NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF RURAL, SURVEYING AND GEOINFORMATICS ENGINEERING

PROGRAM OF POSTGRADUATE STUDIES

Masters Thesis

Utilizing Self Supervised Methods in Unsupervised Metric Learning

Ioannis Andreas P. Tsiotas Niachopetros

ATHENS

SEPTEMBER 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ-ΜΗΧΑΝΙΚΩΝ ΓΕΩΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

Διπλωματική Εργασία

Χρήση Αυτοεπιβλεπόμενων Μεθόδων Μηχανικής Μάθησης στην μη Επιβλεπόμενη Μάθηση Μετρικής

Ιωάννης Ανδρέας Π. Τσιώτας Νιαχοπέτρος

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2022

Masters Thesis

Utilizing Self Supervised Methods in Unsupervised Metric Learning

Ioannis Andreas P. Tsiotas Niachopetros
A.M.:60202123

SUPERVISOR: Konstantinos Karantzalos, Associate Professor, National Technical University of Athens

EXAMINATION COMMITTEE:

Konstantinos Karantzalos, Associate Professor, National Technical University of Athens

Lazaros Grammatikopoulos, Associate Professor, University of West Attica

Maria Pateraki, Assistant Professor National Technical University of Athens

RSLab
Remote Sensing Laboratory
National Technical University of Athens
√ Sensing √ Analytics √ Monitoring

SEPTEMBER 2022

Διπλωματική Εργασία

Χρήση Αυτοεπιβλεπόμενων Μεθόδων Μηχανικής Μάθησης στην μη Επιβλεπόμενη Μάθηση Μετρικής

Ιωάννης Ανδρέας Π. Τσιώτας Νιαχοπέτρος
Α.Μ.:60202123

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Κωνσταντίνος Καράντζαλος, Αναπληρωτής Καθηγητής, Εθνικό Μετσόβιο Πολυτεχνείο

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Κωνσταντίνος Καράντζαλος, Αναπληρωτής Καθηγητής, Εθνικό Μετσόβιο Πολυτεχνείο

Λάζαρος Γραμματικόπουλος, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Δυτικής Αττικής

Μαρία Πατεράκη, Επίκουρη Καθηγήτρια, Εθνικό Μετσόβιο Πολυτεχνείο

RSLab
Remote Sensing Laboratory
National Technical University of Athens
√ Sensing √ Analytics √ Monitoring

ΣΕΠΤΕΜΒΡΙΟΣ 2022

ABSTRACT

Over the past decade deep learning has achieved considerable breakthroughs, however training any model in a supervised manner requires very expensive and time consuming labeling and big models with a large number of parameters. Self supervised learning tries to remedy this problem by training the model on a pretext task without labels and just does the evaluation on the task at hand. Knowledge distillation tries to transfer knowledge from a big to a smaller model

Metric learning tries to create an embedding space where similar objects are pulled together and dissimilar objects are repulsed. Supervised metric learning methods are constantly pushing forward the State of the Art, however its unsupervised counterpart does not get the same attention. That happens, despite the fact that labels have a very important limitation when used as the indicator of similarity. An embedding space which relies on labels cannot sufficiently capture intraclass dissimilarity and interclass affinity.

In this thesis we explore the potential of using methods which have been developed for self supervised learning and knowledge distillation in order to solve metric learning tasks. Using these methods we propose a new framework for metric learning and achieve State of the Art Recall@1 values in the CUB200-2011 dataset.

More specifically we reproduce the paper which achieves State of the Art Recall@1 in supervised metric learning [1]. This framework uses a pretrained backbone transformer's attention blocks adding a fully connected layer as a head. The backbones are pretrained using the methodologies from Vit [2], Dino [3] and Deit [4]. The final fully connected layer projects the outputs to a hyperbolic instead of the euclidean vector space. Apart from the reproduction of experiments we also train the framework using pretraining with the methodology from Ibot [5]. We display Recall@1 77.8% from Ibot compared to 77.3% Dino.

Next we tried to use the Ibot for metric learning purposes by just evaluating the features extracted from the Ibot model on a metric learning setting. The model collapsed in all 12 different setups despite our extensive experimentations in order to stabilize it.

The main contribution of this thesis is the creation of an unsupervised framework for metric learning. This framework utilizes self distillation inspired by Dino having two instances of the same network trained simultaneously. The first one updates its parameters via backpropagation and the second one updates towards the direction of the first using exponential moving average. Our framework also uses relaxed contrastive loss which allows the creation of a metric embedding space.

Another important detail regarding the methodology is that the head features are projected to a hyperbolic embedding space instead of the classic euclidean. The pretraining of our model is done using Ibot pretrained model instead of some model trained in a supervised method. This framework achieves State of the Art Recall@1 values of 70.5% in CUB200-2011 dataset but displays its instability underperforming by 20% in CARS196 and 6% SOP datasets respectively compared to State of the Art methods.

In conlusion our method displays the large potential of using methods derived from self supervised learning in metric learning. It also reaffirms the effectiveness and limitations of using transformers in metric learning.

SUBJECT AREA: Computer Vision, Deep Learning

KEYWORDS: Metric Learning, Self Supervised Knowledge, Knowledge Distillation, Neural Networks

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια η βαθιά μάθηση έχει πραγματοποιήσει πολύ σημαντική πρόοδο αναγνωρίζοντας εικόνες καλύτερα από τον άνθρωπο, πραγματοποιώντας αυτόματες μεταφράσεις, και αποδεικνύοντας τις δυνατότητες της σε μια σειρά από άλλες εργασίες. Παρά τις σημαντικές αυτές εξελίξεις στην βαθιά μάθηση, η εκπαίδευση των μοντέλων παραμένει μια εξαιρετικά υπολογιστικά ακριβή διαδικασία η οποία απαιτεί τεράστια σετ δεδομένων. Μέχρι την τελευταία πενταετία ο κύριος όγκος των μοντέλων εκπαιδευόταν με επιβλεπόμενο τρόπο με το μοντέλο να δέχεται ένα σετ δεδομένων στο οποίο το κάθε δείγμα συνοδευόταν από μια ετικέτα η οποία το περιέγραφε.

Όπως είναι κατανοητό η διαδικασία της χειροκίνητης δημιουργίας τέτοιων ετικετών είναι εξαιρετικά ακριβή και χρονοβόρα. Επιπλέον δεν επιτρέπει την χρήση της πλειοψηφίας του όγκου των δεδομένων που συλλέγονται σήμερα καθώς ένα πολύ μικρό υποσύνολο τους έχει κατηγοριοποιηθεί με ετικέτες. Ακόμα μία από τις πιο σημαντικές εκφάνσεις της ευφυίας είναι η ικανότητα γενίκευσης σε νέα δεδομένα με βάση ήδη υπάρχουσα γνωστική υποδομή. Όταν ένα μοντέλο εκπαιδεύεται με ένα συγκεκριμένο σετ δεδομένων με ένα συγκεκριμένο στόχο και αξιολογείται στο ίδιο σετ δεδομένων και τον ίδιο στόχο αυτή η ικανότητα τίθεται υπό αίρεση.

Στην προσπάθεια αντιμετώπισης όλων αυτών των προβλημάτων τα τελευταία χρόνια έχει εισαχθεί η αυτοεπιβλεπόμενη μάθηση η οποία δεν χρησιμοποιεί τις προαναφερθείσες ετικέτες κατά την εκπαίδευση του μοντέλου. Πιο συγκεκριμένα τα μοντέλα εκπαιδεύονται με κάποια προσχηματική εργασία (pretext task) η οποία προκύπτει από τα ίδια τα δεδομένα και αξιολογούνται συνήθως ως προς την ικανότητα τους να πραγματοποιήσουν μια από τις κλασσικές εργασίες της βαθιάς μάθησης όπως η ταξινόμηση, η σημασιολογική κατάτμηση και η ανίχνευση αντικειμένων.

Σε εργασίες αυτόματης αναγνώρισης κειμένου η αυτοεπιβλεπόμενη μάθηση έχει ήδη αποδείξει τις δυνατότητες με δημοσιευμένες εργασίες όπως το BERT [6] να ανταγωνίζονται και να υπερνικούν επιβλεπόμενες μεθόδους. Σε ότι αφορά την όραση υπολογιστών τα πράγματα είναι σημαντικά συνθετότερα καθώς η φύση του σήματος (εικόνες) είναι συνεχής ενώ στην αναγνώριση κειμένου (προτάσεις) είναι διακριτή πράγμα που αυξάνει την πολυπλοκότητα. Οι προσχηματικές εργασίες στην όραση υπολογιστών παίζουν πολύ σημαντικό ρόλο, και μπορεί να είναι η αναγνώριση των στροφών μιας εικόνας, η επίλυση ενός παζλ από τμήματα της εικόνας ή ακόμα και η αναπαραγωγή κρυμμένων τμημάτων της εικόνας με βάση τα υπάρχοντα τμήματα της.

Η Βαθιά Μάθηση Μετρικής είναι το βασικό πρόβλημα το οποίο θα προσπαθήσουμε να επιλύσουμε με τεχνικές δανεισμένες από την αυτοεπιβλεπόμενη Βαθιά Μάθηση. Ο σκοπός της μάθησης μετρικής είναι η δόμηση ενός διανυσματικού χώρου στον οποίο τα όμοια αντικείμενα έλκονται και τα ανόμοια απωθούνται. Αυτή η διαδικασία έλξης-απώθησης εκφράζεται από μια μετρική η οποία είναι σε θέση να αυξάνει την απόσταση μεταξύ όμοιων και να μειώνει την απόσταση μεταξύ ανόμοιων αντικειμένων.

Ένας περιορισμός στην παραδοσιακή μορφή μάθησης μετρικής είναι ότι το μέτρο της ομοιότητας ή ανομοιότητας μεταξύ των αντικειμένων είναι οι ετικέτες στις οποίες αναφερθήκαμε. Αυτός όμως ο ορισμός της ομοιότητας είναι στενός και δεν επιτρέπει να αποτυπωθούν σωστά η ομοιότητα μεταξύ των αντικειμένων διαφορετικών κλάσεων και η ανομοιότητα μεταξύ αντικειμένων της ίδιας κλάσης. Γίνεται λοιπόν ξεκάθαρη η ανάγκη για

ανάπτυξη μη επιβλεπόμενων μεθόδων μάθησης μετρικής

Μια ακόμα έννοια που έπαιξε σημαντικό ρόλο σε αυτή την διπλωματική είναι η απόσταξη γνώσης. Απόσταξη γνώσης ονομάζεται η διαδικασία με την οποία ένα μικρού μεγέθους μοντέλο επιχειρεί να μιμηθεί τις πιθανοτικές κατανομές που προκύπτουν από ένα μεγαλύτερου μεγέθους μοντέλο. Το πρώτο μοντέλο ονομάζεται μαθητής και το δεύτερο δάσκαλος. Ουσιαστικά πρόκειται για μεταφορά γνώσης από το μεγάλο στο μικρότερο δίκτυο με την βοήθεια της μίμησης. Η μέθοδος αυτή βασίζεται στην ιδέα ότι τα νευρωνικά δίκτυα έχουν τις ίδιες μαθησιακές δυνατότητες ανεξαρτήτως μεγέθους. Η δυσκολία εκπαίδευσης ενός μικρού δικτύου σε σχέση με ένα μεγαλύτερο οφείλεται στην ευαισθησία του μικρού δικτύου σε ότι αφορά την παραμετροποίηση [7]. Αυτή η διπλωματική επικεντρώθηκε κυρίως στην αυτοαπόσταξη γνώσης που είναι μια παραλλαγή της απόσταξης στην οποία η μεταφορά γνώσης γίνεται εντός του ίδιου δικτύου.

Στόχος αυτής της διπλωματικής είναι ο συνδυασμός ιδεών της αυτοεπιβλεπόμενης μάθησης και της απόσταξης γνώσης για την δόμηση μιας μεθοδολογίας μάθησης μετρικής το οποίο να δομεί τον μετρικό διανυσματικό χώρο με την ελάχιστη δυνατή επίβλεψη. Στην συνέχεια θα παρουσιαστούν περιληπτικά τα σημαντικότερα στοιχεία αυτής της μεθοδολογίας.

Τα πρώτα πειράματα που παρουσιάζονται αφορούν την αναπαραγωγή των αποτελεσμάτων του paper με τις πιο υψηλές ακρίβειες στην επιβλεπόμενη μάθηση μετρικής. Το paper αυτό βασίζεται σε προεκπαιδευμένες αρχιτεκτονικές transformer αντί των κλασσικών συνελικτικών δικτύων που χρησιμοποιούν τα περισσότερα papers του αντικειμένου. Οι αρχιτεκτονικές που χρησιμοποιούνται είναι το vit [2] και το deit [4]. Για την αρχιτεκτονική Vit χρησιμοποιήθηκε ένα μοντέλο εκπαιδευμένο με επιβλεπόμενο [2] και ένα με μη επιβλεπόμενο τρόπο [3].

Το μοντέλο αυτό επανεκπαιδεύεται στο σετ δεδομένων που μας ενδιαφέρει και ακολούθως μια στρώση από πλήρως συνδεδεμένους νευρώνες προβάλλουν την έξοδο του transformer σε έναν ευκλείδειο διανυσματικό χώρο. Ακολούθως και με την βοήθεια των αρχών της προβολικής γεωμετρίας αυτές οι έξοδοι επαναπροβάλλονται σε ένα υπερβολικό διανυσματικό χώρο ο οποίος μοντελοποιείται με την βοήθεια της σφαίρας Poincare. Εκτός από την αναπαραγωγή των πειραμάτων του paper πραγματοποιήθηκε και εκπαίδευση με χρήση του Vit εκπαιδευμένου με την αυτοεπιβλεπόμενη μέθοδο Ibot [5]. Η τιμή της ανάκλησης (Recall@1) βελτιώνεται κατά 0.5% σε σχέση με την χρήση του Dino επιτυγχάνοντας την μέγιστη ανάκληση που έχει επιτευχθεί στην επιβλεπόμενη μάθηση μετρικής με χρήση μοντέλου εκπαιδευμένου με μη επιβλεπόμενο τρόπο.

Στην συνέχεια επιχειρήθηκε η χρήση του πλαισίου του Ibot για σκοπούς μάθησης μετρικής. Ο τρόπος με τον οποίο πραγματοποιήθηκε αυτό είναι με την χρήση προεκπαιδευμένων στο Imagenet [8] μοντέλων του Ibot τα οποία επανεκπευδεύονται στο CUB200-2011 [**?**]. Στην συνέχεια τα χαρακτηριστικά που εξάγονται από τα blocks του transformer αξιολογούνται ως προς την ανάκληση τους στο σετ του CUB200-2011. Τα αποτελέσματα αυτής της σειράς πειραμάτων ήταν απογοητευτικά με την εκπαίδευση να καταρέει σε κάθε περίπτωση και ανεξαρτήτως των υπερπαραμέτρων οι οποίες χρησιμοποιήθηκαν. Κατέστη λοιπόν ξεκάθαρο ότι απαιτείται μια ειδική συνάρτηση κόστους η οποία να ωθεί το σετ δεδομένων να εκπαιδευτεί να αναγνωρίζει την ομοιότητα ή ανομοιότητα μεταξύ των δεδομένων.

Εν τέλει αποφασίστηκε να δομηθεί ένα πλαίσιο το οποίο βασίστηκε στην αυτοαπόσταξη γνώσης όπως αυτή παρουσιάζεται στο Dino [3]. Συγκεκριμένα 2 δίκτυα ίδιας αρχιτεκτονικής και ίδιας αρχικοποίησης παραμέτρων εκπαιδεύονται ταυτόχρονα (σχήμα δάσκαλος-μαθητής). Τα δίκτυα αποτελούνται από τα blocks του transformer και μια στρώση από πλήρως συνδεδεμένους νευρώνες που προβάλουν τα χαρακτηριστικά που μας ενδιαφέ-

ρουν σε έναν διανυσματικό χώρο με την διαστατικότητα που επιθυμούμε. Οι παράμετροι του μαθητή βελτιστοποιούνται με την βοήθεια των gradients που υπολογίζονται με την οπίσθια διάδοση ενώ του δασκάλου υπολογίζονται με την βοήθεια ενός εκθετικού κινούμενου μέσου. Ουσιαστικά ο εκθετικός κινούμενος μέσος είναι μια γραμμική συνάρτηση η οποία συνδέει τις παραμέτρους του μαθητή με αυτές του δασκάλου με τον κύριο συντελεστή της να είναι το momentum. Το momentum μεταβάλλεται κατά την διάρκεια της εκπαίδευσης ξεκινώντας από την τιμή 0.9998 και καταλήγοντας στο 1.

Βασικό σκέλος του παραπάνω πλαισίου είναι η συνάρτηση κόστους. Η συνάρτηση που επιλέχθηκε είναι μια μορφή της αντιθετικής (contrastive) συνάρτησης κόστους [9] η οποία αντί για τις ετικέτες ως μέτρο ομοιότητας χρησιμοποιεί την ομοιότητα που προκύπτει από την σύγκριση κατά ζεύγη μεταξύ των εξόδων των 2 δικτύων. Η συνάρτηση κόστους αναπτύχθηκε αρχικά ως συνάρτηση κόστους για μάθηση μετρικής με απόσταξη γνώσης αλλά εδώ χρησιμοποιήθηκε σαν συνάρτηση κόστους για μάθηση μετρικής με αυτοαπόσταξη γνώσης.

Τα τελευταία σημαντικά στοιχεία της μεθοδολογίας αφορούν το μοντέλο που χρησιμοποιήθηκε για προεκπαίδευση και τον διανυσματικό χώρο στον οποίο προβάλλονται τα διανύσματα. Σε ότι αφορά την προεκπαίδευση αυτή αποφασίστηκε να γίνει με ένα μοντέλο προεκπαιδευμένο στο Imagenet με την αυτοεπιβλεπόμενη μέθοδο του Ibot. Η προεκπαίδευση αποφασίστηκε να γίνει με αυτοεπιβλεπόμενη μέθοδο καθώς δεν έχει πραγματοποιηθεί έτσι ξανά στην βιβλιογραφία και είχε ενδιαφέρον να μελετηθεί αν θα επηρεαστεί η συμπεριφορά του μοντέλου από την προεκπαίδευση. Ο διανυσματικός χώρος στον οποίο προβάλλονται οι έξοδοι του δικτύου είναι ο ευκλείδειος σε κάποια πειράματα και σε άλλα ο υπερβολικός. Η προσσέγγιση του υπερβολικού διανυσματικού χώρου γίνεται με μία σφαίρα Poincare ακριβώς όπως περιγράφηκε και στην μεθοδολογία επιβλεπόμενης μάθησης που αναλύθηκε παραπάνω. Μετά από εκτενή πειράματα οι τιμές των βασικών υπερπαραμέτρων που υιοθετήθηκαν είναι Momentum 0.9998, Lr 0.00003, Weight Decay 0.00001 και batch size 120.

Έγινε ένα πλήθος πειραμάτων πάνω στο πλαίσιο μάθησης μετρικής που δομήθηκε. Η πρώτη σειρά πειραμάτων αφορούσε το κατά πόσο ωφελεί η χρήση υπερβολικού έναντι του κλασσικού ευκλείδειου χώρου. Από τα πειράματα φάνηκε ότι είτε επιτρέποντας είτε όχι σε όλες τις στρώσεις του δικτύου να επανεκπαιδευτούν στο CUB200-2011 η τιμή της ανάκλησης αυξάνεται κατά 0.6% με προβολή των διανυσμάτων στον υπερβολικό χώρο σε σχέση με την προβολή στον ευκλείδειο. Καταδεικνύονται έτσι τα οφέλη της χρήσης υπερβολικού σε σχέση με τον ευκλείδειο διανυσματικό χώρο. Σε ότι αφορά την τιμή της καμπυλότητας του χώρου έγιναν τέσσερα διαφορετικά πειράματα κύριο συμπέρασμα των οποίων ήταν ότι έχοντας ως βάση την καμπυλότητα 0.1, μείωση της σε 0.01 δίνει τιμή ανάκλησης κατά 0.3% υψηλότερη ενώ αύξηση της καμπυλότητας πέραν της τιμής 0.1 οδηγεί σε σημαντικά χαμηλότερες τιμές ανάκλησης.

Στην συνέχεια αναλύθηκε η συνεισφορά που έχουν οι τεχνικές επαύξησης των δεδομένων. Σε ότι αφορά την επαύξηση των δεδομένων υπάρχουν αρκετές τεχνικές που χρησιμοποιήθηκαν ενώ σε πολλές περιπτώσεις ως είσοδος στο μοντέλο δόθηκαν πάνω από μια εκδοχή της επαυξημένης εικόνας. Η μέγιστη τιμή της ανάκλησης που επιτεύχθηκε στο CUB200-2011 ήταν 71% τιμή η οποία αποτελεί και ρεκόρ για μεθοδολογία μη επιβλεπόμενης μάθησης μετρικής, ξεπερνώντας το προηγούμενο ρεκόρ κατά 3%. Για τα επόμενα πειράματα που πραγματοποιήθηκαν προτιμήθηκε ο συνδυασμός επαυξήσεων με τον οποίο επετεύθχη τιμή ανάκλησης 70.5% επειδή κρίθηκε ότι η μικρή αύξηση στην ανάκληση δεν δικαιολογούσε την σημαντική αύξηση σε υπολογιστικό κόστος.

Εκτός από πειραματισμός με διαφορετικές επαυξήσεις πραγματοποιήθηκε και σειρά από

πειράματα με συνδυασμό της αντιθετικής συνάρτησης κόστους με την συνάρτηση κόστους που χρησιμοποιήθηκε στο Dino. Τα αποτελέσματα με όλους τους συνδυασμούς ήταν κατώτερα και έτσι η χρήση συνδυασμού συναρτήσεων κόστους εγκαταλείφθηκε.

Τα τελευταία αποτελέσματα που παρουσιάστηκαν αφορούσαν την σύγκριση των αποτελεσμάτων μας με τα αποτελέσματα των papers που αφορούν την μη επιβλεπόμενη μάθηση μετρικής. Όπως αναφέρθηκε η μεθοδολογία μας πετυχαίνει State of the Art ανακλήσεις στο CUB200-2011. Ωστόσο όταν η μεθοδολογία δοκιμάζεται σε άλλα σετ δεδομένων όπως το CARS196 και το SOP τα αποτελέσματα δεν είναι αντίστοιχα. Πιο συγκεκριμένα στο CARS196 η διαφορά στις ανακλήσεις είναι 20% χαμηλότερες σε σχέση με το State of the Art paper [10]. Σε ότι αφορά το σετ δεδομένων SOP η διαφορά με το State of the Art paper είναι 6% κατά του δικού μας. Αξίζει να αναφερθεί ότι αυτή η τάση για πολύ υψηλές ανακλήσεις στο CUB200-2011 με χρήση transformers και συγκριτικά χαμηλές στα dataset CARS196 και SOP έχει παρατηρηθεί ήδη σε δημοσιευμένες εργασίες [1], [11].

Από την εργασία αυτή αποδείχθηκε ότι είναι εφικτός ο συνδυασμός μεθόδων μάθησης μετρικής με μεθόδους αυτοεπιβλεπόμενης μάθησης. Πιο συγκεκριμένα αποδείχθηκε η βιωσιμότητα χρήσης αντιθετικής μη επιβλεπόμενης συνάρτησης κόστους σε συνδυασμό με αυτοαπόσταξη γνώσης. Επιπλέον διαφαίνεται ότι στο CUB200-2011 τα αποτελέσματα είναι ανώτερα από τα αντίστοιχα που επιτυγχάνονται με μεθόδους που έχουν ώς βάση τις ψευτοετικέτες-pseudolabels [12], [13]. Εν τέλει επιβεβαιώθηκε η δυναμική αλλά και οι προβληματικές της χρήσης της αρχιτεκτονικής transformers στην μάθηση μετρικής.

Στο τελευταίο κεφάλαιο αφιερώνεται σε 2 πειράματα που αφορούν την αυτοεπιβλεπόμενη μάθηση και πραγματοποιήθηκαν στην αρχή της διπλωματικής και αφορούν τον συνδυασμό μιας κλασσικής μεθόδου αυτοεπιβλεπόμενης μάθησης με μια μέθοδο απόσταξης γνώσης. Αποφασίστηκε τα πειράματα να μην συμπεριληφθούν εντός του κύριου μέρους της εργασίας αλλά στο παράρτημα. Επίσης στο παράρτημα περιλαμβάνονται και κάποιες οπτικοποιήσεις της προσοχής του δικτύου της προταθείσας μεθοδολογίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Όραση Υπολογιστών, Βαθιά Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μετρική Μάθηση, Αυτοεπιβλεπόμενη Μάθηση, Απόσταξη Γνώσης, Νευρωνικά Δίκτυα

ACKNOWLEDGEMENTS

I would like to thank my supervisor Konstantinos Karantzalos for his tremendous support, guidance, intuition and the endless motivation and trust he provided me. There was nothing more I could have asked for.

Additionally, I would like to thank Bill Psomas and Yannis Kakogeorgiou as their ideas and intuitions on different subjects were extremely important during these past months. Without them this thesis would have been impossible to write. The amazing working atmosphere and the many hours we spent discussing the details of this implementation were exceptionally important for me.

Finally, I would like to thank my family and friends for the love and support they have provided me throughout this project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

The breakthroughs in AI systems over the last couple of years have been stunning. In a time span of less than 10 years AI systems have been able to surpass humans in their ability to identify objects [14], automatically translate sentences [15] and even analyze the context of sentences [16]. Despite the huge leaps forward achieved by State of the Art AI methods there are still important limitations regarding traditional deep learning methods. Supervised learning requires a vast amount of predominantly manually labeled data to properly operate. The labeling of these data can be very expensive, time consuming and highly subjective. It also does not allow us to fully exploit the potential of big data. The second problem that arises from trying to teach models using labeled data is that the AI systems learn to solve these task specific problems but they can hardly generalize their observation to other similar problems.

Over the past few years AI scientist have been trying to address the aforementioned issues proposing a new learning system where the supervisory signal is extracted from the data itself, without the need for expensive and time consuming labeling. Additionally the fact that these models do not rely on labels in order to train means that when asked to perform inference on labeled data they have to generalize their observations regarding data thus solving the second problem. This family of methods is called self supervised learning and it has achieved tremendous progress over the past few years rapidly closing the gap with its supervised counterpart.

The second and equally important pillar this thesis stands on is metric learning. Metric learning is all about the highly subjective concept of similarity. Its goal is to create a vector space where similar data are grouped together and dissimilar data are kept apart. However, the problem with traditional metric learning methods is that they utilize labels during the creation of this space which significantly narrows the definitions of similarity and dissimilarity to be label-driven.

Therefore, the need for unsupervised metric learning methods becomes apparent, and this is our motivation in pursuing to propose a robust unsupervised metric learning framework based on self supervised learning. Another motivation was the will to integrate this self supervised metric learning framework with the very impactful idea of knowledge distillation.

1.2 Challenges

Next we will briefly mention some of the most important challenges which arise in our metric learning framework:

1. There is no specific augmentation set for metric learning, which means that we have to rely on techniques developed for other tasks and datasets. This probably does not allow as to exploit the full potential of any metric learning framework.
2. There is no precedent to our knowledge of a self supervised method which is utilized for unsupervised metric learning purposes.
3. Training deep learning models is a computationally expensive procedure.

1.3 Contribution

Having outlined our fields of interest we will now highlight the main contribution of our thesis.

1. We propose the utilization of self distillation for metric learning purposes.
2. We train our model with a model pretrained in self supervised fashion which is a very rare case in metric learning.
3. We propose the use of a batchwise loss with hyperbolic distances.
4. We propose the use of an unsupervised metric loss along with self distillation. To the best of our knowledge self distillation has only been used along classification losses [3], [17], [18]
5. We conduct extensive ablations in order to see which element contributes most to our framework

1.4 Structure

In the next section we will analyze the structure of this thesis:

- **Chapter 1** makes a brief reference to the motivations behind this thesis, the challenges we met during the experiments, and the contribution of our thesis in self supervised and metric learning.
- **Chapter 2** presents the most important papers which inspired this thesis. It also briefly explains some ideas behind neural networks and the main concepts our paper is based upon.
- **Chapter 3** presents in detail the methodologies used or developed during this thesis.
- **Chapter 4** presents the dataset used, the evaluation protocol and the experimental setup of our experiments. It also presents the results of reproduced experiments and the experiments we conducted to tune the frameworks. It includes extensive ablations regarding key components such as data augmentation and hyperbolic embeddings. In conclusion, it includes extensive comparisons between ours and the State of the Art methodologies in unsupervised metric learning.
- **Chapter 5** outlines the findings and future research directions of this thesis.
- **Chapter 6** includes 2 different sections. In the first some attention maps from our architecture are presented and in the second one some experiments in self supervised learning irrelevant to the rest of the thesis are presented.

2. BACKGROUND

2.1 Relative Work

In this chapter the papers that are most essential for this thesis are documented and their respective methodologies analyzed. The main volume of bibliography for this projects consists of papers in the fields of Self-supervised learning and Metric learning.

Caron et al. [3] proposed a self supervised method which utilizes a variant of self distillation. This framework utilizes 2 identical networks the parameters of which are initialized identically (teacher-student schema). More specifically every image in the dataset gets augmented and cropped both locally and globally. All crops are used as inputs of the student network while only global crops are used as input in the teacher network. Then the networks are trained so that the one mimics the probability output of the other. The datasets used in this paper are Imagenet for classification and Oxford and Paris image retrieval purposes. The achieved accuracies in Imagenet utilizing a VIT-S architecture are 77,0% for linear classification and 74,5% for knn clustering.

Zhou et al. [5] proposed a self supervised method which studies masked image modeling in a a self supervised framework. The basis of this method is self distillation similar to [3]. Specifically self distillation is performed on masked patch tokens and the teacher network is taken as the online tokenizer along with self distillation on the patch token to get visual semantics. Evaluation of the method was conducted on Imagenet, Flowers, Cifar 10 and Cifar 100 for classification purposes COCO for object detection and ADE20K for image segmentation purposes. Ibot achieves 82.3% for linear classification.

Kim et al. [9] proposed a novel unsupervised batchwise loss for knowledge distillation in metric learning. This loss is based upon the contrastive loss but replaces the labels utilized in contrastive loss as a metric of intraclass affinity and interclass dissimilarity with the pairwise similarity between the 2 embeddings. These embeddings are the outputs of teacher and student networks respectively. This method was evaluated on CARS196, CUB200-2011 and SOP datasets and it achieves 72.1% on CUB200-2011, 89.6% on CARS196 and 79.8% on SOP.

Ermolov et al. [1] proposed a straightforward method which combines transformers and hyperbolic geometry for metric learning purposes. They evaluate their method in CARS196, CUB200-2011, SOP and inshop datasets. Their method achieves State of the Art recall values in metric learning achieving 89.2% in CARS196, 85.6% in CUB200-2011 3 85.9% in SOP and 92.5% in Inshop.

Gidaris et al. [19] proposed a novel pretext task for the training of self supervised models. Every image in the dataset is rotated in 90,180, 270 degrees and these images along with the non rotated one are the inputs of a cnn architecture that predicts these rotations. The extracted features are then used as inputs of a linear classifier that classifies the dataset. The datasets that were used are Cifar 10, ImageNet and Places-205. Apart from classification this pretext task is also used for segmentation and object detection. The achieved accuracies for Imagenet are 72,9% for classification, 54,4% for segmentation and 50,0% for object detection.

Ishan Mishra and Laurens Van der Maaten [20] proposed a methodology that learns invariant features with data augmentation using jigsaw pretex task. The methodology is based on contrastive learning, where an image and its transformed counterpart are used

as inputs of 2 networks with the networks been encouraged to recognize the pair of images as a positive one despite the transformation (augmentation invariant learning). The initial and augmented data are compared with the help of a NCE (Noise Contrastive Estimator) [21]. Additionally, a memory bank is introduced which allows the replacement of negative samples with an exponential moving average of previous samples.The dataset used is Pascal VOC, with the achieved accuracies ranging from 33,7% up to 68,6%.

Carl Doersch and Andrew Zisserman [22] experimented with multiple combinations of pretext tasks in order to train models that were afterwards used for classification, object detection and depth detection. The four pretect task which were combined are jigsaw [23], colorization [24], exemplar [25] and motion segmentation [26]:

- A naive based method where an independent classifier is used for every pretext task
- A method where a single head receives a linear combination of feature vectors that are weighted based on a sparse matrix.

The datasets used are Imagenet, Pascal VOC and NYU V2. The achieved accuracies vary from 36,21% up to 66,82% for Imagenet classification task.

Zhang etal. [27] proposed a mutual learning strategy akin to knowledge distillation where a cohort of neural networks collaboratively teach each other. In this case the models share the loss function which is the sum of a cross entropy loss and a Kullback Leibler loss that connects the various networks. The datasets used are CIFAR 100 and Market1501 and the accuracies vary from 20,12% up to 95,90%.

2.2 Knowledge Background

In this chapter the fundamental methodological elements are analyzed:

2.2.1 Feedforward Neural Networks

Deep Feeedforward Neural networks are mathematical models that approximate a function f by defining a mapping y = f(x,θ) and learn the value θ that achieves the best function approximation. The basic unit of a neural network is a neuron. The neuron takes as input a vector x and computes an output z that is a linear combination of that vector:

$$z_n = w_n * x_n + b_n.$$

where w is a weight vector and b is the bias term. A non-linear transformation is usually applied to the output, which is called activation function. The activation function is used in order to allow the network to learn a non-linear mapping of the input to output. The most widely used activation functions are ReLU, tanh and sigmoid function.

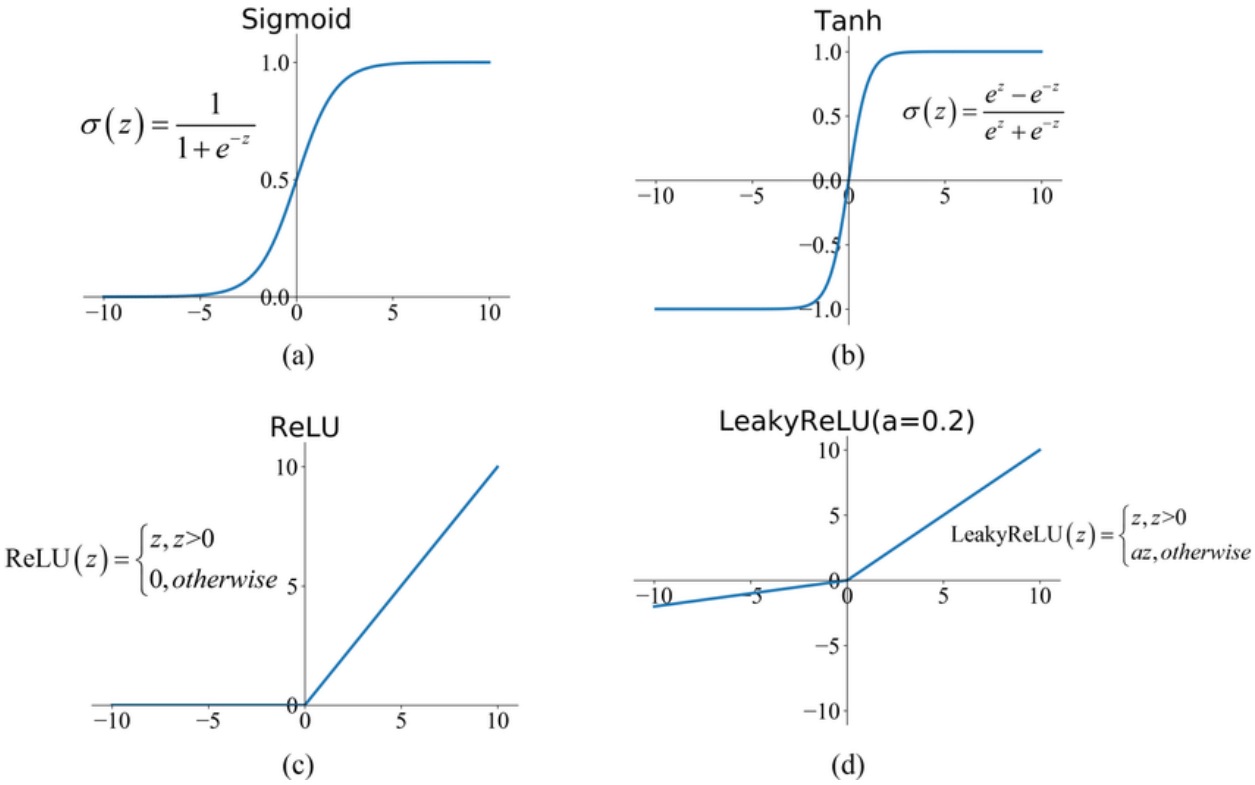


Figure 1: The most important activation functions.

All neurons of a neural network are grouped into layers with every neuron applying a linear transformation to the output from the neurons of the previous layer. All neural networks are a chain of functions applied to an input. Every layer of a neural network apart from the first and the last layer is known as a hidden layer. Every neuron of one layer of the network is connected to all other neurons of the neighbouring layers (dense connections). The first layer is called input layer and the last layer is called output layer. A feed forward neural network has no feedback connections where the outputs of the model are fed back to the model. A network that uses feedback connection is a recurrent neural network (RNN) [28].

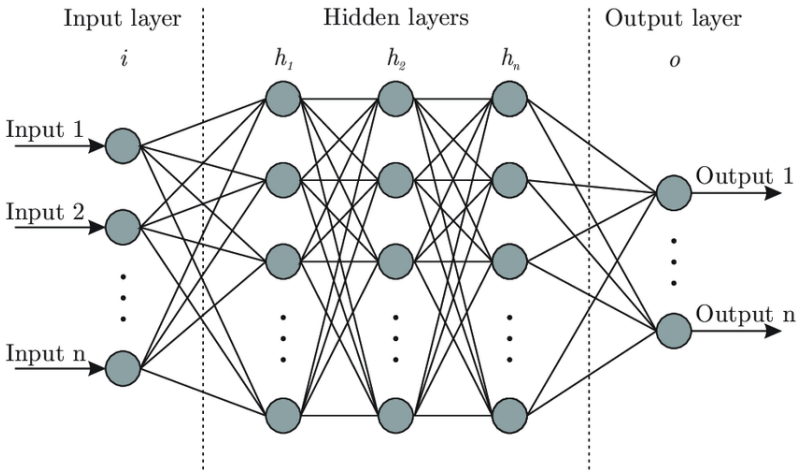


Figure 2: Graphical representation of a neural network with 3 hidden layers

Training a neural network is an optimization problem. It is analyzed to learning the optimal parameters (w_n, b_n) for all the layer neurons of the network. The initialization of weights is a crucial part of successfully training a neural network. The initialization can be done

either randomly with values close to zero or with a function written by the programmer. The input is firstly propagated through the network to obtain its value (forward propagation). After that the output is compared to its expected value (grountruth) through a loss function L(y[exp], y). After that the loss is propagated backwards through the layers of the neural network [29]. This is the way that weights and biases are updated. The gradient of loss with respect to each layer parameter is computed and this gradient is used to update the parameter. The factor that determines how ”aggressive” the optimization is going to be is learning rate. An example of the optimization process for the weight matrix and the bias term is as follows:

$$w \leftarrow w - * \frac{L(S,w)}{w}$$

$$b \leftarrow b - * \frac{L(S,b)}{b}$$

This parameter update is gradient descent. Due to computational reasons many times gradient descent is performed in accumulations of samples known as batches (batch gradient descent). Gradient descent that is performed on individual samples is known as stochastic gradient descent. There are many optimization methods but the two most widely used are Stochastic gradient descent with momentum [30] and Adam [31].

2.2.2 Convolutional Neural Networks

Feedforward neural networks have significant limitations when it comes to computer vision. Images are represented as arrays with 3 dimensions **(height, width, number of bands)** with the value of every pixel been a combination of the different bands. In a feedforward neural network every neuron in the input layer is connected to every neuron of the first hidden layer. This is very computationally expensive for data as high dimensional as images and therefore only recently have feedforward neural networks started been used for computer vision problems, achieving state of the art performance using transformers [[3], [2]].

Convolutional neural networks (CNNs) [32] overcame the limitations of feedforward neural networks in computer vision by utilizing convolutions for some of the layers. Convolutions in the simplest case of a grayscale image apply a moving filter array (usually 3*3, 5*5, 7*7) which is known as kernel, that replaces the image pixels with the sum of the dot products of the image pixels and the kernel. The same kernel is applied to every position of the array. When the image is 3dimensional the convolution kernels are also 3dimensional. However the convolution is 2d since information changes only along the height and width axis [33]. For every convolution step the kernel moves by a pixel window more than 1. That number of pixels is known as a stride. The immediate result of using a stride is that the output's size is reduced compared to the input [34]. Another type of layer in CNNs is the pooling layer. Pooling layers reduce the dimensionality of data by combining the output of neurons at one layer into a single neuron at the next layer [33]. Pooling layers either replace the output of neurons with their maximum or their average value and are either known maxpool or averagepool layers respectively. Local pooling layers combine small tiles, whereas global pooling layers act on all the neurons of the feature maps. Often activation functions are applied after the convolutional layers in order to apply a linear transformation to the output of the convolutional layers. Additionally batch normalization and dropout are usually used with CNNs to regularize the outputs of the convolutional layers.

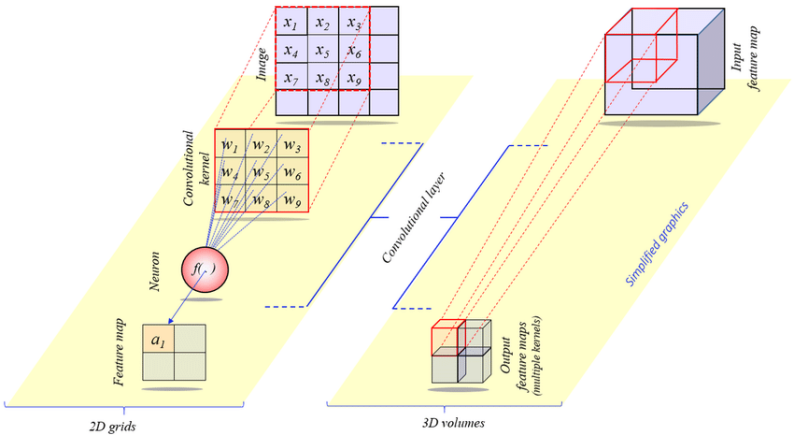


Figure 3: Visualization of convolution operations in a CNN

Now we will attempt a deep dive to the mechanisms behind CNNs. As mentioned a simplistic CNN consists of convolutional and Pooling layers (Backbone part) and some feed-forward layers on top (Classifier part). We will assume that the input to this CNN is a grayscale image for simplicity. The output of every layer in the backbone part has Dx-HxW where D denotes the number of features this layer has extracted. D*W denotes the **number of areas** of interest that have been extracted from image. For example, if the backbone output has 512x14x14 dimensions this means that there are 512 different features extracted in 14x14=196 areas of interest. These areas of interest are the output activations for some given filter and they are known as feature maps. Usually feature maps are visualized after convolutional layers of a CNN.

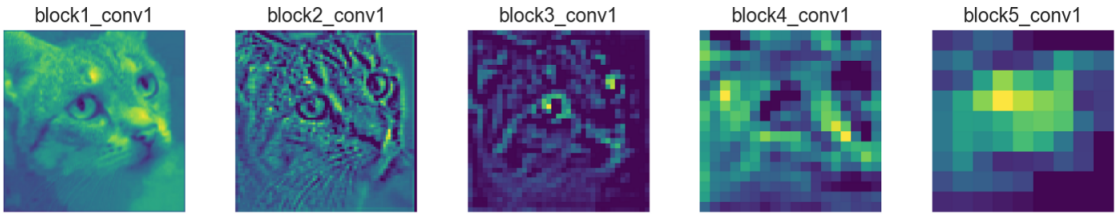


Figure 4: Visualization of the feature maps in a CNN.

The idea behind convolutional neural networks is that signals are compositional [33]. For example in an image, pixels assemble to form oriented edges and oriented edges assemble to form corners. Therefore a convolutional neural network is trained to recognize these hierarchical representations starting from very low level such as lines and edges, and in the end been able to recognize high level features such as complex objects.

According to Goodfellow etal. [35] a CNN is equivalent to a fully connected network with an infinitely strong prior of weights. This strong prior requires all weights in a spatial network to be shared but shifted in position.

2.2.3 Transformers

In this chapter the transformer architecture will be analyzed. Transformer [15] is a neural network architecture that holds the state of the art in self supervised learning for computer vision tasks on the Imagenet dataset [5], [36]. This architecture was firstly developed for Natural Language Processing tasks but it was quickly adopted in computer vision. In order to better comprehend transformer architecture some key terms need to be explained:

- Encoder Decoder Network: A network inspired by NLP models, which uses an encoder that converts a group of sentences to 2 dimensional vectors. These sentences are fed to the network sequentially. These 2 dimensional vectors are then fed to another neural network which converts these vectors to sequences of words. This can be applied to vision where instead of sentences the input is a sequence of pixels (image) and the output is a segmentation mask of this image [37].
- Self-Attention: A mechanism which allows the network decoder to extract information about all the encoder past states using a weighted sum of these states. Given a set of input vectors [x1, x2, ..., xn] and another set of output vectors [y1, y2, ..., yn] self attention is computed as the weighted average of the input vectors.
- Transformer: A transformer is an Encoder Decoder Network that utilizes multi-head self attention. Next, the architecture of a typical vision Transformer will be described. However the basic principles apply to NLP transformers as well. The researchers who developed vision transformers tried to create an architecture with the least number of possible changes compared to the original NLP transformer.
    1. The image is split in patches of specific size called tokens and a positional embedding is created for every patch describing the relative position of one patch compared to all the others. A learnable class embedding (token) is added.
    2. Therefore the image consists of a group of patches (usually sized 16x16)
    3. The patches are flattened and linearly projected to vectors on an embedding space.
    4. These vectors along with their respective positional embeddings are afterwards used as the input of the transformer
    5. The multi-layer self attention assigns weights to tokens based on their relative importance in the image. An MLP head encodes the output of the attention block. This is repeated multiple times
    6. An MLP head encodes the output of the attention network and produces the logits
    7. Softmax function can be applied in order to convert these logits to propabilities

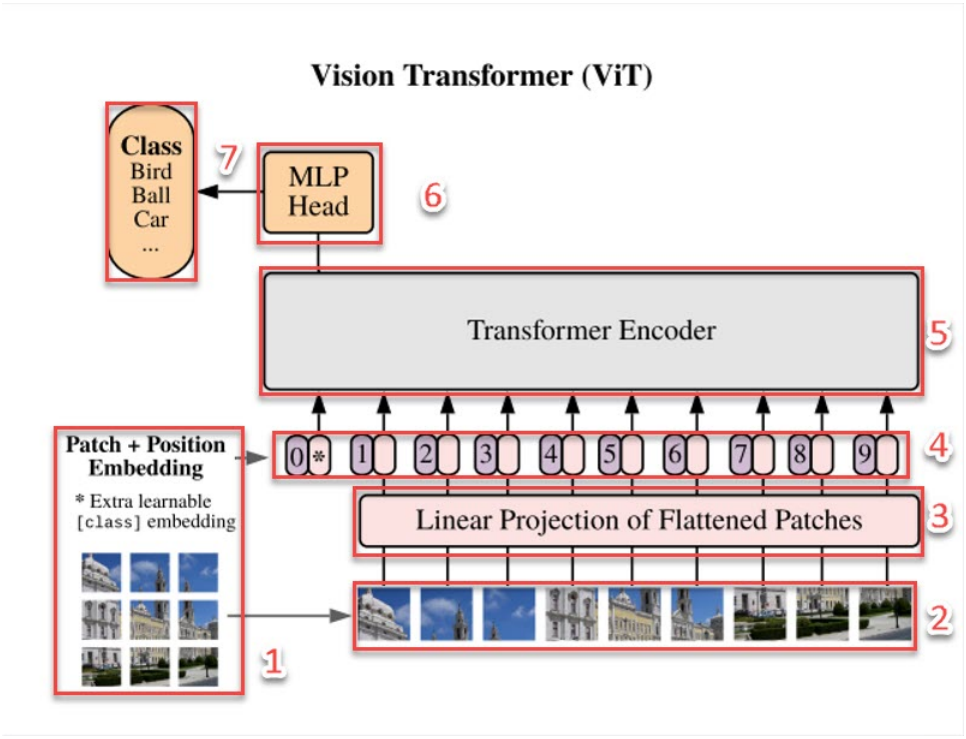


Figure 5: Visualization of VIT pipeline.

2.2.4 Energy Based Models (EBM)

Energy based models learn a data distribution by analyzing a data subset [19] i.e the model is able to predict multiple outputs.

$$F(\overline{x}, \overline{y}, \overline{z}) = argminE(x, y, z).$$

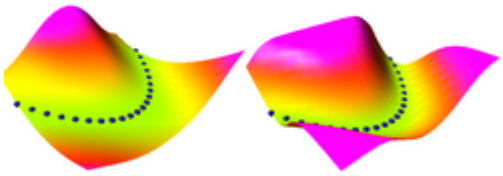


Figure 6: Visualization of an energy based model with the line representing x variables.

The valleys on the above visualization indicate high energy variables to be predicted y relatively to the observed variables x i.e high correlation. On the contrary the elevations of the visualization indicate low energy variables to be predicted y relatively to the observed variables x i.e low correlation. Energies are measured in arbitrary units and therefore they need to be converted to probabilities [38]. This is done via the Gibbs Distribution:

$$P(x, y) = \frac{e^{(x,y)}}{\int y \in y^{e^{(x,y)}}}.$$

Training an energy based model relies on 2 classes of learning methods:

- Contrastive Methods where the model parameters are configured in a way that all the points of the data manifold have low energy and all the points outside the data manifold have high energy.

- Architectural Methods where the volume of high energy points is minimized or maximized through regularization methods.

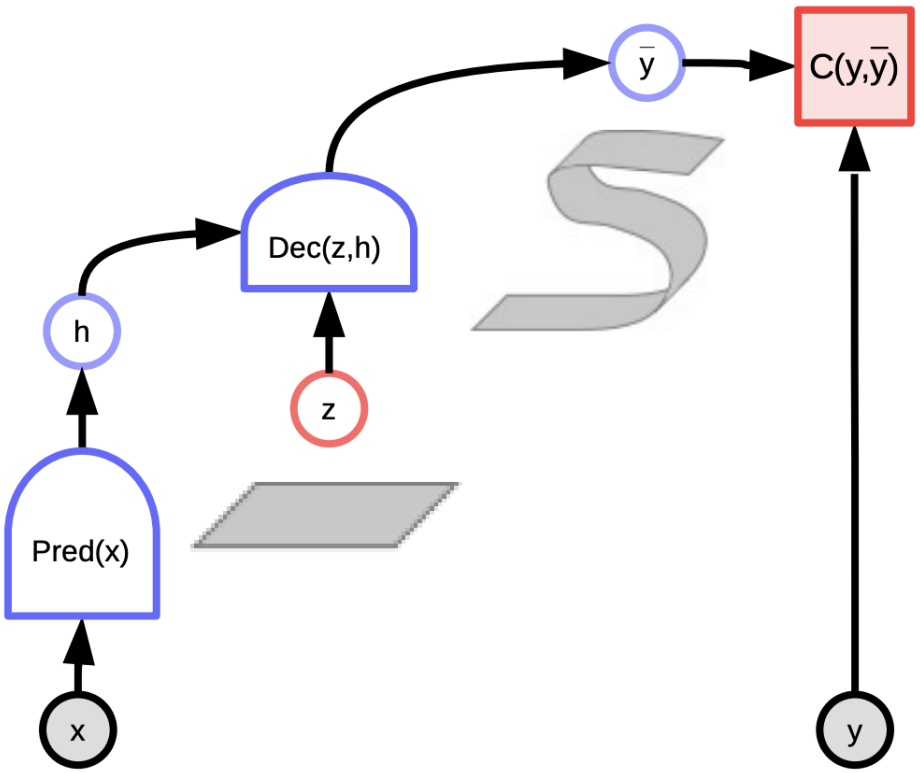


Figure 7: Computation graph for energy based models

2.2.5 Self Supervised Learning

Supervised learning in computer vision produces state of the art results when provided with a large amount of data. Simultaneously, major technological advancements have enabled the collection and storage of vast amounts of unlabeled data. However, obtaining labelled data is usually very expensive and time consuming [39]. There are semi-automatic ways for obtaining labels (hashtags, GPS locations) [39] but they still heavily rely in some form of human supervision.

Unsupervised learning is the type of learning that does not involve any manual labeling. Self Supervised learning (SSL) encompasses supervised and unsupervised learning by using supervisory tasks derived from the data itself. These are the **pretext tasks** whereas the tasks that utilize the labels based on the features learned from the pretext tasks are **downstream tasks**. For these downstream tasks the inputs are usually the visual feature representations learned from the intermediate layers of the model that was trained using the pretext task. Therefore the goal of the pretext task is extracting meaningful features that can be used for the supervised task. A self supervised model transforms high dimensional data to high level feature representations.

An important example of Self Supervised Learning in natural language proccesing is BERT [6] where the model predicts masked words given the words that appear before and after it. However in computer vision things are more complicated as it is more difficult to represent

uncertainty in computer vision problems than it is for language prediction models [40]. That is due to the fact that it is significantly more difficult to associate an image feature to all the other features of an image than it is to associate one word to all other words of a limited vocabulary [40].

As mentioned self Supervised learning consists of a multitude of methods which attempt to learn useful representations by using various auxiliary/pretext tasks. Pretext tasks usually involve hiding parts of complex signals (e.g sound, images, videos) from a neural network and forcing the neural network to predict them [39]. The main categories of Self Supervised learning methods are described below:

- Generative Methods [[41], [42]] where a distribution of the data and their latent embedding is created. The final feature representations are the ones learned from this distribution.

- Discriminative methods which rely on energy based models and are divided into two subgroups:
  - Contrastive methods where the signal is transformed and then both the transformed and initial view of the sample are used as input of the network. Contrastive methods were first introduced in [43] and they are borrowed from metric learning.
  - Non Contrastive methods [[44], [45], [46]] rely only on the positive pairs for every sample by minimizing the distance between the elements of the positive pair. These methods require extensive fine tuning so they are in danger of collapsing to constant solutions in the representation space without the network learning anything [44].

2.2.6 Metric Learning and Losses

Metric learning is an approach of deep learning that determines the similarity or dissimilarity between objects based on a distance metric [47]. It aims to simultaneously decrease the distance between similar (positive pair) and increase the distance between dissimilar objects (negative pairs) in an embedding space [48].

In metric learning the final output is an embedding space where pairs of embedding vectors which derive from similar images are close to each other whereas pairs of embedding vectors which derive from dissimilar images are away from each other. Therefore metric learning is a very different from classification or image segmentation.

A classic metric learning framework is as follows:

1. The classes of the dataset are split 50-50 with the first subset been used for training and the second subset been used for the validation of the method. Unlike classification the method is not evaluated in the categories it is trained on. The reason is that the goal is for the network to create good quality embedding spaces for data of which the labels are unknown.

2. The network is trained in a similar manner as a network trained for classification purposes with one important difference. The final layer instead of representing the number of classes we wish our model to be classified in it represents the dimensionality of the embedding space we wish the vectors to reside into

3. The final important element of metric learning methodologies is the loss function used for the metric learning task. The loss function used cannot be one of the more traditional loss functions utilized for classification and regression purposes. Therefore next we will describe the main loss function families used for metric learning.

The metric loss used for this push-pull operation is of paramount importance. There are two types of loss functions used in metric learning embedding losses which operate between batch members and classification losses which utilize a weight matrix that transforms the embedding space to vector of class logits [48]. Some of the most well known loss functions introduced for metric learning purposes are the following:

Contrastive loss: It makes the distance of positive pairs smaller than a specified threshold and the distance of negative pairs larger than the threshold.

$$L = (d_p - m_{pos})_+ + (m_{neg} - d_n)_+.$$

A major disadvantage of contrastive methods is that they require a large amount of negative samples, which renders them very computationally expensive. The mining of negative pairs is also time consuming. One way of dealing with this was proposed in [20] where a memory bank along with noise contrastive estimator are utilized in order to represent negative pairs more efficiently. Another way is proposed in [49] which deals with the problem by using very large batches.

Triplet Loss: A loss that consists of an anchor, a positive and a negative sample where the positive is closer to the anchor than the negative sample [47].

$$L = (d_p - d_n + m)_+.$$

Mining hard triplets is essential for the proper training of the model otherwise it is possible that training will stagnate. The problem with hard triplets mining is that it is a time consuming procedure and an ill defined problem [50]. If mining is inadequate there is a high chance that the model will stagnate and not train at all. These are the main ideas behind most metric losses and using these main ideas it is possible to extract most other metric losses.

2.2.7 Transfer Learning

Transfer learning is a family of techniques that use the weights of models pretrained on large datasets for training instead of randomly initializing them. This drastically reduces the huge computational cost that is required in order to train even medium sized datasets like Imagenet [8] allowing for an approach that adapts the pretrained model to a different domain. According to Chollet [51] transfer learning can be divided into two branches, feature extraction and fine-tuning. In feature extraction the network is split into a backbone and classifier part with only the weights of the classifier been trained in the new domain. The weights of the backbone (feature extractor) are not updated. In finetuning the whole network is retrained and all the parameters are updated. According to Yosinski etal. [[52] transferring features without finetuning causes performance degradation due to the specificity of the features extracted as well as the poor optimization for the target network.

2.2.8 Hyperbolic Embedding Space

The most common way of data representation in deep metric learning is embedding in euclidean space. This happens mostly for convenience reasons, as it has vectorial struc-

ture and it is also the natural generalization of the intuitive to us 3dimensional space [53]. However it has been displayed experimentally that in cases such as graph models the data nature is not euclidean and therefore the well known and documented euclidean space cannot provide sufficient representation quality [54]. In such cases a non euclidean domain would be truly useful. Nikel and Kiela [55] proposed the use of a hyperbolic embedding space i.e an embedding space with a constant negative curvature instead of zero. Hyperbolic embedding space in neural networks is usually approximated using an n-dimensional Poincare ball. Hyperbolic embedding space displays interesting tree like properties which can be very useful when trying to represent data of hierarchical structure to the embedding space [53]. The representational capacity of embedding space is of vital importance in metric learning methodologies and therefore state of the art methodologies in both supervised and unsupervised metric learning utilize it [1], [56].

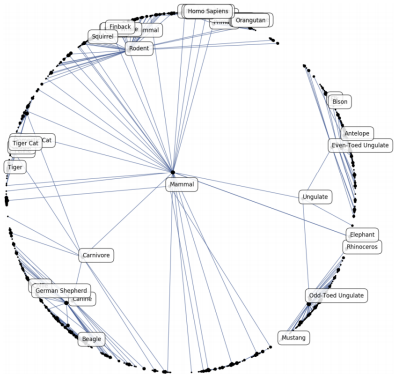


Figure 8: Hyperbolic embedding space.

2.2.9 Knowledge Distillation

Knowledge distillation is the practise of transferring knowledge from a large and cumbersome model to a smaller one. It is based upon the assumption that neural networks have similar representation capacity irrespective of their size [57]. However, smaller networks tend to be harder to train. Therefore the difficulty of training a smaller network is mostly a problem of optimization and it is possible to transfer knowledge from a small to a large neural network.

Knowledge distillation was first introduced in [58] who trained a large network and then used its data probability distribution to train a smaller model that mimicked the larger model data probability distribution. This procedure is heavily dependent on the training data and therefore if the training dataset changes the model requires retraining.

Hinton etal. [7] proposed a knowledge distillation framework that instead of utilizing data distributions utilized a loss function called distillation loss. This method relies on a large cumbersome model (teacher) and small (student) model. The large model output is used as soft label for the smaller model. The methodology works as follows.

- For each test sample the most probable class is found according to the teacher.
- The full probability distribution q over all classes is calculated that minimizes:

$$KL(p^g, q) + \sum_{m \in A_k} KL(p^m, q).$$

where KL denotes KL divergence, and p^m, p^g denote the probability distribution of the teacher and the student model respectively. Therefore, the network contains 2 losses, the distillation loss which minimizes the distance between the student the teacher (soft labels) outputs and the cross entropy loss function which minimizes the distance between the student outputs and the groundtruth (hard labels).

2.2.10 Self Distillation

Self distillation [59] is a methodology which allows a network to distill knowledge from itself. The idea behind self distillation is diminishing the need for a cumbersome teacher and a student who tries to mimic the teacher's outputs by allowing a model to learn from itself. The two main schools of thought regarding self distillation in deep learning are the following:

1. Zhang etal. [59] proposed a network for self distillation which utilizes one model. More specifically after each of the model blocks a bottleneck and a fully connected layer are applied to the outputs of the network along with softmax function. This allows us to construct 2 losses for every layer. The first one is a cross entropy loss between the output of each layer and the labels, and the second one is Kullback Leibler Divergence between the output of every layer and the final output of the network. This method is much better explained using the paper's graphical abstraction.

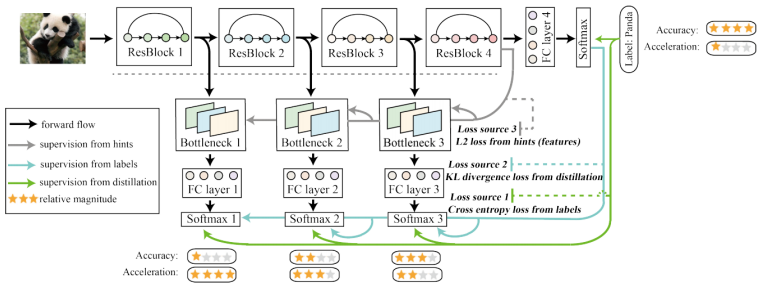


Figure 9: Self distillation graphical abstract

Caron etal. [3] proposed a methodology which rather than teaching the shallow layers of the network from the deep layers of that network uses two versions of the same network with the parameters of the one network been updated slightly with the help of the parameters of the first network. This methodology is the backbone of the thesis and therefore it will be further explained in the methodology chapter.

3. METHODOLOGY

3.1 Hyperbolic Embedding Learning Reproduction

The first series of experiments we will analyze are the ones which we reproduced from the paper [1]. In an effort to assess the usefulness of Ibot pretraining in metric learning we conducted another experiment where we initialized the weights with a Vit trained with Ibot.

3.1.1 Hyperbolic Embedding Space

The most important part of this methodology is the hyperbolic embedding space our vectors are projected into. It is well known that hyperbolic space cannot be isometrically embedded to an euclidean space [60] and therefore our solution is to use a model of hyperbolic geometry in order to endow a subset of the euclidean space with a hyperbolic metric [61]. Our model of choice is Poincare Ball model. The Poincare ball model is defined by the manifold $D^n = (x \in R^n : ||x|| < 1)$ endowed with a Riemanian metric $g^D(x) = \lambda_x^2 g^E$ where g^E is the euclidean metric tensor and λ_x is the conformal factor.

The most important parameter of the Poincare model is its curvature C(x). Finding the proper value of C(x) can either be done through trial and error or using the next empirical formula [55]:

$$C(x) = (\frac{0.144}{\delta_\chi})^2$$

Where shows how close to the hyperbolic our structure is [1]. However it has been experimentally proven [55] that this formula does not always provide the curvature which achieves the best results since image representations computed by CNNs/transformers might not be totally accurate.

From the extensive bibliography about the different operations regarding hyperbolic space the only one we are interested in is the exponential mapping to a hyperbolic embedding space. We need to find a bijective map in order to map a euclidean vector to the hyperbolic space.

$$exp_x^c(v) = x \oplus_c (\tanh{(\sqrt{c}\frac{^c_x||v||}{2})}\frac{v}{\sqrt{c}||v||}).$$

where ||υ|| denotes the second order norm of the speed vector on the Poincare ball, c denotes the curvature, λ denotes the conformal factor and x the euclidean vector.

The final element of the methodology that needs to be discussed is clipping. Clipping is performed by norm after the exponential mapping which constrains the norm not to exceed $\frac{1}{\sqrt{c}}(110 - 3)$

3.1.2 Architectures

One of the most important components of this paper is the transformer architectures it used. These architectures are vit-s [2], [3], [5] pretrained in both a supervised and unsupervised fashion and deit [4] architecture trained only in a supervised fashion. The features extracted from the attention blocks of these architectures are either evaluated

(final embedding size 384), or passed through a fully connected layer which creates an embedding space of 128 dimensions. In case hyperbolic embedding space is used, the final layer of the network is the layer which bisects the euclidean space to the hyperbolic one, otherwise it is a simple normalization layer. By default the patch embedding layer is frozen, its parameters not trained during the training stage.

3.2 Metric learning with IBOT

Next we will examine our methodology for Ibot use in metric learning. First we will explain Ibot framework and then our utilization of it.

3.2.1 IBOT Method

The most important element of Ibot is the introduction of bert like masked lingual modeling for computer vision tasks. The basic idea of the paper is to use Bert-like masking adjusted to image space. This method consists of the steps which will be analyzed next:

1. 2 random augmentations are applied to every image yielding 2 distorted views u and ʋ. Afterwards blockwise masking is applied to these 2 views and we get their masked counterparts , .
2. Both views are used as input of the networks. The [CLS] token is extracted and the loss function minimizes the cross entropy between the categorical distributions of the teacher and the student networks.

$$L_{CLS} = -P_{\acute{\theta}}^{CLS} \upsilon^T log P_{\theta}^{CLS}(u).$$

3. The masked version of one view is used as input in the teacher and the masked version of the other version is used as input in the student. The [PATCH] token is extracted and the loss objective is to minimize the cross entropy between the categorical distributions of the teacher and student networks:

$$L_{MIM} = -\sum_{n=1}^{N} m P_{\acute{\theta}}^{PATCH} {u_i}^T log P_{\theta}^{PATCH}(\hat{u_i}).$$

4. The 2 losses are added and the gradients are propagated through the student network whereas the teacher is updated using EMA (for more details see our method)

3.2.2 Metric IBOT

Our idea was to utilize Ibot model for metric learning purposes with the slightest possible changes in the framework. The dataset we chose to use was CUB200-2011 which is the smallest of the datasets used as metric learning benchmarks. We created a set of augmentations (see augmentations section) for validation purposes. This set of validation augmentations is relatively simple and allows our model to be trained on heavily distorted views of the images and then be evaluated on images with slight if any distortions.

This is a very important part of the success of any metric learning framework as it should learn to learn features invariant to transformations during training and then be able to

embed new images to the vector space irrespective of their transformations. Next the features were extracted from the last block of the transformer and then evaluated using Recall@1 metric. Both teacher and student models were pretrained Vit small networks using Ibot methodology.

3.3 Proposed Framework Analysis

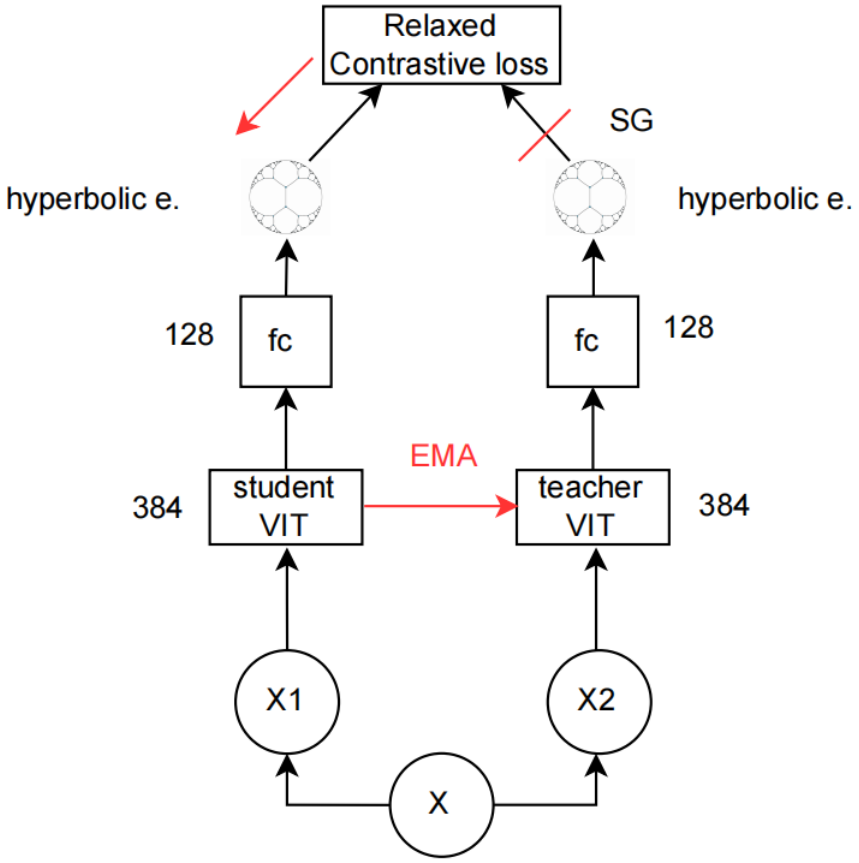


Figure 10: Graphical Abstract of our method.

3.3.1 Architecture

Our model uses Vit small architecture adapted for metric learning purposes. The vision transformer architecture proposed in [2] remains almost intact apart from the final MLP head which is replaced by a fully connected layer that projects the transformer features to an embedding space of 128 dimensions. We tried using l2 normalization after this fully connected layer but it proved detrimental for the training of the model and therefore we abandoned it.

3.3.2 Self Distillation

At the center of our method sits the Dino [3] self distillation method with some important changes in order to be functional in a metric learning framework. Like traditional knowl-

edge distillation the method uses 2 networks, a teacher and a student. These networks should be perceived as 2 instances of the same network as they have the same architecture and their parameters are initialized with the same values. Therefore ours is a self distillation method.

In Dino framework the goal is to match the probability distributions of the teacher and the student networks by minimizing a cross entropy loss between the 2 outputs. Our method significantly differentiates from this idea as instead of minimizing differences between probability distributions it utilizes the embedding vectors in order to calculate their pairwise similarity. That assists in deciding the distance of these vectors on the embedding space. Some important details of our method stemming from Dino are the following:

1. The gradients are propagated through the student (see chapter 2) and its parameters are updated.
2. The teacher network is optimized using exponential moving average (EMA) and therefore no gradients are propagated through the teacher. EMA on a high level is a linear function which updates the teacher slightly towards the direction of the student updates via the momentum encoder [62] . The formula which is used for this is displayed next:

$$\theta_t = \lambda * \theta_t + (1 - \lambda)\theta_s.$$

where λ denotes the momentum which changes according to a scheduler, θs denotes the feature of the student and θt the features of the teacher. It becomes apparent that the scheduler which is used in order to update the momentum is of the utmost importance as it controls the degree to which student training affects teacher training. However it is not the only scheduler that is used in this model. Both weight decay and lr are updated in every iteration of every epoch using a scheduler. In the next section we will briefly describe these three schedulers.

3.3.3 Schedulers

Firstly, it is crucial to address the importance of schedulers in our model. Self supervised learning has been shown from a multitude of experiments conducted during this thesis to be unstable and sensitive to hyperparameter tuning, more so than its supervised counterpart. Therefore any of the widely used Lr updating strategies like decreasing the Lr with a steady step or reducing it when some accuracy metrics remain steady for many iterations are insufficient. Instead we opted for a cosine scheduler [63] which updates the learning rate in every iteration of every epoch based on a cosine function. A different cosine scheduler is also used for momentum and weight decay updates.

Cosine scheduler is a complex one which updates the model at every iteration of every epoch. A diagram displaying the momentum scheduler along with the one used for Lr and weight decay appears next.

The above diagrams display the mean values of Momentum, Weight Decay and Lr for every epoch. Momentum starts at 0.9998 and increases up to one, because there is need for the two models to start training with the minimum amount of parameter correlation. The values of the hyperparameters should converge as training continues. The same

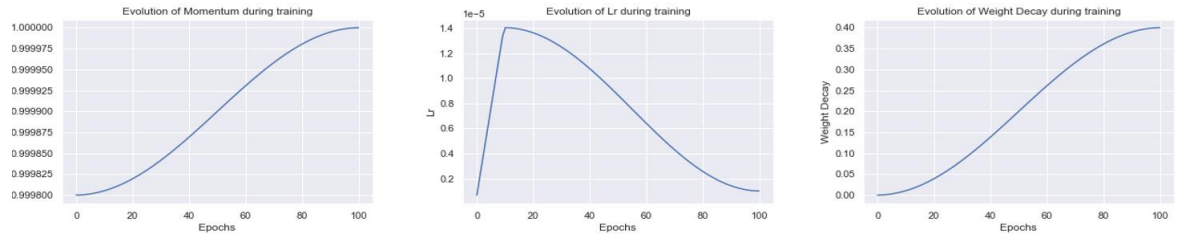


Figure 11: Schedulers of our Method.

logic of steady increase is applied to weight decay, whereas Lr follows the opposite rout of decreasing as learning goes on. This is very important as our Lr has to decrease when minimum parameter changes are required and increase when aggressive training is required. The steady increase over the first 10 epochs is explained by the fact that they are used for warmup where no update of parameters takes place.

3.3.4 Relaxed Contrastive Loss

The most critical component of our framework is relaxed contrastive loss [9]. This loss is the first batchwise unsupervised metric loss and it was originally developed for distillation in metric learning achieving state of the art results. The original contrastive loss consists of an attracting and a repelling term which forces vectors that originate from instances of the same class to be close to each other in the embedding space. Simultaneously, it forces feature vectors originating from instances of different classes to be apart from each other. The main objective of relaxed contrastive loss is to replace the labels of the classes with another indicator of similarity.

Relaxed contrastive loss relies on the outputs of 2 networks which are trained simultaneously (online distillation). Instead of relying on labels as the decisive factor of the push-pull operation it uses the semantic similarity of the feature vectors in order to decide whether they form a positive or negative pair. The semantic pairwise similarity is formulated as the distance of the feature vectors in the teacher embedding space and it forms weights for the new loss function. The labels are therefore relaxed using weights which derive from pairwise similarity. The mathematic formula of this operation is displayed next:

$$w_{ij}^s = exp(-\frac{||f_i - f_j||_2^2}{\sigma}) \in [0, 1].$$

Then relaxed contrastive loss is formulated as such:

$$L(X) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}^s (\frac{d_{ij}^t}{i})^2 - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(1 - w_{ij}^s)[-(\frac{d_{ij}^t}{i})^2]_+^2.$$

where d_{ij} denotes the euclidean distance between embedding vectors f_i and f_j, δ is a margin and μ_i is the average distance of all pairs associated with f_i in the batch. μ_i is used in order to make the scales of pairwise distances similar for both positive and negative pairs. It therefore makes possible for us to avoid l2 normalization. In the original paper Relaxed contrastive loss required the normalization of the teacher embeddings. Experimentally, we found that in our network this creates big differences in the magnitude of teacher and student vector values. The losses skyrocketed which rendered any further training of the model unachievable.

Relaxed contrastive loss was originally developed for knowledge distillation purposes in a metric learning setting. However, our methodology is self distillation [3] and therefore our goal is to train two views of the same network instead of one network assisting the training of another. As mentioned relaxed contrastive loss requires as input the embedding vectors of 2 networks and therefore we decided to apply it to the [CLS] token embeddings we extract for every image.

In conclusion we would like to discuss why an unsupervised batchwise metric loss is as important as it is, apart from the obvious cost effectiveness and time saving aspect of the expulsion of labels we have already mentioned. Relying on labels for the push-pull operation, significantly limits the ability of the space to recognize intraclass dissimilarity and interclass affinity. E.g in the next series of images 3 birds from the classes Common and Forster terns of CUB200-2011 dataset are represented. The first and the third image represent a common tern whereas the second represents a forster tern. For an untrained eye the first and the second images are more similar than the first and the third. If the metric space which is proposed relies on labels it becomes obvious that these relationships cannot be captured effectively.

Figure 12: Examples of images in CUB200-2011.

3.3.5 Dino Loss

For some of the experiments we implemented the loss proposed in Dino [3] as an auxiliary loss to help the model train. The idea behind using this second loss function stems from the fact that the pretrained model we utilized was trained on a self distillation loss applied on the [CLS] token of the transformer and a MIM loss applied on the [PATCH] token. Therefore we thought that using relaxed contrastive loss along with one of these 2 losses would allow better finetuning on the new dataset.

Self distillation loss minimizes the cross entropy between the categorical distributions of the [CLS] tokens from the student and the teacher networks. The formula of Dino loss is displayed next:

$$L_{CLS} = -P_{\acute{\theta}}^{CLS} v^T log P_{\theta}^{CLS}(u).$$

It is important to mention 2 techniques used along this loss function in order to avoid collapse:

1. The [CLS] token of the teacher and the student networks is scaled using temperature i.e the [CLS] token is divided by the temperature. The temperature of the student is consistently 0.1 whereas the teacher temperature is changed using a scheduler.

   This scheduler is used so that temperature can linearly increased from an initial to an end value during the warmup phase. In our case we chose the temperature to remain 0.04 during the whole training procedure. The next figure displays the

scheduler's behaviour with an initial temperature of 0.04 an end end temperature of 0.07 and 10 warmup epochs. This procedure is known as sharpenning

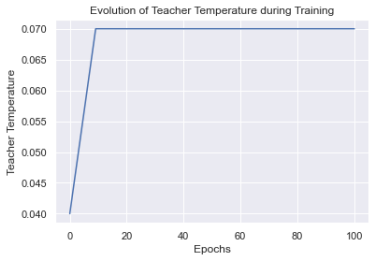


Figure 13: Temperature Scheduler of Dino loss.

2. There is also a centering operation which takes place during the loss calculation. Centering in essence is an operation similar to batch normalization but simpler. It calculates the per batch mean of the teacher output and then subtracts it from the teacher output. The formula of this operation is displayed next:

$$c = mc + \frac{(1-m)}{B}\sum_{i=1}^{B} g_\theta(x_i).$$

where B denotes batch size and m is a rate parameter.

Some last details we would like to assess regarding differences from our implementation of dino loss with the original one. Firstly the output of the original Dino network is 65.536 dimensions whereas in our version the dimensionality is 384 or 128 depending on the output of the [CLS] token we wish to have. As is understood this can make a great difference in the effectiveness of that loss and it was decided for memory efficiency. Combining the losses was done by simply adding them. The whole methodology can be better seen the pseudocode next:

Methodology Pytorch Pseudocode

$vit1.params = vit2.params$
for $x\ in\ loader$: **do**
$x'_1, x'_2 = augmentations(x_1), augmentions(x_2)$
$s_1, s_2 = vit1(x'_1), vit2(x'_2)$
$e_1, e_2 = FC1(s_1), FC2(s_2)$
$g_1, g_2 = hyp(e_1), hyp(e_2)$
$loss = relaxedcontrasive(g1, g2) + dinoloss(g1, g2)$
$loss.backwards$
$update(vit1 + FC1)$
$(vit2 + FC2).params = l * (vit2 + FC2).params + (1 - l) * (vit1 + FC1).params$
$C = m * C + (1 - m) * cat([t1, t2]).mean(dim = 0)$

3.3.6 Data Augmentations

A key part of any deep learning framework are the augmentations which are introduced for the training of model. In our framework there are 2 types of training transformations as it was decided that in some experiments 2 different views for every image will be used.

These transformations are inspired by Dino transformations. The training transformations are complicated for every view including techniques such as Color Jitter, Gaussian Blur and Random Flip allowing the model to learn reperesentations invariant to transformations. This is one of the most important goals of modern self supervised methods [20]. On the contrary, validation dataset transformations are much fewer and simpler as the end goal is the ability of the model to create an embedding space which projects images with slight modifications instead of very strong ones.

Table 1: Augmentation list.

	Global transform1	Global transform2	Local transform	Validation transform
Color Jitter	✓	✓	✓	✗
Random Flip	✓	✓	✗	✗
Normalization	✓	✓	✓	✓
Gaussian Blur	✓	✓	✓	✗
Solarization	✗	✓	✗	✗
Resized image	✓	✓	✓	✓
Center Crop Size	✗	✗	✗	✓

3.3.7 Using Self Supervised instead of Supervised Pretraining on Imagenet

The final element of this methodology which needs to be addressed is the model which is used for pretraining. Most metric learning datasets are comparatively small and it is therefore very hard for any model to learn meaningful representations just by utilizing these datasets. This is why very few frameworks train their networks from scratch [64], [10] achieving at the same time noteworthy Recall@1 values.

Therefore we decided that we would be using pretraining for our model. However instead of initializing our model with Vit weights pretrained on Imagenet in a supervised way it was decided that we would initialize our model with pretraining from Ibot which is unsupervised. This was important for the framework for 3 reasons:

1. The goal of this thesis has been to create a competitive metric learning framework with no supervision. Initializing our model with weights which had been extracted using a supervised framework such as Vit would mean that at some point during the pipeline supervised learning would have been used.

2. Since no paper to our knowledge uses self supervised pretraining for unsupervised metric learning, we thought it would be interesting to see the effect self supervised pretraining would have on our model.

3. Using Ibot pretraining gave some pretty interesting results when tried on the State of the Art supervised metric learning method (previous section). These results were less impressive than the ones achieved using supervised pretraining but the margin was relatively small.

Therefore it was decided that for our experiments we would utilize the Ibot Vit s pretrained model which is publicly available in the official repository of the framework. This model has been trained for 800 epochs.

4. EXPERIMENTS

In this chapter we analyze the experiments conducted in this thesis.

4.1 Datasets

The proposed approach is tested on the most important benchmark datasets used for metric learning methods evaluation. CUB-200-2011 dataset [65] contains 11.788 bird species images. The first 100 classes with 5.994 images are used for the model training and the remaining 100 classes (5,794 images) are used for model validation.

Figure 14: CUB200-2011 dataset images.

CARS-196 dataset [66] contains 16.185 images split into 196 car classes. Data are typically split into 8.144 training images and 8.041 validation images.

Figure 15: CARS196 dataset images.

Stanford Online Products dataset [67] (SOP) consists of 120.053 images of 22.634 product images from eBay.com. The first 59.551 images with 11.318 products are used for training and the other 60.502 images of 11.316 products are used for validation purposes.

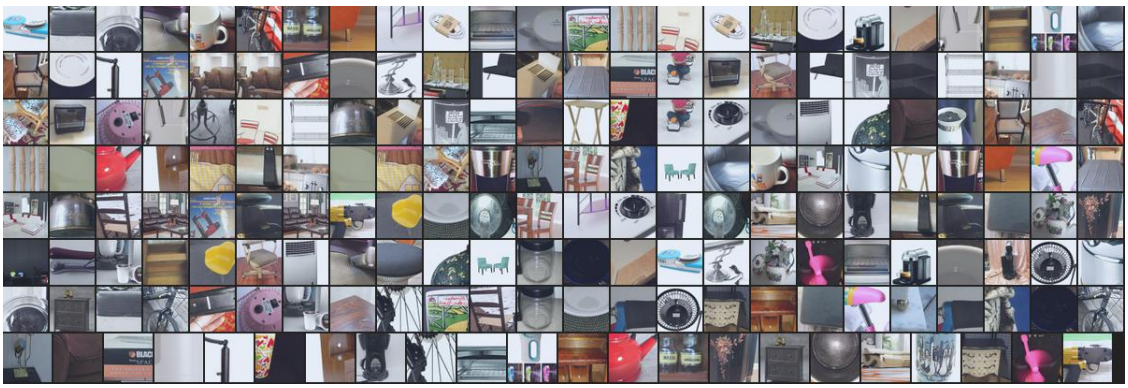


Figure 16: SOP dataset images.

4.2 Implementation Details

Hyperparameter tuning is an ingredient of key importance for the good performance of any model. Therefore, in the first metric learning experiment [1] which is a reproduction of paper's experiments with minimal changes, it was decided that hyperparameter tuning would be kept intact from the paper as it was supposed that the authors of this framework had done exhaustive search in order to conclude on the best possible hyperparameters. The only hyperparameter we had to change for computational reasons was batch size which became 120 from the original 900.

For our proposed methodologies we conducted extensive experiments in order to create a robust model setup and tune the hyperparameters. The setup we concluded on for most experiments utilized Lr 0.00003, Momentum 0.9998, Weight Decay 0.00001 and batch size 120 (150 for SOP dataset). The schedulers for momentum, Lr and weight decay are cosine schedulers and they have been analyzed in the previous chapter. All our experiments were conducted using 2 NVIDIA RTX A500 .

4.3 Evaluation Protocol

There is need for an evaluation metric in order to evaluate the quality of the embedding space. The two metrics which are used in most metric learning frameworks are the following:

Recall k: For every image of the validation dataset its k nearest neighbors are retrieved. If any image of the same class is retrieved among the samples then the metric receives 1 score otherwise it receives 0. Recall k is the average score of all the images in the test set. The values K are very important for the experiments as they are directly linked to the datasets size. This is the case since for a very large dataset chances are that the values of Recall@1 and recall@2 will not differ significantly. For the CARS196 and CUB200-2011 datasets which both have less than 20.000 images the values of k are usually 1, 2 and 4. For the SOP dataset with more than 100.000 these values are usually 1, 100, 1000.

NMI: NMI is a metric which gives the reduction in class entropy when cluster labels are known.

The problem with both of these metrics is that they require labels in order to be calculated which seems problematic when evaluating unsupervised methods. However since

all metric learning papers use Recall@1 as the metric of choice and due to the lack of any alternative this will be our metric of choice.

4.4 Supervised Metric Learning (Hyperbolic Metric Learning)

Table 2: Recall k for different methods on CUB200-2011 with size 384.

	R@1	R@2	R@4	R@8
Dino	80.0	87.8	92.7	95.4
Deit	76.4	85.4	91.5	95.0
Vit	84.5	91.1	94.2	96.3
Ibot	**81.2**	**87.8**	**92.6**	**95.5**

Table 3: Recall k for different methods on CUB200-2011 with size 128.

	R@1	R@2	R@4	R@8
Dino	77.3	86.2	91.6	94.9
Deit	73.5	83.2	89.8	93.7
Vit	82.6	89.8	93.9	96.1
Dino Sph	76.0	84.7	90.3	94.1
Deit Sph	73.3	82.4	88.7	93.0
Vit Sph	83.2	89.7	93.6	95.8
Ibot	**77.8**	**86.1**	**91.6**	**95.1**

Some of the table's recall metrics were calculated for the features of the transformer in a classic euclidean embedding space without the implementation of the Hyperbolic Embedding Space. They are included in order to compare the difference in the embedding's quality between the two embedding spaces. It is important to note that due to its higher dimensionality the embedding space of the first table has an inherent advantage over the embedding space of the second table. However, we observe that the use of hyperbolic embedding space significantly bridges the gap between the larger and the smaller embedding space. Additionaly the achieved recallk in our experiments is $\approx 1\%$ smaller than the recallk in the experiments of the paper. This difference is very small and can be fully explained by the reduced batch size we had to use.

Furthermore it is observed that using Vit gives the best results with the difference in the recall values been massive between this method and every other method if we use spherical embeddings (83.2 for Vit, 76.0 for Dino and 73.3 for Deit). However if hyperbolic embedding space is used the gap is bridged (82.6 for Vit 77.3 and 73.5 for Dino and Deit respectively). As mentioned Vit gives the best recall values with Dino achieving considerably smaller recall than Vit and Deit achieving the worst recalls of all the models.

It is very important to note that Dino is a self supervised method whereas Vit is a supervised method and therefore the difference in recall values can be justified despite the fact they use the same transformer Architecture. Deit achieves considerably smaller recalls than Vit despite the fact that they have almost the same amount of parameters (22milion for Deit, 23 million for Vit-s).

We conducted another experiment apart from reproducing the already existing ones using Ibot pretraining this time. We observe that using Ibot pretraining instead of Dino improves Recall@1 to 77.8 from 77.3 with 128 embedding size and to 81.2 from 80.0 with

384 embedding size. It is observed that Ibot achieves better recall values remaining fully unsupervised like Dino. This difference can be mainly attributed to the Masked Image Modelling utilized by Ibot.

4.5 Metric Learning without any Learning Stage

The next experiments presented are the ones conducted by evaluating the features extracted by 2nd stage ibot pretraining on CUB-200. A big number of experiments were conducted in order to manage to stabilize the model changing a plethora of standard neural network hyperparameters (such as Momentum, Lr and Batch Size) as well as some Ibot specific hyperparameters regarding Ibot data augmentations (Local Crop Scale, Global Crop Scale) and Ibot architecture such as shared head.

Table 4: Finetuning Ibot for metric learning.

Lr	Momentum	Batch Size		Local Crops Number	NMI	Recall@1
0.0005	0.996	128	0.04	0	60.2	37.6
0.0001	0.998	100	0.04	0	60.9	51.0
0.00005	0.999	110	0.04	0	53.2	56.6
0.0005	0.996	48	0.04	6	65.3	46.1
0.0001	0.999	90	0.04	6	54.2	55.4
0.0001	0.99996	90	0.04	6	46.9	66.3
0.0005	0.99996	90	0.04	10	50.0	64.4
0.00003	0.99996	90	0.04	10	46.4	66.1
0.00003	0.99996	90	0.04	10	46.4	66.2
0.00003	0.99996	90	0.4	10	46.4	66.2
0.00003	0.9999	90	0.4	10	48.8	66.3
0.00001	0.9999	90	0.4	10	49.1	64.5

The first hyperparameter we tried to stabilize was momentum as in the original Ibot it had a profound impact at the accuracy achieved by the model. We started of by using a relatively small momentum of 0.996. Despite the fact that NMI progressively increased Recall@1 decreased significantly over training. Therefore, we tried increasing the momentum in order to exploit the already good level of CUB200-2011 features learned from the Ibot pretraining on Imagenet. This, despite the fact that it helped the model stabilize its recall values during training does not allow any further training.

Lr is another very important hyperparameter. Since the task at hand is finetuning a pretrained model to a new dataset rather than training from scratch we chose to use a small Lr. We tried many different learning rates but they seemed to have minimum impact. Regarding batch size we tried sizes ranging from 90 up to 120 images per batch. Additionally we experimented with different local crops numbers ranging from 0 to 10.

The best recall values were achieved using a very aggressive learning rate along with a large number of local views. The best recall values which were achieved after the end of training were 66,3% which is smaller than the 66,6% Recall@1 we were able to achieve just by extracting the features of the transformer blocks and then evaluating on CUB200-2011 without any further training. It therefore became clear that using Ibot for metric learning purposes without any design changes to the model is impossible.

4.5.1 Unsupervised Metric Learning with Relaxed Contrastive Loss.

The next series of experiments presented are the results of the proposed framework. First, we present the experiments conducted by utilizing self distillation relaxed contrastive loss and the hyperbolic embedding space in order to finetune the model.

Table 5: CUB200-2011 Lr and Momentum tuning.

(a) Tuning Momentum.

	Embedding Size	Momentum	Recall@1
1st	512	0.99996	67.1
2nd	512	0.9999	67.4
3rd	512	0.9996	66.0
4th	512	0.99984	67.9
5th	512	**0.9998**	**68.1**

(b) Tuning Lr.

	Embedding Size	Lr	Recall@1
1st	512	0.00004	67.7
2nd	512	0.00001	66.3
3rd	512	0.00002	67.9
4th	384	0.00003	68.7
5th	128	**0.00003**	**68.3**

The first hyperparameter we tried to stabilize was again momentum. 5 different experiments were conducted for 100 epochs, with embedding size of 512 and different momentum values in order to decide which gave the best Recall@1. In the end a Momentum =0.9998 gave the best results (Recall@1 68.1) and it was decided to be kept that way. The second hyperparameter we tried to finetune after we stabilized momentum is Lr. During the Momentum finetuning we stabilized Lr value at 0.00003. We experimented with 4 different Lr values and the initial Lr =0.0003 gave the best results, therefore it was decided to keep it as the default Lr of all CUB200-2011 experiments.

We find that reducing the embedding size from 512 to 128 for this experiment improves the Recall@1 values achieved. This can be explained as the output of Vit attention blocks is 384 and therefore creating an embedding space with larger dimensionality can only degrade performance.

4.6 Unsupervised Metric learning with Relaxed Contrastive Loss and Hyperbolic Embedding

4.6.1 Hyperbolic Embedding and Freezing Layers

Next we examined if there is any merit in projecting the embedding vectors to the hyperbolic embedding space. It was also decided to freeze the first 2 of the transformer layers which are the pos drop and patch embed layers to find if it gives any boost to the performance.

Table 6: Comparison of hyperbolic-non hyperbolic embeddings with embedding size 128.

	Freeze Layers	Hyperbolic Embedding	R@1	R@2	R@4	R@8
Experiment 1	True	True	68.3	79.0	87.1	92.7
Experiment 2	False	True	67.3	78.2	87.0	92.4
Experiment 3	True	False	67.7	78.7	87.4	92.8
Experiment 4	False	False	67.4	78.5	87.0	92.8

We observe that if the first 2 layers are frozen the achievable Recall@1 increases by 0.4% when we use the euclidean embedding space and by 0.6% when we use hyperbolic

embedding space. Therefore it was decided that the first 2 transformer layers weights will not be updated during training. We observe that despite the fact that using the hyperbolic embedding space without freezing these layers gives no significant boost in performance, freezing the layers and using hyperbolic embedding space achieves 1% better Recall@1 values (experiment 1) than experiment 2 and 0.6% better Recall@1 than experiment 3. The benefits of using hyperbolic embedding space are therefore apparent.

4.6.2 Curvature of Hyperbolic Space

It has been discussed during the methodology analysis that curvature is the most important parameter of the hyperbolic embedding space. Therefore we conducted 4 different experiments with different curvature values to assess the differences in Recall@1 when tuning curvature differently.

Table 7: Experiments with the setup's curvature with embedding size 128.

	Hyperbolic Curvarture	R@1	R@2	R@4	R@8
Experiment 1	0.1	68.3	79.0	87.1	92.7
Experiment 2	0.3	67.2	78.3	87.1	92.8
Experiment 3	0.05	68.4	79.0	87.1	92.9
Experiment 4	0.01	68.6	79.8	87.5	92.8

The first experiment was conducted with the standard curvature used in most hyperbolic embedding papers (0.1) and it is included as a baseline. The second experiment used curvature 0.3. The Recall@1 it achieved is $\approx 1\%$ less than the one achieved with curvature 0.1. Experiments 3 and 4 which are conducted with smaller curvatures than the baseline achieve marginally better Recall@1 values of 0.1 and 0.3 % respectively. We observe that bigger values than 0.1 give worse performance, whereas smaller than 0.1 give slightly better performance. In order to be comparable with [1] and [61] we opted for 0.1 curvature value since the difference in Recall@1 with the other experiments was not significant.

4.6.3 Experiments with Different Augmentation Sets

Next we analyze the results achieved with the different augmentations sets. It is important to mention that all models were stabilized by the 100th epoch. However, it was observed that the models with no local or global crops reached the point where they stabilized their Recall@1 values by the 70th epoch, whereas the models with local or global crops stabilized by the 90th epoch. This displays the importance of local and crops and the increased learning capacity they provide.

Table 8: Experiments with different sets of augmentations.

	Dimensionality	R@1	R@2	R@4	R@8
1 global view	128	68.3	79.0	87.1	92.7
1 global view	384	68.7	79.4	87.4	92.8
1 global +10 local views	128	69.3	79.8	88.1	93.0
1 global +10 local views	384	70.5	81.1	88.5	93.5
2 global views	128	69.0	79.7	87.9	93.3
2 global views	384	69.5	80.3	87.9	93.3
2 global + 10 local views	128	69.7	80.4	88.0	93.3
2 global + 10 local views	384	71.0	81.1	88.3	93.6

It can be seen that the best results are achieved using both global and local crops in training which is expected if we factor in that more data allow better training. The experiments we will analyze have 384 dimensions. Some interesting observations which can be derived from this table are that in case we have 2 or one global augmentation the Recall@1 values differ only by 0.5%. The situation slightly shifts when the local crops are added with the difference in Recall@1 values been over 1%. It is also important to mention that the use of different augmentations adds 2.3 % to Recall@1 in the best case scenario which is not negligible.

The experiments with a combination of local and global crops achieve the best Recall@1 values and the local crops seem to consistently contribute more to the achievable Recall@1 than the second global crop. This is apparent if we consider that the model with local crops and 384 dimensionality achieves 70,5% whereas the same experiment with 2 global crops achieves 69.5%. For the next series of experiments we decided to use the augmentation strategy of 1 global and 10 local crops as we observed that there is no significant difference between using 1 or 2 global crops.

4.6.4 Combined Dino and Relaxed Loss

Next we tried to see if an extra self supervised loss (Dino loss) would fit in the general self supervised framework and allow further training. The 2 losses were combined and next we display the results for each one of these methods.

Table 9: Losses combination.

	loss	embedding size	R@1
1st	relaxed	384	70.5
2nd	relaxed + dino loss	384	68.1
3rd	relaxed + 0.5*dino loss	384	68.4
4th	0.5*relaxed + dino loss	384	67.3

We observe that relaxed contrastive loss alone gives the best Recall@1 of 70.5. The distillation loss achieves much worse Recall@1 than relaxed contrastive loss. The second best results are achleved with relaxed loss and 0.5*dino loss (Recall@1 68.4). The other 2 cumulative losses achieve much worse Recall@1. The best possible results are achievable using only relaxed contrastive loss. In our experiments from now on will be using only the relaxed contrastive loss.

4.6.5 Tuning in Cars Dataset

Next we will present the experiments we conducted to tune our hyperbolic unsupervised framework in order to train in the CARS196 dataset.

Table 10: Finetuning on CARS196 dataset (Red colour represents the experiments that collapsed by the 50th epoch.

	Lr	Momentum	Hyperbolic Plain/Curvature	R@1
first	0.00003	0.9999	Yes/0.1	31.2
second	0.00003	0.9998	Yes/0.1	**45.0**
third	0.00003	0.9998	No/0	39.4
fourth	0.00003	0.9998	No/0	41.2
fifth	0.000005	0.9998	No/0	41.1

From these experiments we extract 2 different conclusions

1. Despite the similar size of the CARS196 and CUB200-2011 dataset the achievable Recall@1 values of the first are far greater than the ones of the second. This unexpected tendency has already been observed in [11]. Of course, an immediate comparison between the 2 results is not possible because in this paper the authors only conducted inference in the CARS196, SOP, CUB200-2011 datasets with no training on the dataset.

2. The only setup that achieves constant training throughout 100 epochs is the hyperparameter setup for CUB200-2011 with a hyperbolic embedding layer at the end. In every other case the model collapsed after 50 epochs. This displays the importance of the hyperbolic embedding space and the strong hierarchical representations it achieves as a stabilizing factor in training.

4.6.6 Tuning the SOP dataset

The final dataset we will experiment on is SOP dataset.

Table 11: Finetuning on SOP dataset (Red colour represents the experiments that collapsed by the 50th epoch.

	Epochs	Lr	Momentum	Hyperbolic Plain/Curvature	R@1
first	50	0.000003	0.9998	Yes/0.1	21.8
second	50	0.000003	0.99996	Yes/0.1	**63.2**
third	35	0.000003	0.9999	Yes/0.1	41.1

In SOP dataset we could not conduct more thorough hyperparameter investigation, due to the size of the dataset and our inability to have a batch size larger than 300 samples. Initially we tried the same hyperparameter tuning as the other 2 datasets but the model collapsed. We had to decrease the Lr 10 times and also use a significantly larger momentum than the ones used for the other datasets in order to be able to train our model.

For the SOP dataset it is observed that our results are significantly better than the ones achieved in CARS196. However they lack behind the results of CUB200-2011 which will be further analyzed in the next sections.

4.7 Fair and Unfair Comparisons with State of the Art Methods

4.7.1 CUB

Next we will compare the results achieved for the CUB200-2011 dataset in the thesis to the results of some state of the art papers in unsupervised metric learning. This is the closest comparison to State of the Art methodologies which can be made. The first series of comparisons is ”fair” in terms of model’s size and the model used in the other experiments is Inception V2 [68]. The fairness of these comparisons has to do both with the size of our model which is considerably smaller than the models utilized in the other methods as well as the embedding size which is smaller than the ones utilized by them.

Table 12: Comparing our results on CUB200-2011 to State of the Art unsupervised methods.

Methodologies	Dim	Number of Parameters	R@1	R@2	R@4
UDML-SS	512	54 m	63.7	75.0	83.8
STML	512	54 m	68.0	78.8	86.4
OUR METHOD (1 view)	384	21 m	68.7	79.4	87.4
OUR METHOD (multicrop)	384	21 m	**70.5**	**81.1**	**88.5**

Our method achieves State of the Art results in unsupervised metric learning with a significantly smaller amount of trainable parameters than the other 2 methods. In this table we include both the results of the model with and without multicrop augmentation strategy. With multicrop augmentation strategy our model significantly outperforms the other State of the Art methods in unsupervised metric learning by 2.5% with our model having half the parameters their models have. Even without multicrop augmentation strategy our model marginally outperforms State of the Art by 0.7%.

Similar results are observed for R@2 and R@4 metrics with our method outperforming previous methods by more than 0.5% if multicrop augmentation strategy is not used. If multicrop augmentation strategy is used the difference increases to more than and 2%. It is also worth mentioning that shares relaxed contrastive loss with our methodology, however their framework includes another loss and it is significantly more complicated than ours.

Since our method is the first unsupervised method to our best knowledge to surpass 70% Recall@1 we decided to conduct an unfair comparison with the supervised metric learning frameworks some of which use vit-small architectures.

Table 13: Comparing our results on CUB200-2011 to State of the Art unsupervised methods.

	Dim	R@1	R@2	R@4	R@8
A-BIER	512	57.5	68.7	78.3	86.2
ABE	512	60.6	71.5	79.8	87.4
SM	512	56.0	68.3	78.2	86.3
XBM	512	65.8	75.9	84.0	89.9
HTL	512	57.1	68.8	78.7	86.5
MS	512	65.7	77.0	86.3	91.2
SoftTriple	512	65.4	76.4	84.5	90.4
Horde	512	66.8	77.4	85.1	91.0
Proxy-Anchor	512	68.4	79.2	86.8	91.6
NSoftmax	512	61.3	73.9	83.5	90.0
ProxyNCA++	512	69.0	79.8	87.3	92.7
IRTr	384	76.6	85.0	91.1	94.3
Hyperbolic Dino	384	80.9	87.6	92.4	95.6
Our Method	384	**70.5**	**81.1**	**88.5**	**93.5**

Our method surpasses all metric learning and image retrieval methods which do not utilize transformer architecture even without using labels. Our method surpasses [69] by 1.5% however when transformers are entered into the equation the situation changes drastically. IRT_r [70] surpasses our method by a significant margin of 5% and the dino proposed in [1] surpasses our method by 10%. Therefore it becomes apparent that using labels has a very significant influence in the achievable Recall@1 values. It is also worth mentioning that these methodologies surpass our own in R@2, R@4 and R@8 metrics but by a smaller margin than in Recall@1.

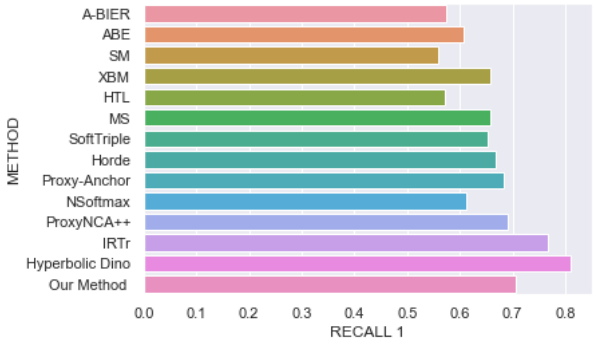


Figure 17: Recall@4, Recall@100 comparison between datasets.

Next we conduct some unfair comparisons with self supervised methods using 128 dim embedding size. These comparisons are unfair since our achitecture has 3 times more parameters than the one used for the experiments in all other frameworks (Googlenet) [71].

Table 14: Comparing our results on CUB200-2011 to State of the Art unsupervised methods with 128 dimensionality.

Methodologies	Dim	Number of Parameters	R@1	R@2	R@4
Exemplar	128	7 m	38.2	50.3	62.8
NCE	128	7 m	39.2	51.4	63.7
DeepCluster	512	7 m	42.9	54.1	65.6
ISIF	128	7 m	46.2	59.0	70.1
PSLR	128	7 m	48.1	60.1	71.8
ROUL	128	7 m	56.7	68.4	78.3
SAN	128	7 m	55.9	68.0	78.6
STML	128	7 m	59.7	71.2	81.0
OUR METHOD	128	21 m	**69.3**	**81.1**	**88.5**

It is apparent that our methodology significantly outperforms all methods which use googlenet. It outperforms STML [10] which is the state of the current State of the Art by 9.6%. The same tenedency is observed in recall@2 and recall@4 in favor of our methodology.

4.7.2 Cars

Next we conduct fair comparisons of our framework (384 embedding size) with STML (512 embedding size) trained on Inception V2 in CARS196 dataset.

Table 15: Comparing our results on CARS196 to State of the Art unsupervised methods.

	R@1	R@2	R@4
STML	66.2	74.5	81.9
OUR METHOD	**45.6**	**56.3**	**67.1**

Our methodology achieves significantly worse results than STML on CARS196 dataset with the difference in Recall@1 values been more than 20%. This will be further analyzed in the next section which includes a comparative study between the models

Next we conduct unfair comparisons with our framework trained and validated on the CARS196 dataset against most of the State of the Art methodologies. Again the architecture used is Googlenet and the dimensionality of the embeddings is 128

Table 16: Comparing our results on CARS196 to State of the Art unsupervised methods with 128 dimensionality.

	R@1	R@2	R@4
Deep Cluster	32.6	43.8	57.0
NCE	37.5	48.7	59.8
ISIF	41.3	52.3	63.6
PSLR	43.7	54.8	66.1
SAN	44.2	55.5	66.8
OUR METHOD	**45.0**	**56.3**	**67.0**
ROUL	45.0	56.9	68.4
STML	49.0	60.4	71.3

Our method surpasses most unsupervised metric learning methods experiments, with the exception of STML. It achieves the same Recall@1 as ROUL [13] however, ROUL

achieves slightly better results in recall@2 and recall@4. Despite the superior architecture we use compared to the one used in every other experiment our results are $\approx 4\%$ under the State of the Art. This displays that the State of the Art results we achieved in CUB200-2011 results cannot be translated to CARS196 dataset despite the fact that both datasets have similar size and in most papers tend to perform similarly.

4.7.3 SOP

The final series of experiments which will be presented are the ones conducted on the SOP dataset. The first comparison is a fair one between our method and the current State of the Art in terms of parameter number and dimensionality.

Table 17: Comparing our results on SOP to State of the Art unsupervised methods.

	R@1	R@2	R@4
STML	69.7	82.7	91.2
OUR METHOD	**63.4**	**78.2**	**88.4**

We observe that our methodology results are 6.3% less than the ones achieved in STML. However we observe that the difference in recall values is significantly smaller for the SOP dataset compared to CARS196. The final comparison will be the unfair we conducted in all other experiments comparing our method with State of the Art methods which utilize Googlenet architecture.

Table 18: Comparing our results on SOP to State of the Art unsupervised methods with 128 dimensionality.

	R@1	R@2	R@4
Deep Cluster	34.6	52.6	66.8
NCE	46.6	62.3	76.8
ISIF	48.9	64.0	78.0
PSLR	51.1	66.5	79.8
SAN	58.7	73.1	84.6
OUR METHOD	**63.2**	**77.9**	**88.2**
ROUL	53.4	68.8	81.7
STML	65.8	80.1	89.9

The Recall@1 achieved by our methodology is 2.6% smaller than the one achieved in the State of the Art paper. It is observed that our method is again outperformed despite the fact it has a lot more parameters than the State of the Art method.

4.7.4 Comprarative Study of the dataset recall values

The final part of this chapter is the comparative study of the ”fair” results between our method and the State of the Art. We therefore create cumulative bar diagrams of Recall@1, recall@2 (recall@10 for the SOP dataset) and recall@4 (recall@100 for SOP dataset) and then compare the results.

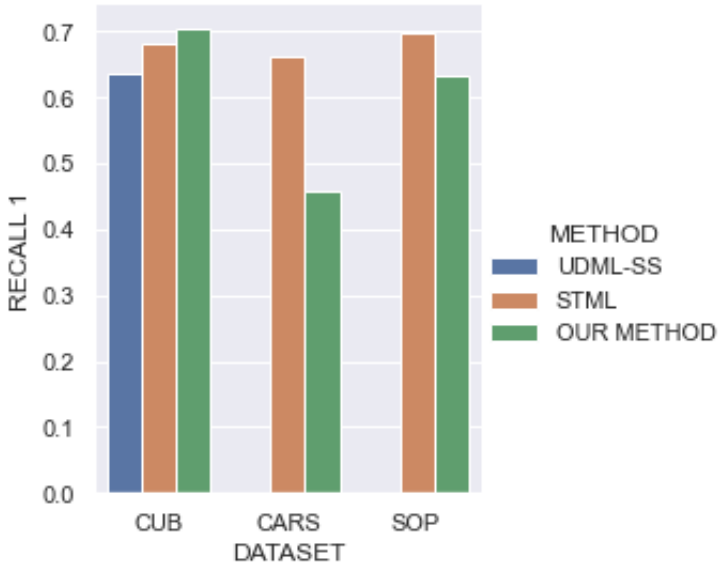


Figure 18: Recall@1 comparison between datasets.

The first bar plot we will analyze is the Recall@1 plot for the 3 datasets. As mentioned our method outperforms the other State of the Art methods in the CUB200-2011 dataset by 2.5%. However it is outperformed by STML by a significant margin of $\approx 20\%$ in the CARS196 dataset. The Recall@1 margin for the SOP dataset is significantly smaller than the one in CARS196 dataset been $\approx 6\%$ against our method.

When comparing the Recall@1 values in CUB200-2011 and CARS196 datasets we observe that in the CUB200-2011 dataset Recall@1 surpasses 70% whereas in the CARS196 dataset it is $\approx 45\%$. The biggest difference however is observed in the behaviour of the STML [10] and our method. In our method the difference between the datasets Recall@1 is $\approx 25\%$ whereas in STML the difference between datasets Recall@1 is $\approx 2\%$. Regarding SOP and CUB200-2011 the margin is considerably smaller than the one in CARS196. More specifically in our method CUB200-2011 outperforms SOP by $\approx 7\%$ whereas in STML SOP outperforms CUB200-2011 by $\approx 2\%$.

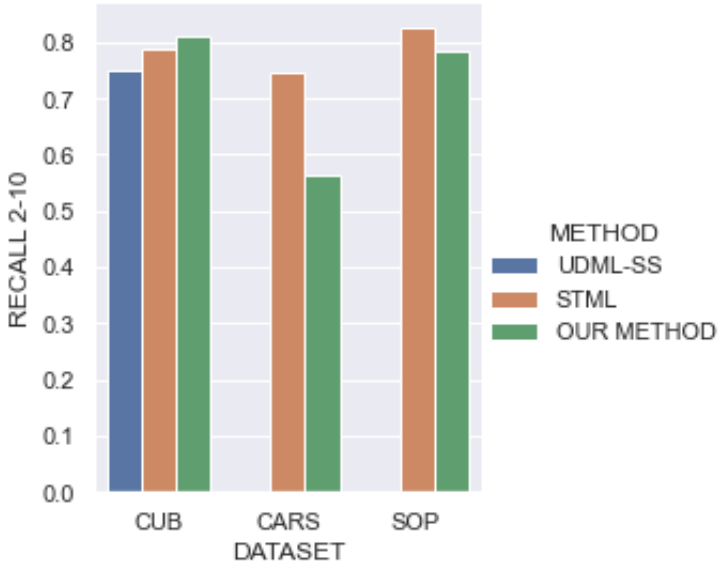


Figure 19: Recall@2, Recall@10 comparison between datasets.

The second bar plot we will analyze is the recall@2, recall@10 for the datasets. The difference in recall@2 between State of the Art and our method is 2.3% in favor of our method. On the contrary in CARS196 the difference is $\approx 18\%$ against our method. In SOP dataset the difference between our method and State of the Art is 4.5% against our method. It is apparent that in all cases recall@2 differences are smaller than Recall@1 differences.

The difference in recall@2 between CARS196 and CUB200-2011 datasets is again $\approx 20\%$ in our method. The difference between these datasets is 4.3% for the STML methods highlighting the inconsistency of results achieved by our method. In conclusion the difference between recall@4 of CUB200-2011 and recall@10 of SOP is $\approx 3\%$ in favor of CUB200-2011. The difference between these datasets in STML methodology is $\approx 4\%$ in favor of SOP. Generally the same incosistencies in datasets are observed in Recall@1 and recall@2 however it seems that the gap between our method and STML is slightly bridged in all 3 datasets.

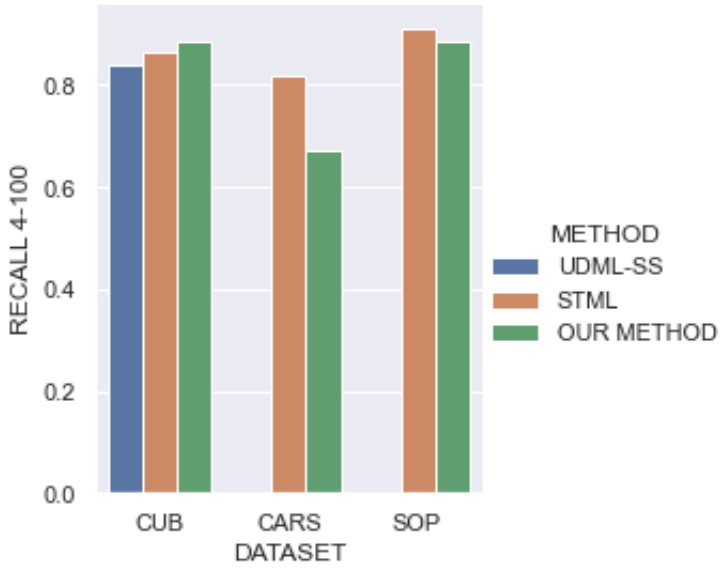


Figure 20: Recall@4, Recall@100 comparison between datasets.

In the third bar plot comparisons between recall@4, recall@10 are conducted. The difference in recall@4 between our method and STML $\approx 2\%$ in favor of our method. In CARS196 recall@4 changes in favor of STML by $\approx 15\%$. In the SOP dataset the difference in recall@4 and recall@100 is 3% in favor of STML. The difference for SOP dataset is considerably smaller than the ones in previous experiments.

The difference in recall@4 for CUB200-2011 and CARS196 dataset is again $\approx 21\%$ in favor of CUB200-2011 dataset. The difference between these datasets is $\approx 5\%$ in STML. The difference between CUB200-2011 and SOP is less than 1% in our method. The difference in STML is less than 5% in STML.

5. CONCLUSIONS AND FUTURE WORK

The breakthroughs in deep learning models along with the unprecedented amount of raw data which is collected today requires rethinking of the methods we have been utilizing to train our models. Deep learning models tend to require immense amounts of carefully labeled data during the training phase, but it is impossible for us to label all the aforementioned raw data and therefore utilize them. Over the past few years self supervised learning has allowed the training of robust models managing to totally avoid the use of labels. Therefore self supervised learning has opened a path for the utilization of big data in training our models

Supervised metric learning has achieved some impressive results over the past few years recognizing similar and disimilar objects even in famously difficult datasets such as CUB200-2011. However unsupervised metric learning seems to have been comparatively neglected with no papers utilizing the latest breackthroughs in AI during training. The need for the utilization of new architectures and new ideas for the replacement of labels becomes apparent.

In this thesis we presented a novel method for unsupervised metric learning based on a combination of ideas prevalent in self supervised learning. We displayed the potential of utilizing self supervised methods in unsupervised metric learning achieving State of the Art values in the CUB200-2011 dataset. We proved that these ideas despite originating from self supervised pretext tasks are very well integrated to metric learning tasks. Next we will assess the main contributions of our thesis.

1. We combined relaxed contrastive loss with self distillation achieving State of the Art Recall@1 values in the CUB200-2011 dataset.
2. We displayed that relaxed contrastive loss can be a robust alternative to pseudo-clustering [61], [12] for unsupervised metric learning.
3. We proved that hyperbolic embedding spaces allow better embedding quality in usnupervised metric learning as well as they do in supervised metric learning
4. We displayed the viability of using unsupervised pretrained models instead of their supervised counterparts.
5. We reaffirmed the potential of using transformers in supervised metric learning, and displayed their potential when used in unsupervised metric learning.
6. We showed that despite their capabilities transformers can be unstable when transfered between datasets achieving State of the Art recall values in CUB200-2011 while performing very poorly in CARS196 and relatively bad in SOP.

Regarding our future work the goal is to find more efficient ways to combine Dino loss with relaxed contrastive loss as well as finding clever ways to better utilize the different views created by the augmentations. Additionaly, we would wish to use supervised methods for pretraining in order to compare the results of training using unsupervised versus supervised pretraining methods. In conclusion we would like to use bninception instead of Vit transformer as our backbone architecture in order to be able to conduct immediate comparisons with the State of the Art methods.

6. APPENDIX

Next we are going to present the attention maps produced from our experiments and a series of experiments we conducted combining self supervised learning and distillation methods which did not fit it the main chapters of our thesis. It is important to mention that these are not the only experiments which we conducted in self supervised learning.

6.1 Attention Maps

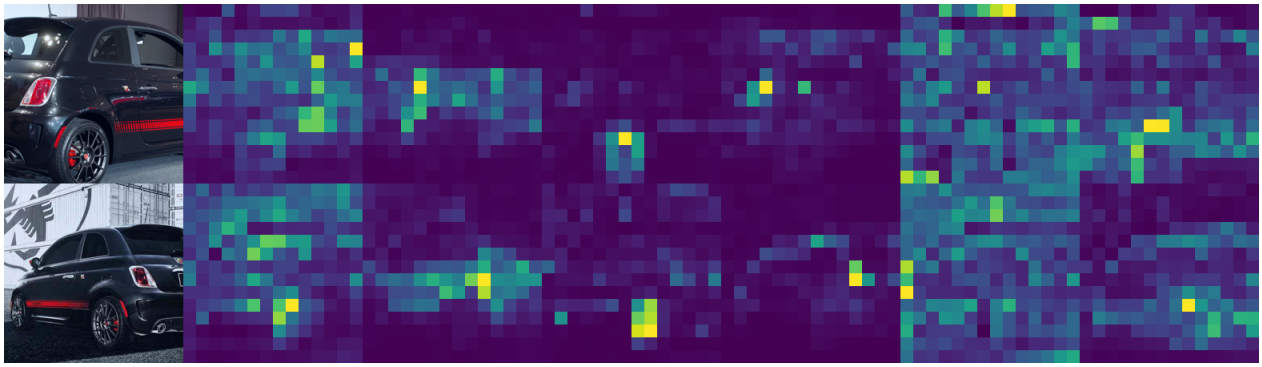


Figure 21: CARS196 attention maps.

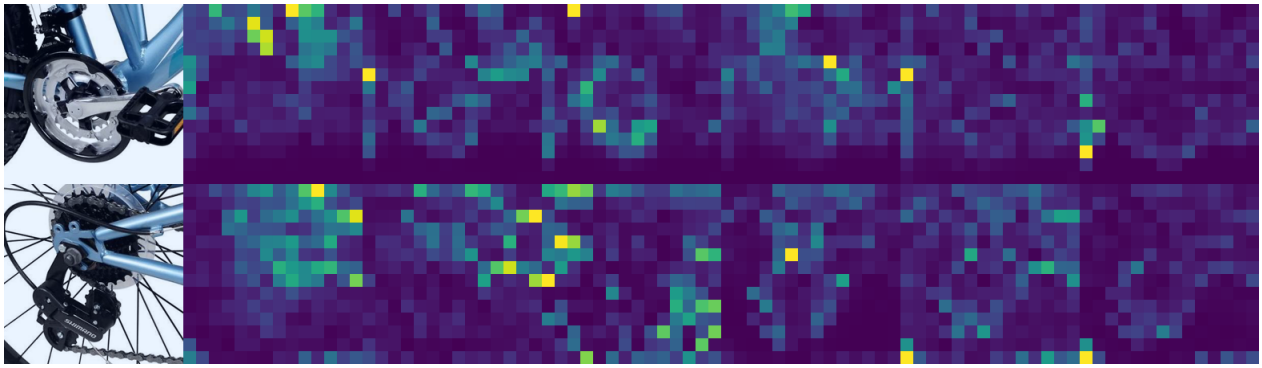


Figure 22: SOP attention maps.

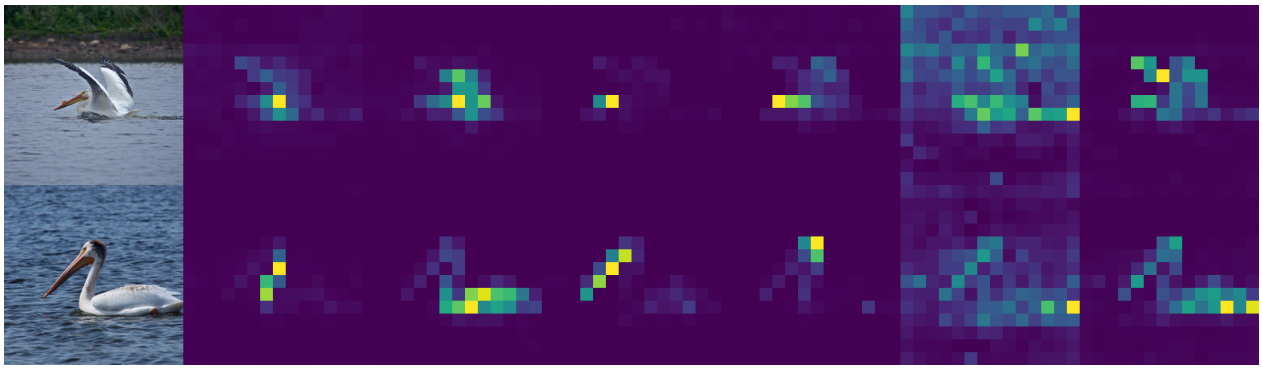


Figure 23: CUB200-2011 attention maps.

We observe that the attention maps which have been produced from the CUB200-2011 dataset have the best overall quality, with them reminding of segmentation maps. Especially in the second case the goose's beak is segmented very clearly. The goose's body is

also segmented very well. The first attention head attends the head the second the body the third the beak and the forth attends the head. The fifth attends to the whole image and the sixth attends and efficiently segments the whole body.

The attention maps of the CARS196 and the SOP dataset display significantly worse results than the CUB200-2011 dataset which can be expected considering the worse overall metrics of the models. In the case of CARS196 images the attention maps are segmented in a better way than the ones from SOP dataset. This has to do mostly with what the images display. The CARS196 images display a whole car, which is relatively easily segmentable while the SOP images display parts of bicycles which are relatively hard to seperate from the background.

6.2 Self Supervised learning Experiments

Next we will be displaying the experiments in self supervised learning.

6.2.1 Dataset

The dataset that was chosen in order to conduct the experiments is Cifar 10. The Cifar 10 images were collected by Alex Krizhevsky, Vinor Nair and Geofrey Hinton. This dataset was first introduced in [22] and was used to train a classifier on the top of Restricted Boltzman Machines (RBMs) [72], and Deep Belief Networks (DBNs). This dataset consists of 60000 images of 32x32x3 size. This dataset has 10 classes and it is divided in to training and testing data. Training data consist of 50000 images (5000 images per class) and Testing data consist of 10000 images (1000 images per class). This dataset was chosen for its relatively small size its adequate representation of all ten classes as well as its use in some recent self supervised learning papers [[19], [45]]

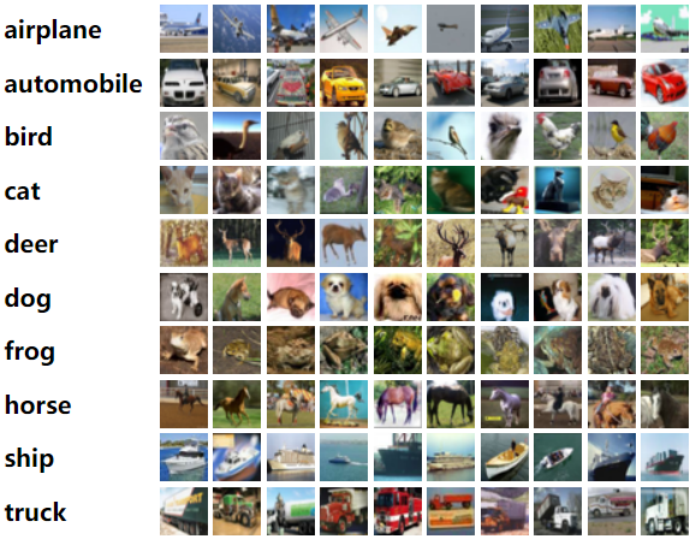


Figure 24: Images of Cifar 10.

6.2.2 Methodology

In this section we will briefly explain the methodology used for the self supervised learning experiments. The idea of our method was to compare the quality of features extracted by a network using rotnet to the quality of features extracted by using rotnet along with knowledge distillation. Therefore 2 different experiments will be presented:

1. The reproduction of the [19] experiments by swapping NIN architecture with Resnet 18 architecture and the Lr scheduler from multistep learning rate to Reduce on Plateau. More specifically every image is rotated 90, 180 and 270 and along with the original they are used as input of the neural network. The network consists of four convolutional layers for feature extraction and one linear layer which classifies each image to its respective rotation. Therefore the network's output is the rotation of every image. .
2. The combination of the aforementioned technique with online mutual learning of 2 networks. Two models are trained using the rotnet methodology while simultaneously distilling their knowledge to one another (online distillation). The hyperparameter tuning remains the same with and without distillation.

It is important to mention that like any method of self supervised learning we will evaluate the experiments after the training phase on a downstream task instead of the pretext tasks. This is important as better performance in the pretext task does not always imply better performance in the downstream task at hand. The network is either evaluated using a simple linear classifier trained on top of the features or using a convolutional classifier which tunes the images to the new task before classification. In the next table we will present the elements of the linear and the convolutional classifier:

Table 19: Presentation of the linear, convolutional classifier.

Convolutional Classifier	Linear Classifier
Convolutional Block1(numb feat, 192, 3)	Linear Layer(numb feat,200)
Convolutional Block2(192, 192, 1)	Batch Normalization
Convolutional Block3(192, 192, 1)	Linear Layer(200, 200)
Global Average Pooling	Batch Normalization
Linear Layer(192, 10)	Linear Layer(200, 10)

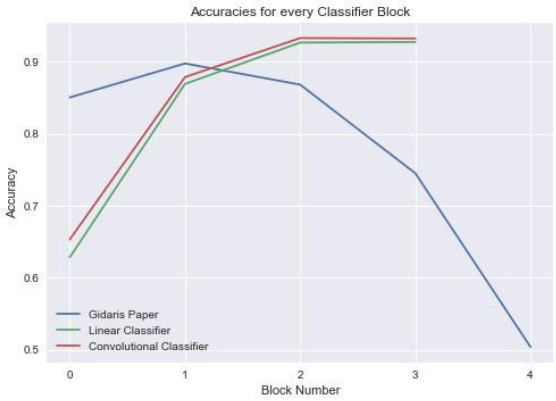
The next array displays the hyperparameters of the experiments:

Table 20: Hyperparameters.

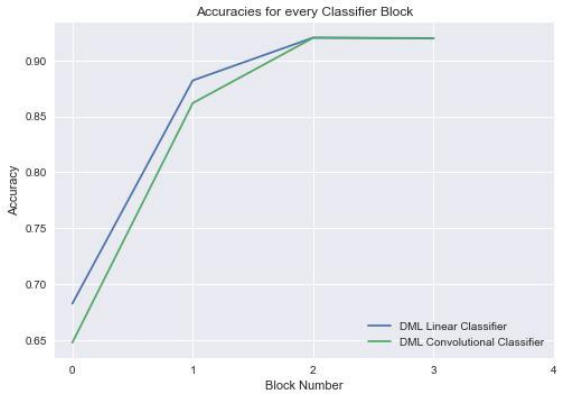
	Lr	Momentum	Weight Decay	Nesterov	Scheduler	
Pretext Task	0.1	0.9	5e-4	True	Reduce on Plateau	150
Convolutional Classifier	0.1	0.9	5e-4	True	Multiple Step Lr	30
Linear Classifier	0.1	0.9	5e-4	True	Multiple Step Lr	30

6.2.3 Experiments

Next we are going to display the diagrams of our experiments:



(a) Experiments in Rotnet



(b) Experiments with Rotnet and DML

Table 21: Hyperparameters.

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Paper Convolutional	85.0	89.8	86.8	74.5	50.4
Rotnet Convolutional	65.3	87.9	93.3	93.2	-
Rotnet Linear	62.8	86.9	92.7	92.7	-
DML Convolutional	64.8	86.2	93.3	93.2	-
DML Linear	68.3	86.9	92.6	92.7	-

We observe that the architecture of paper displays a different behaviour than Resnet architecture. More specifically we observe that the features from the first layer achieve the worst accuracies, the features from the second layer achieve relatively better accuracy and the features of the third achieve the best overall accuracies. Similar behaviour is observed for the experiments of Rotnet with distillation. This behaviour differs significantly from the one we observe in the original rotnet where the features of the second layer achieve the best accuracies and the features of the third and the forth layer achieve significantly worse metrics.

finally we include a confusion matrix with the results of the third rotnet layer.

Categories	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	sum	PA
airplane	937.0	3.0	18.0	6.0	1.0	2.0	5.0	4.0	20.0	8.0	1004.0	0.93
automobile	3.0	960.0	0.0	2.0	0.0	1.0	2.0	0.0	6.0	19.0	993.0	0.97
bird	14.0	0.0	898.0	17.0	11.0	11.0	16.0	7.0	2.0	0.0	976.0	0.92
cat	9.0	2.0	19.0	846.0	13.0	73.0	20.0	7.0	3.0	3.0	995.0	0.85
deer	5.0	1.0	26.0	24.0	945.0	15.0	6.0	18.0	1.0	1.0	1042.0	0.91
dog	2.0	0.0	20.0	84.0	11.0	886.0	4.0	26.0	1.0	0.0	1034.0	0.86
frog	3.0	0.0	16.0	10.0	6.0	3.0	945.0	1.0	0.0	0.0	984.0	0.96
horse	1.0	0.0	0.0	5.0	11.0	8.0	0.0	935.0	0.0	0.0	960.0	0.97
ship	17.0	6.0	2.0	3.0	1.0	0.0	1.0	1.0	959.0	6.0	996.0	0.96
truck	9.0	28.0	1.0	3.0	1.0	1.0	1.0	1.0	8.0	963.0	1016.0	0.95
sum	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	kappa:	0.92
UA	0.94	0.96	0.9	0.85	0.94	0.89	0.94	0.94	0.96	0.96	OA:	0.93

Figure 26: Confusion Matrix.

Analyzing the confusion matrix we observe that comission errors are in almost every case less than 10%. Two distinctive outliers are cat and dog categories. More specifically the cat category presents comission error 15% and dog category presents comission error 11%. This is interesting considering the fact that cats and dogs are 2 animals which look quite similar unlike the other objects in the dataset. Similar tendencies are observed with the omission error of the cat been 15% and the comission error been 14%.

It is observed that out of 1000 cats 84 are classified by mistake as dogs. Similarly out of 1000 dogs 73 are classified as dogs. Therefore the two categories seem to be easily confused.

ABBREVIATIONS - ACRONYMS

GPU	Graphical Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Networks
MLP	Multi-layer Perceptron

BIBLIOGRAPHY

[1] A. Ermolov, L. Mirvakhabova, V. Khrulkov, N. Sebe, and I. Oseledets, "Hyperbolic vision transformers: Combining improvements in metric learning," 2022.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[3] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

[4] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers amp; distillation through attention," 2020.

[5] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. L. Yuille, and T. Kong, "ibot: Image BERT pre-training with online tokenizer," *CoRR*, vol. abs/2111.07832, 2021.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[9] S. Kim, D. Kim, M. Cho, and S. Kwak, "Embedding transfer with label relaxation for improved metric learning," 2021.

[10] S. Kim, D. Kim, M. Cho, and S. Kwak, "Self-taught metric learning without labels," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[11] I. Kakogeorgiou, S. Gidaris, B. Psomas, Y. Avrithis, A. Bursuc, K. Karantzalos, and N. Komodakis, "What to hide from your students: Attention-guided masked image modeling," *CoRR*, vol. abs/2203.12719, 2022.

[12] Y. Li, S. Kan, and Z. He, "Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss," 2020.

[13] S. Kan, Y. Cen, Y. Li, V. Mladenovic, and Z. He, "Relative order analysis and optimization for unsupervised deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13999–14008, June 2021.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[17] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," 2019.

[18] Z. Zhang and M. R. Sabuncu, "Self-distillation as instance-specific label smoothing," 2020.

[19] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," 2018.

[20] I. Misra and L. van der Maaten, "Self-supervised learning of pretext-invariant representations," 2019.

[21] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304, JMLR Workshop and Conference Proceedings, 2010.

[22] C. Doersch and A. Zisserman, “Multi-task self-supervised visual learning,” 2017.

[23] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” 2016.

[24] W. Zou, S. Zhu, K. Yu, and A. Ng, “Deep learning of invariant features via simulated fixations in video,” *Advances in neural information processing systems*, vol. 25, 2012.

[25] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with convolutional neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.

[26] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658, 2015.

[27] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4320–4328, 2018.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.

[29] A. Canziani and Y. Lecun, “Week 2 – lecture: Stochastic gradient descent and backpropagation.”

[30] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[31] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[32] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten digit recognition with a back-propagation network,” *Advances in neural information processing systems*, vol. 2, 1989.

[33] A. Canziani and Y. Lecun, “Week 3 – lecture: Convolutional neural networks.”

[34] A. Canziani and Y. Lecun, “Week 3 – practicum: Natural signals properties and cnns.”

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[36] P. Zhou, Y. Zhou, C. Si, W. Yu, T. K. Ng, and S. Yan, “Mugs: A multi-granular self-supervised learning framework,” 2022.

[37] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[38] R. H. M. R. F. J. H. Yann LeCun, Sumit Chopra, “A tutorial on energy based learning.” University Lecture, 2000.

[39] A. Canziani and Y. Lecun, “Week 10 – lecture: Self-supervised learning (ssl) in computer vision (cv).”

[40] Y. LeCun and I. Misra, “Self-supervised learning: The dark matter of intelligence,” 2021.

[41] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” *arXiv preprint arXiv:1605.09782*, 2016.

[42] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[43] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[44] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, “Bootstrap your own latent-a new approach to self-supervised learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 21271–21284, 2020.

[45] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, pp. 1597–1607, PMLR, 2020.

[46] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

[47] A. Bellet, A. Habrard, and M. Sebban, “A survey on metric learning for feature vectors and structured data,” 2013.

[48] K. Musgrave, S. Belongie, and S.-N. Lim, “A metric learning reality check,” 2020.

[49] G. E. Hinton, “Deep belief networks,” *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[50] A. Schumann and R. Stiefelhagen, “Person re-identification by deep learning attribute-complementary information,” *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1435–1443, 2017.

[51] F. Chollet, *Deep learning with Python*. Simon and Schuster, 2021.

[52] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *Advances in neural information processing systems*, vol. 27, 2014.

[53] O.-E. Ganea, G. Bécigneul, and T. Hofmann, “Hyperbolic neural networks,” 2018.

[54] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond euclidean data,” *IEEE Signal Processing Magazine*, vol. 34, pp. 18–42, jul 2017.

[55] M. Nickel and D. Kiela, “Poincaré embeddings for learning hierarchical representations,” 2017.

[56] Y. Li, S. Kan, and Z. He, “Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss,” 2020.

[57] J. Ba and R. Caruana, “Do deep nets really need to be deep?,” *Advances in neural information processing systems*, vol. 27, 2014.

[58] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[59] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, “Be your own teacher: Improve the performance of convolutional neural networks via self distillation,” 2019.

[60] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky, “Hyperbolic image embeddings,” 2019.

[61] J. Yan, L. Luo, C. Deng, and H. Huang, “Unsupervised hyperbolic metric learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12465–12474, June 2021.

[62] Y. Kilcher, “Dino: Emerging properties in self-supervised vision transformers (facebook ai research explained).”

[63] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” 2016.

[64] M. Ye, X. Zhang, P. C. Yuen, and S.-F. Chang, “Unsupervised embedding learning via invariant and spreading instance feature,” 2019.

[65] M. T. W. C. S. F. B. S. P. P. Welinder P., Branson S., “Caltech-ucsd birds 200,” 2010.

[66] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization,” in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, (Sydney, Australia), 2013.

[67] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, “Deep metric learning via lifted structured feature embedding,” 2015.

[68] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015.

[69] E. W. Teh, T. DeVries, and G. W. Taylor, “Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis,” 2020.

[70] A. El-Nouby, N. Neverova, I. Laptev, and H. Jégou, “Training vision transformers for image retrieval,” 2021.

[71] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” 2014.

[72] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.