriptions. However, there is no need to keep the number of samples per class, n_{shot}, the same as the one during training. We then classify the test images serving as the queries in the Few-shot Learning setting.

3.4 Experiments

The proposed framework is evaluated on both the Zero-shot Learning and the Generalized Zero-shot Learning settings and compared to prominent or recent state-of-the-art approaches. First, we present the datasets and their statistics, evaluation metrics, the

Dataset	$\lvert D_S\rvert + \lvert D_U\rvert$	$\lvert D_S\rvert$	$\lvert D'_S\rvert$	$\lvert D_U\rvert$	$\lvert \mathcal{Y}_S\rvert + \lvert \mathcal{Y}_U\rvert$	$\lvert \mathcal{Y}_S\rvert$	$\lvert \mathcal{Y}_U\rvert$	d_a
CUB	11788	8855	7057	2933	200	150	50	312
AwA2	37322	30337	23527	6985	50	40	10	85
SUN	14340	12900	10320	1440	717	645	72	102

Table 3.2: *Dataset overview of CUB [19], AwA2 [14] and SUN [20] in terms of size, number of classes and dimension of available attributes. The statistics presented are, by column: dataset size, training set size, training set size in the Generalized Zero-shot Learning setting, test set size, number of classes, number of seen classes, number of unseen classes, and dimension of attributes.*

implementation details afterwards and finally our results and ablation studies.

3.4.1 Datasets

We use the Caltech UCSD Bird 200 (CUB) [19] which consists of 11788 images of birds belonging to 200 species (classes). Its auxiliary descriptions are class-level 312-dimensional semantic attribute vectors. We also test our framework on Animals with Attributes 2 (AwA2) [14], which is the "sequel" to Animals with Attibutes (AwA) [36] but without copyright licenses to its images. It contains 37322 images from 50 categories and 85-dimensional per-class attributes. Our last dataset is the SUN Scene Classification (SUN) [20] dataset, with 14340 images of 717 categories and 102-dimensional attributes for each class.

As already noted, we use attributes as the class-level descriptions, instead of other alternatives like sentence embeddings from [64]. In particular, we use the continuous-valued attributes, on which we perform L2-normalization, instead of the binary ones, as suggested in [14]. We extract image features from the datasets using the ResNet-101 [49] up to its second to last layer, the adaptive average pooling, pretrained on ImageNet [48]. This renders $d_f = 2048$. While extracting from the images, we use 10 crops per image (center, top-right, top-left, bottom-right, bottom-left and again with a horizontal flip) as augmentation. We use the central crop without the flip for testing. The central crops are basically the original images without any preprocessing, while the other crops extent from their respective corners to 80% of each dimension and then resized to match the original image's dimensions. We also use and report from the train-test and seen-unseen splits proposed in [14] (referred to as "Proposed Splits" contrary to the "Standard Splits" used in many previous works), which were established to avoid overlap of unseen classes and ImageNet classes used to pretrain the feature extractor. A comprehensive overview of the information for each benchmark is presented in Table 3.2.

3.4.2 Evaluation Metrics

We evaluate our framework with the widely accepted and used metrics for the problems we examine. For standard Zero-shot Learning, we use the *average per-class* [top-1] *accuracy*, defined as

	Zero-shot Learning			Generalized Zero-shot Learning		
Hyperparameter	CUB	AwA2	SUN	CUB	AwA2	SUN
a_h	$5 \cdot 10^{-5}$	10^{-5}	10^{-5}	10^{-3}	10^{-5}	$5 \cdot 10^{-5}$
N_h	12000	15000	10000	10000	12000	8000
n_{hidden}	0	1	1	0	1	1
n_{way}	25	10	40	25	10	50
n_{shot}	5	5	5	5	5	5
n_{query}	10	15	5	10	15	2

Table 3.3: *Hyperparameter configuration per setting and dataset for the Prototypical Network's pretraining. From top to bottom, we have the learning rate of the learner, the number of episodes, the number of hidden layers, the number of classes in an episode, the number of support examples per class and the number of queries per class.*

$$acc_{\mathcal{Y}} = \frac{1}{\|\mathcal{Y}\|} \sum_{y \in \mathcal{Y}} \frac{\text{\# correct predictions in y}}{\text{\# samples in y}}, \tag{3.7}$$

where in the case of Zero-shot Learning $\mathcal{Y} = \mathcal{Y}_U$. This metric essentially applies a weight to each class that can be thought of as a monotonically decreasing function of the class cardinality, so it is in effect used to balance the metric of an unbalanced dataset. For Generalized Zero-shot Learning setting, we instead use the *harmonic mean* of the average per-class accuracy of seen classes and that of unseen classes, which is summarized in its definition as

$$H = 2\frac{u \cdot s}{u + s}, \tag{3.8}$$

where we define $u = acc_{\mathcal{Y}_U}$ and $s = acc_{\mathcal{Y}_S}$ for convenience and adherence to establisted notation. The purpose of this metric is to further balance seen and unseen classes' accuracies in a way that punishes the neglect of either seen or unseen classes by the classifier. This is essential for the task as its premise was the additional difficulty in alleviating the bias for seen classes, which has already been discussed. For instance, in the extreme case where unseen classes are completely ignored, it is easy to see that H is equal to 0, which is of course the lowest possible score.

3.4.3 Implementation Details

We pretrain the Few-shot learner, a Prototypical Network, as suggested in the original paper [17], i.e. in an episodic manner for N_h episodes and present its hyperparameters in Table 3.3. The network of the Prototypical Network is implemented as a feedforward neural network with n_{hidden} hidden layers, with ReLU activations. Note that we initialize the weigths of the Prototypical Network by setting the diagonal elements of the weight matrices to 1 and all other elements are initialized randomly as usual, resulting in square

	Zero-shot Learning			**Generalized Zero-shot Learning**		
Hyperparameter	CUB	AwA2	SUN	CUB	AwA2	SUN
a_f	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
finetune h	True	True	True	False	True	False
β	1	1	1	10	10	1
γ	1	1	1	10	1	1
n_{way}	50	20	80	50	20	80
n_{shot}	5	5	5	5	5	5
$n_{S,shot}$	-	-	-	5	5	5
n_{query}	10	20	5	10	20	5
N	6000	8000	5000	7000	6000	10000

Table 3.4: *Hyperparameter configuration per setting and dataset for our main experiments. We have, from top to bottom, the learning rate of the Zero-shot Learner, whether we elect to finetune the Few-shot learner, the coefficient of the WGAN loss, the coefficient of the Few-shot Learning loss, the number of classes in a training episode, the number of generations per class in a training episode, the number of generations per seen class during testing (recall that for unseen classes it is kept constant at 500), the number of image features per class in a training episode and the number of episodes.*

weight matrices and, in turn, in the input dimension dictating the dimension of all the layers, i.e. restricting it to be identical. The Few-shot learner's learning rate is kept the same during the Generator Training Stage (stage I) and the Classifier Training Stage (stage II) as it was during its pretraining.

As suggested in [66, 67], we set the coefficient of the regularization term of the WGAN $\lambda = 10$ and the training steps of the discriminator $n_{critic} = 5$. The discriminators are also feedforward neural networks with one hidden layer of 4096 neurons, Leaky ReLU hidden activation (0.2 negative slope) and linear output activation.

The generator and the encoder are too feedforward neural networks with 2 hidden layers each, 4096-8192 neurons for the generator and the reverse for the encoder, Leaky ReLU hidden activations (0.2 negative slope), linear output activation for the encoder and a sigmoid activation for the generator. The noise dimension is simply chosen to be the same as the dimension of the attributes, d_a.

We perform minmax normalization on the features (based on the relevant train split) and project them to $[0, 1]$ (essential for the BCE loss and the sigmoid activation). We set n_{shot} during testing (arbitrarily) to 500 for unseen classes, whereas we search for the respective value for seen classes, $n_{S,shot}$, in the Generalized Zero-shot Learning setting. During the CLassifier Training Stage (stage II), we always set the number of training episodes $N_h = 25$, and n_{shot} and n_{query} are kept the same as in normal training. The optimizer used in all cases is Adam [78] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We apply gradient clipping with a threshold of 5 per value. We implement everything in Pytorch [79].

We present the hyperparameters of our search, in Table 3.4, where, from top to bottom, we present the learning rate of the Zero-shot Learner a_f, whether we elect to finetune the Few-shot learner, the coefficient of the WGAN loss β, the coefficient of the Few-

	Zero-shot Learning			Generalized Zero-shot Learning								
	CUB	AwA2	SUN	CUB			AwA2			SUN		
Method	**aT1**	**aT1**	**aT1**	**u**	**s**	**H**	**u**	**s**	**H**	**u**	**s**	**H**
CONSE [50]	34.3	44.5	38.8	1.6	72.2	3.1	0.5	90.6	1.0	6.8	39.9	11.6
SAE [43]	33.3	54.1	40.3	7.8	54.0	13.6	1.1	82.2	2.2	8.8	18.0	11.8
LATEM [15]	49.3	55.8	55.3	15.2	57.3	24.0	11.5	77.3	20.0	14.7	28.8	19.5
SJE [40]	53.9	61.9	53.7	23.5	59.2	33.6	8.0	73.9	14.4	14.7	30.5	19.8
ESZSL [41]	53.9	58.6	54.5	12.8	63.8	21.0	5.9	77.8	11.0	11.0	27.9	15.8
SYNC [44]	55.6	46.6	56.3	11.5	70.9	19.8	10.0	90.5	18.0	7.9	43.3	13.4
DEVISE [39]	52.0	59.7	56.5	23.8	53.0	32.8	17.1	74.7	27.8	16.9	27.4	20.9
ALE [38]	54.9	62.5	58.1	23.7	62.8	34.4	14.0	81.8	23.9	21.8	33.1	26.3
CVCZSL [45]	54.4	71.1	62.6	47.4	47.6	47.5	56.4	81.4	**66.7**	36.3	42.8	39.3
f-CLSWGAN [18]	57.3	-	60.8	43.7	57.7	49.7	-	-	-	42.6	36.6	39.4
LisGAN [46]	58.8	-	61.7	46.5	57.9	51.6	-	-	-	42.9	37.8	40.2
f-VAEGAN [16]	61.0	-	64.7	**48.4**	60.1	**53.6**	-	-	-	**45.1**	38.0	41.3
GDAN [12]	-	-	-	39.3	**66.7**	49.5	32.1	67.5	43.5	38.1	**89.9**	**53.4**
OCD [51]	60.3	**71.3**	63.5	44.8	59.9	51.3	**59.5**	73.4	65.7	44.8	42.9	43.8
Z2FSL(VAEGAN, PN)	**62.0**	63.2	**66.5**	42.6	58.7	49.4	50.5	**81.8**	62.4	42.1	33.3	37.2

Table 3.5: *Comparison of our approach Z2FSL(VAEGAN, PN) (using f-VAEGAN [16] and Prototypical Networks [17]) to previous work.* ***aT1*** *denotes the average top-1 accuracy, defined in Equation 3.7, and* ***u****,* ***s*** *and* ***H*** *denote unseen class accuracy, seen class accuracy and their harmonic mean respectively, as defined in Equation 3.8.* ***Bold*** *indicates state-of-the-art accuracy, red denotes second best. For Generalized Zero-shot Learning, we only consider viable the methods that achieve a decent* ***u*** *and thus* ***H****, indicated by the bar.*

shot Learning loss γ, the number of classes in a training episode n_{way}, the number of generations per class in a training episode n_{shot}, the number of generations per seen class during testing $n_{S,shot}$ (recall that for unseen classes it is kept constant at 500), the number of image features per class in a training episode n_{query} and the number of episodes N.

3.4.4 Comparison with State-of-the-Art

We present our results in Table 3.5. We report the median result of at least 5 attempts w.r.t. average per-class accuracy for Zero-shot Learning and harmonic mean for Generalized Zero-shot Learning. We use an aggregation function in order to present a more robust metric and refrain from using the mean of our results due to the harmonic mean not being a linear operator of the average per-class accuracies. To illustrate this in a simple example, suppose s resulted in 0 in the first run and 40 in the second, and vice versa for u, and therefore the harmonic mean is in both cases 0. Then, it is easy to see that even though the mean harmonic mean is 0, the mean s and u is 20, which of course do not result into $H = 0$.

Our approach achieves state-of-the-art performance in 2 out of the 6 setting-benchmark combinations, CUB and SUN in Zero-shot Learning, while also getting the top "seen" accuracy in AwA2 for Generalized Zero-shot Learning. We can generally see that the framework struggles with Generalized Zero-shot Learning, although it achieves sufficiently good

	SUN
Method	**aT1**
f-VAEGAN [16] (original)	64.7
f-VAEGAN [16] (our impl.)	60.1
f-VAEGAN [16] (our impl., w/o aug.)	61.1
Z2FSL(f-VAEGAN, PN) (w/ softmax)	62.8
Z2FSL(f-VAEGAN, PN) ($\gamma = 0$, no finetuning)	64.3
Z2FSL(f-VAEGAN, PN)	66.5

Table 3.6: *Comparison between the reported results of f-VAEGAN, our implementation of f-VAEGAN, our framework with f-VAEGAN and Prototypical Network [17] but without the Prototypical Network during testing, our framework with f-VAEGAN and Prototypical Network but without using (aka $\gamma = 0$ in Equation 3.6) nor finetuning the Prototypical Network during the Generator Training Stage, and our framework's actual results.* ***aT1*** *denotes the average top-1 accuracy, defined in Equation 3.7.*

	CUB	AwA2	SUN
Method	**aT1**	**aT1**	**aT1**
f-VAEGAN [16] (original)	61.0	-	64.7
f-VAEGAN [16] (our impl.)	53.4	61.6	60.1
Z2FSL(f-VAEGAN, PN)	62.0	63.2	66.5
Absolute Performance Gain (original)	+1.0	-	+1.8
Relative Performance Gain (original)	+1.6	-	+2.8
Absolute Performance Gain (our impl.)	+8.6	+1.6	+6.4
Relative Performance Gain (our impl.)	+16.1	+2.6	+10.6

Table 3.7: *Comparison between the reported results of f-VAEGAN (the plain Zero-shot learner), our implementation of it and our framework using it. We present the absolute and relative gains in accuracy of our framework first compared to the reported results and then compared to our implementation.* ***aT1*** *denotes the average top-1 accuracy, defined in Equation 3.7.*

results in AwA2 in that setting.

3.4.5 Ablation Studies

In order to establish that the Few-shot learner is the source of the improvement and further examine the benefits of our framework, we design and perform several additional experiments.

Comparison with Linear Classifier

To demonstrate the actual effects of the Few-shot learner in our framework, we compare our framework with the original results of the plain Zero-shot learner, our implementation of that Zero-shot learner, our framework but without the Few-shot learner during training or during testing and, of course, our proposed framework.

The results are presented in Table 3.6. We see that our implementation of f-VAEGAN,

	SUN				AwA2			
Method	**aT1**	**u**	**s**	**H**	**aT1**	**u**	**s**	**H**
Z2FSL(None, PN)	65.3	36.0	**36.7**	36.4	63.0	46.3	69.2	55.5
Z2FSL(f-VAEGAN, PN)	**66.5**	**42.1**	33.3	**37.2**	**63.2**	**50.5**	**81.8**	**62.4**

Table 3.8: *Comparison between our approach using no Zero-shot learner (denoted by None, can be though of as clamping the Zero-shot Learning loss $\mathcal{L}_{ZSL}$ to 0) and using f-VAEGAN [16]. Results for the latter (top row) presented without finetuning during the Classifier Training Stage, because it significantly deteriorates performance.* ***aT1*** *denotes the average top-1 accuracy, defined in Equation 3.7, and* ***u***, ***s*** *and* ***H*** *denote unseen class accuracy, seen class accuracy and their harmonic mean respectively, as defined in Equation 3.8.* ***Bold*** *indicates the better alternative between the two.*

with and without data augmentation, is considerably worse that the results reported by [16], 4.6% and 3.6% respectively. We attribute this to the absence of an official open-source implementation and the dearth of technical details in the original paper. Anyway, with these experiments we establish that data augmentation is not the reason we achieve an increase in performance, as the model without it achieves a 1% increase in performance compared to the model with it.

Additionally, we experiment with using the Few-shot learner only during training or only during testing. Our experiments show that both have merits of their own. Using the Few-shot learner only during training provides a 2.7% boost, acting as an effective regularizer, while using the Few-shot learner during testing instead of a linear classifier provides a 4.2% boost. Notice, also, that the overall gain from using the Few-shot learner both during training and testing is 6.4%, which is almost equal to the sum of the aforementioned gains (6.9%), suggesting that the usage of the Few-shot learner during training is orthogonal w.r.t its advantages to its usage during testing.

To further establish our method as superior to the respective plain Zero-shot learner, we also report our results on the plain f-VAEGAN for the all datasets in the Zero-shot Learning setting. Our results are shown in Table 3.7. We see that, indeed, our framework improves vastly upon the accuracy of the plain Zero-shot learner. However, as noted earlier, we are unable to replicate the results of the original paper [16].

Training Only with Classification loss

One possible alternative is to completely dispose of the traditional generative frameworks, e.g. the VAEGAN, and just train the generator to facilitate the classifier, meaning we only care about the classification accuracy that the generator leads to and only train it using the classification loss of the Few-shot learner. This is possible to a large extent because the classifier is the same during training and during testing, therefore the generator not only learns to generate discriminative features, but these features are catered towards the classifier that is going to be used during testing. We present our results in Table 3.8. We can see that results remain competitive. Notice that the Zero-shot Learning accuracy on the SUN dataset would be *state-of-the-art* compared to previous work if

	Zero-shot Learning			**Generalized Zero-shot Learning**								
	CUB	AwA2	SUN	CUB			AwA2			SUN		
Method	**aT1**	**aT1**	**aT1**	**u**	**s**	**H**	**u**	**s**	**H**	**u**	**s**	**H**
Z2FSL([16], [17]) (w/o)	61.9	**63.3**	**66.6**	43.1	56.9	49.1	50.8	80.5	62.3	44.2	32.9	**37.7**
Z2FSL([16], [17])	**62.0**	63.2	66.5	42.6	58.7	**49.4**	50.5	81.8	**62.4**	42.1	33.3	37.2

Table 3.9: *Comparison of our approach with (bottom row) and without (top row) finetuning the Few-shot learner during the Classifier Training Stage.* ***aT1*** *denotes the average top-1 accuracy, defined in 3.7, and* ***u***, ***s*** *and* ***H*** *are defined and used as in Equation 3.8.* ***Bold*** *indicates the better between the two.*

	SUN
Method	**aT1**
Z2FSL([16], [17]) (no pretraining)	61.3
Z2FSL([16], [17])	**66.5**

Table 3.10: *Comparison of our approach with (bottom row) and without (top row) the pretraining of the Few-shot learner. Results of the latter presented without finetuning during the Classifier Training Stage, because it consistently deteriorates performance.* ***aT1*** *denotes the average top-1 accuracy, defined in 3.7.* ***Bold*** *indicates the better alternative between the two.*

it wasn't for the method we present in this work.

The Merits of Finetuning the Few-shot learner on unseen descriptions

In Table 3.9, we present our results with and without finetuning the Few-shot learner in the Classifier Training Stage. We can immediately see that finetuning is not consistent in increasing the accuracy. In particular, in Zero-shot Learning, finetuning does not seem to have noticeable effects **on average**. In Generalized Zero-shot Learning, there is a bigger yet still a small effect. We noticed that behavior during training too, so we were not actually able to search for good "finetuning" performance, so we simply selected based on its performance before that. We, nonetheless, present the result with finetuning in order to present a complete framework.

The Merits of Pretraining the Few-shot learner

So far, whenever and wherever we used the Few-shot learner, we always considered it to be pretrained on D_S. In this section, we examine the effect of pretraining the Few-shot learner. In Table 3.10, we can see a sharp drop in performance, 5.2% absolute drop and roughly 8% relative, when no pretraining is carried out, which we deem to be sufficient evidence for the necessity of pretraining.

3.5 Conclusions

In this work, we augment generative approaches to Zero-shot Learning by using the same supervised classifier during both the training and the testing of the Zero-shot learner. Since the choice of classifier class is restricted by the Zero-shot setting, we use a Few-shot learner as a classifier, as it qualifies with only mild assumptions. In this manner, we couple the two settings by reducing Zero-shot Learning to Few-shot Learning via the former's generative network. We formulate our method in a way where we are able to use a broad class of Zero-shot and Few-shot learners. Empirically, we show that our framework significantly improves upon the results of the plain Zero-shot learner and that the changes we propose in every stage of the algorithm result in orthogonal gains w.r.t. one another. Our results are competitive or state-of-the-art in multiple benchmarks.

Chapter 4

Conclusions

In this work, we discuss how the state of affairs in Deep Learning has led to the surging interest in low-data regimes where Deep Learning approaches seem to lag behind compared to other areas of Artificial Intelligence research. We focus our discussion on the Zero-shot Learning setting, where we have a complete absence of the data and are instead provided with auxiliary descriptions of the respective classes, and discuss to which extent the current state-of-the-art approaches in Zero-shot Learning, which utilize generative networks, rely on Deep Learning techniques. In particular, we point out that for the final classification the problem is transformed into a standard supervised task and classifiers like linear classifiers or SVMs are utilized, and how features of images from a pretrained extractor are used rather than the images themselves.

4.1 Our Contributions

In an effort to improve Zero-shot Learning classification accuracy and extend the usage of Deep Learning techniques in low-shot settings, we introduce a Few-shot Learning algorithm to serve as the classifier of a generative approach to Zero-shot Learning, allowing as to use the same classifier during both training and testing. We do so by reducing Zero-shot Learning classification tasks to Few-shot Learning classification tasks.

Another important contribution is the formulation of our approach as a simple plug-and-play framework where any previous Zero-shot Learning approach consisting of a generative network and a Few-shot Learning approach can be plugged in and improve the results of the Zero-shot learner on its own. This way, our framework becomes as general as possible, with clear guidelines on how to augment any Zero-shot learner.

Additionally, we link together the settings of Zero-shot Learning and Few-shot Learning. This could present a step towards a common benchmark, which is more general than Zero-shot Learning and even Generalized Zero-shot Learning. Notice how our framework can seamlessly handle a setting where both a small *real* support is provided for some classes and descriptions for others.

4.2 Discussion

We empirically demonstrate that the Few-shot Learning setting is more appropriate than a standard classification task for classifying real samples of unseen classes in the Zero-shot Learning setting. In particular, our approach achieves state-of-the-art accuracy in 2 out of 3 Zero-shot Learning benchmarks, and more importantly, we illustrate that our framework successfully increases the accuracy of the Zero-shot learner, when that is used in isolation. This justifies the integration of any generative approach to Zero-shot Learning in our Plug-and-Play framework, with only the added pre-training requirements of the classifier. We also show that the merits of our approach can be traced in utilizing a sophisticated algorithm, the Few-shot learner, for classification that is also available during training, allowing the generator to be trained to produce not only realistic samples but samples that aid the classifier in its task. This, in turn, provides a better guarantee for the generalization of the generator to the novel classes during the test setting. We do so by ablating our framework and using the Few-shot Learning classifier only during training or only during testing, where observe gains in performance in both cases compared to the plain Zero-shot learner. It is important to note that these gains are orthogonal/independent, resulting in an increase in accuracy when used in conjunction that is almost equal to the sum of the gains when used in only one setting. One last note for our framework is that the suboptimal performance in the Generalized Zero-shot Learning setting can be attributed to the extra bias we insert with the training of the final classifier on training data, which could bias the classification more towards seen classes than before. Other than our framework, we also demonstrate that the Few-shot learner on its own is competitive in guiding the generator to state-of-the-art results (if we exclude our proposed framework).

4.3 Extensions of our Framework

We include the fact that the framework is Zero-shot-learner and Few-shot-learner agnostic as one of our contributions and merits of our approach. It is, however, completely plausible, possible and highly likely that formulations of Zero-shot learners or Few-shot learners that accommodate the setting we propose or that further entanglement of the two learners can improve upon our results. Notice that the only way we entangle the two learners is by backpropagating the classification loss from the Few-shot learner to the Zero-shot learner when the support set is generated by the generative model, which of course has been proven sufficient for state-of-the-art accuracy compared to previous work.

Reducing the bias towards seen classes which accumulated in their favor because of two different networks being trained on them is of major importance for further advancements.

Moreover, we present a method to finetune the Few-shot learner on the test set. However, the results we observed were mostly inconsistent, although it helped improve the state-of-the-art in some cases, but only by a very narrow margin. We expect that more

sophisticated finetuning methods will allow for even better results. We also observed that this method of finetuning causes severe drops in accuracy if the Few-shot learner is not pretrained.

Using the Few-shot learner alone also seems to be a fertile starting ground on which to build upon, as we demonstrate with our ablations studies. Further pre-training can also be achieved by having the Few-shot learner classify other image dataset, like ImageNet, which is a custom practise in Computer Vision.

4.4 Moving Forward: Personal View

As we noted earlier, ad hoc and complicated approaches tend to improve results in the research setting that has prevailed in Machine Learning, aka standard train and test splits available online for very specific tasks. However, there exists a point of diminishing returns from improvements that arise from added complexity to an already heavily scrutinized basic approach. What we mean by diminishing returns w.r.t complexity is, intuitively, that the time and effort to improve upon an already existing work increases and the marginal gains in performance decrease the more intricate the approaches and the literature become. A pertinent example would have to be the framework on which our work is based on, GANs, in terms of their emergence. GANs have, essentially, single-handedly altered the course of Deep Learning history, largely because they proved to be very effective and versatile. This effectiveness arose from the radical change in philosophy when approaching the training of generative models, which were faced with numerous problems that did not allow for their complete integration in Deep Learning for several years. To briefly elaborate, simply consider the fact that GANs essentially delegated the training of the generative network to Deep Learning itself instead of approximations of the intractable likelihood derived by us, humans, i.e. practitioners let another network decide on what the loss function should be.

Consequently, we made an effort to improve Zero-shot Learning in a similar, albeit less radical and (probably) less consequential, manner. We advocate for a turn towards Few-shot Learning for a proper delegate. We formulate our approach as a simple and network-agnostic framework with the minimum entanglement possible. Finally, in order to experimentally demonstrate the superiority of our approach, we simply used already existing Zero-shot and Few-shot learners. More importantly, we did not merely increase the complexity of a generative approach by adding feedback loops, ad hoc regularization terms, etc, but opted for an intuitive solution that we feel provides fertile ground for further advances in the task we examine and pushes the envelope for the research community.

Bibliography

[1] Artem Oppermann. *Artificial Intelligence vs. Machine Learning vs. Deep Learning*, 2020. https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac.

[2] Jastrow. *File:Didrachm Phaistos obverse CdM.jpg*, 2006. https://commons.wikimedia.org/w/index.php?curid=828070.

[3] Avimanyu786 και Tukijaaliwa. *File:AI-ML-DL.png*, 2020. https://commons.wikimedia.org/w/index.php?curid=90131352.

[4] Egm4313.s12 (Prof. Loc Vu-Quoc). *Neuron3.jpg*, 2018. https://commons.wikimedia.org/w/index.php?curid=72816083.

[5] Elizabeth Goodspeed. *Image*, 2015. https://commons.wikimedia.org/w/index.php?curid=40188333.

[6] Olegalexandrov και Zerodamage. *Gradientdescent.png*, 2012. https://commons.wikimedia.org/w/index.php?curid=20569355.

[7] Vincent Dumoulin και Francesco Visin. *A guide to convolution arithmetic for deep learning*. *arXiv preprint arXiv:1603.07285*, 2016.

[8] Olaf Ronneberger, Philipp Fischer και Thomas Brox. *U-net: Convolutional networks for biomedical image segmentation*. *International Conference on Medical image computing and computer-assisted intervention*, σελίδες 234–241. Springer, 2015.

[9] Christopher Olah. *Image*, 2015. http://chris-chris.ai/2017/10/10/LSTM-LayerNorm-breakdown-eng/.

[10] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra και others. *Matching networks for one shot learning*. *Advances in neural information processing systems*, σελίδες 3630–3638, 2016.

[11] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng και Ahmed Elgammal. *A generative adversarial approach for zero-shot learning from noisy texts*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 1004–1013, 2018.

[12] He Huang, Changhu Wang, Philip S Yu και Chang Dong Wang. *Generative dual adversarial network for generalized zero-shot learning*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 801–810, 2019.

[13] Mohamed Elhoseiny και Mohamed Elfeki. *Creativity inspired zero-shot learning*. *Proceedings of the IEEE International Conference on Computer Vision*, σελίδες 5784–5793, 2019.

[14] Yongqin Xian, Christoph H Lampert, Bernt Schiele και Zeynep Akata. *Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly*. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[15] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein και Bernt Schiele. *Latent embeddings for zero-shot classification*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 69–77, 2016.

[16] Yongqin Xian, Saurabh Sharma, Bernt Schiele και Zeynep Akata. *f-VAEGAN-D2: A feature generating framework for any-shot learning*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 10275–10284, 2019.

[17] Jake Snell, Kevin Swersky και Richard Zemel. *Prototypical networks for few-shot learning*. *Advances in neural information processing systems*, σελίδες 4077–4087, 2017.

[18] Yongqin Xian, Tobias Lorenz, Bernt Schiele και Zeynep Akata. *Feature generating networks for zero-shot learning*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 5542–5551, 2018.

[19] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona και Serge Belongie. *The caltech-ucsd birds-200-2011 dataset*. 2011.

[20] Genevieve Patterson και James Hays. *Sun attribute database: Discovering, annotating, and recognizing scene attributes*. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 2751–2758. IEEE, 2012.

[21] Alex Krizhevsky, Ilya Sutskever και Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. *Advances in neural information processing systems*, σελίδες 1097–1105, 2012.

[22] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson και Sanja Fidler. *Learning to predict 3d objects with an interpolation-based differentiable renderer*. *Advances in Neural Information Processing Systems*, σελίδες 9609–9619, 2019.

[23] Tinghui Zhou, Matthew Brown, Noah Snavely και David G Lowe. *Unsupervised learning of depth and ego-motion from video*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 1851–1858, 2017.

[24] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell και others. *Language models are few-shot learners*. *arXiv preprint arXiv:2005.14165*, 2020.

[25] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. *arXiv preprint arXiv:1810.04805*, 2018.

[26] Sebastian Starke, He Zhang, Taku Komura και Jun Saito. *Neural state machine for character-scene interactions*. *ACM Trans. Graph.*, 38(6):209–1, 2019.

[27] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee και Jehee Lee. *Learning predict-and-simulate policies from unorganized human motion data*. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.

[28] Joseph G Makin, David A Moses και Edward F Chang. *Machine translation of cortical activity to text with an encoder-decoder framework*. Τεχνική Αναφορά με αριθμό, Nature Publishing Group, 2020.

[29] Roman Beliy, Guy Gaziv, Assaf Hoogi, Francesca Strappini, Tal Golan και Michal Irani. *From voxels to pixels and back: Self-supervision in natural-image reconstruction from fMRI*. *Advances in Neural Information Processing Systems*, σελίδες 6517–6527, 2019.

[30] Emmanouil Panagiotou, Georgios Chochlakis, Lazaros Grammatikopoulos και Eleni Charou. *Generating Elevation Surface from a Single RGB Remotely Sensed Image Using Deep Learning*. *Remote Sensing*, 12(12):2002, 2020.

[31] Qin Zou, Lihao Ni, Tong Zhang και Qian Wang. *Deep learning based feature selection for remote sensing scene classification*. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2321–2325, 2015.

[32] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya και others. *Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning*. *arXiv preprint arXiv:1711.05225*, 2017.

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton και others. *Mastering the game of go without human knowledge*. *nature*, 550(7676):354–359, 2017.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. *Proceedings of the IEEE international conference on computer vision*, σελίδες 1026–1034, 2015.

[35] Emma Strubell, Ananya Ganesh και Andrew McCallum. *Energy and policy considerations for deep learning in NLP*. *arXiv preprint arXiv:1906.02243*, 2019.

[36] Christoph H Lampert, Hannes Nickisch και Stefan Harmeling. *Attribute-based classification for zero-shot visual object categorization*. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):453–465, 2013.

[37] Ziad Al-Halah, Makarand Tapaswi και Rainer Stiefelhagen. *Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 5975–5984, 2016.

[38] Zeynep Akata, Florent Perronnin, Zaid Harchaoui και Cordelia Schmid. *Label-embedding for image classification*. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438, 2015.

[39] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato και Tomas Mikolov. *Devise: A deep visual-semantic embedding model*. *Advances in neural information processing systems*, σελίδες 2121–2129, 2013.

[40] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee και Bernt Schiele. *Evaluation of output embeddings for fine-grained image classification*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 2927–2936, 2015.

[41] Bernardino Romera-Paredes και Philip Torr. *An embarrassingly simple approach to zero-shot learning*. *International Conference on Machine Learning*, σελίδες 2152–2161, 2015.

[42] Ruizhi Qiao, Lingqiao Liu, Chunhua Shen και Anton Van Den Hengel. *Less is more: zero-shot learning from online textual documents with noise suppression*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 2249–2257, 2016.

[43] Elyor Kodirov, Tao Xiang και Shaogang Gong. *Semantic autoencoder for zero-shot learning*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 3174–3183, 2017.

[44] Soravit Changpinyo, Wei Lun Chao, Boqing Gong και Fei Sha. *Synthesized classifiers for zero-shot learning*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 5327–5336, 2016.

[45] Kai Li, Martin Renqiang Min και Yun Fu. *Rethinking zero-shot learning: A conditional visual classification perspective*. *Proceedings of the IEEE International Conference on Computer Vision*, σελίδες 3583–3592, 2019.

[46] Jingjing Li, Mengmeng Jing, Ke Lu, Zhengming Ding, Lei Zhu και Zi Huang. *Leveraging the invariant side of generative zero-shot learning*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 7402–7411, 2019.

[47] Vinay Kumar Verma, Dhanajit Brahma και Piyush Rai. *Meta-Learning for Generalized Zero-Shot Learning*. *AAAI*, σελίδες 6062–6069, 2020.

[48] Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li και Li Fei-Fei. *Imagenet: A large-scale hierarchical image database*. *2009 IEEE conference on computer vision and pattern recognition*, σελίδες 248–255. Ieee, 2009.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Deep residual learning for image recognition*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 770–778, 2016.

[50] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado και Jeffrey Dean. *Zero-shot learning by convex combination of semantic embeddings*. *arXiv preprint arXiv:1312.5650*, 2013.

[51] Rohit Keshari, Richa Singh και Mayank Vatsa. *Generalized Zero-Shot Learning Via Over-Complete Distribution*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, σελίδες 13300–13308, 2020.

[52] Stuart Russell και Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4η έκδοση, 2020.

[53] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[54] David H Wolpert. *The lack of a priori distinctions between learning algorithms*. *Neural computation*, 8(7):1341–1390, 1996.

[55] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra και Timothy Lillicrap. *One-shot learning with memory-augmented neural networks*. *arXiv preprint arXiv:1605.06065*, 2016.

[56] Frank Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain*. *Psychological review*, 65(6):386, 1958.

[57] Hans Dieter Block. *The perceptron: A model for brain functioning. i*. *Reviews of Modern Physics*, 34(1):123, 1962.

[58] Andrew R Barron. *Universal approximation bounds for superpositions of a sigmoidal function*. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[59] Sepp Hochreiter και Jürgen Schmidhuber. *Long short-term memory*. *Neural computation*, 9(8):1735–1780, 1997.

[60] Felix A Gers, Nicol N Schraudolph και Jürgen Schmidhuber. *Learning precise timing with LSTM recurrent networks*. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[61] David G Lowe. *Distinctive image features from scale-invariant keypoints*. *International journal of computer vision*, 60(2):91–110, 2004.

[62] David E Rumelhart, Geoffrey E Hinton και Ronald J Williams. *Learning representations by back-propagating errors*. *nature*, 323(6088):533–536, 1986.

[63] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke και Andrew Rabinovich. *Going deeper*

with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 1–9, 2015.

[64] Scott Reed, Zeynep Akata, Honglak Lee και Bernt Schiele. *Learning deep representations of fine-grained visual descriptions*. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, σελίδες 49–58, 2016.

[65] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville και Yoshua Bengio. *Generative adversarial nets*. *Advances in neural information processing systems*, σελίδες 2672–2680, 2014.

[66] Martin Arjovsky, Soumith Chintala και Léon Bottou. *Wasserstein gan*. *arXiv preprint arXiv:1701.07875*, 2017.

[67] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin και Aaron C Courville. *Improved training of wasserstein gans*. *Advances in neural information processing systems*, σελίδες 5767–5777, 2017.

[68] Pascal Vincent, Hugo Larochelle, Yoshua Bengio και Pierre Antoine Manzagol. *Extracting and composing robust features with denoising autoencoders*. *Proceedings of the 25th international conference on Machine learning*, σελίδες 1096–1103, 2008.

[69] Diederik P Kingma και Max Welling. *Auto-encoding variational bayes*. *arXiv preprint arXiv:1312.6114*, 2013.

[70] Yu Xiong Wang, Ross Girshick, Martial Hebert και Bharath Hariharan. *Low-shot learning from imaginary data*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 7278–7286, 2018.

[71] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele και Honglak Lee. *Generative adversarial text to image synthesis*. *arXiv preprint arXiv:1605.05396*, 2016.

[72] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang και Dimitris N Metaxas. *Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks*. *Proceedings of the IEEE international conference on computer vision*, σελίδες 5907–5915, 2017.

[73] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang και Dimitris N Metaxas. *Stackgan++: Realistic image synthesis with stacked generative adversarial networks*. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[74] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang και Xiaodong He. *Attngan: Fine-grained text to image generation with attentional generative adversarial networks*. *Proceedings of the IEEE conference on computer vision and pattern recognition*, σελίδες 1316–1324, 2018.

[75] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient estimation of word representations in vector space*. *arXiv preprint arXiv:1301.3781*, 2013.

[76] Yunlong Yu, Zhong Ji, Jungong Han και Zhongfei Zhang. *Episode-Based Prototype Generating Network for Zero-Shot Learning*. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, σελίδες 14035–14044, 2020.

[77] Yoshua Bengio, Nicholas Léonard και Aaron Courville. *Estimating or propagating gradients through stochastic neurons for conditional computation*. *arXiv preprint arXiv:1308.3432*, 2013.

[78] Diederik P Kingma και Jimmy Ba. *Adam: A method for stochastic optimization*. *arXiv preprint arXiv:1412.6980*, 2014.

[79] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga και others. *Pytorch: An imperative style, high-performance deep learning library*. *Advances in neural information processing systems*, σελίδες 8026–8037, 2019.