

# Few-shot Deep Learning Algorithms for Image Classification

*Konstantinos Ioannis Tzevelekakis*

Thesis submitted in partial fulfillment of the requirements for the  
*Masters' of Science degree in Computer Science and Engineering*

University of Crete  
School of Sciences and Engineering  
Computer Science Department  
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Assistant Prof. *Nikos Komodakis*

---

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Applied and Computational Mathematics (IACM).





UNIVERSITY OF CRETE  
COMPUTER SCIENCE DEPARTMENT


**Few-shot Deep Learning Algorithms for Image Classification**


Thesis submitted by  
**Konstantinos Ioannis Tzevelekakis**  
in partial fulfillment of the requirements for the  
Masters' of Science degree in Computer Science

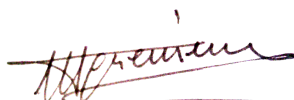
THESIS APPROVAL

Author:   
Konstantinos Ioannis Tzevelekakis

Committee approvals:   
Nikos Komodakis  
Assistant Professor, Thesis Supervisor

  
Yannis Stylianou  
Professor, Committee Member

  
Yannis Pantazis  
Researcher, Committee Member

Departmental approval:   
Polyvios Pratikakis  
Assistant Professor, Director of Graduate Studies

Heraklion, March 2023



# Few-shot Deep Learning Algorithms for Image Classification

## Abstract

Deep learning, a thriving field of machine learning, has witnessed an unprecedented revolution during the last decade. The powerful idea of hierarchical representation learning combined with the abundance of data the digital era effortlessly provides has led to breathtaking achievements in numerous scientific fields. Nevertheless, applications exist where a plethora of annotated training data is not available due to privacy restrictions, annotation difficulties, or prohibitive costs. Developing deep learning approaches that can be effective in such low-data regime scenarios is still a largely open problem.

In this work we consider such a low-data regime scenario for the problem of image classification, which is fundamental problem of Computer Vision. In the literature this is a setting also known as few-shot visual learning. In this case, given only a very small set of annotated images representing the available categories (e.g. even a single annotated image per category), the correct classification of an unlabeled image set is required. A common approach, termed metric learning, is to project both sets on a space, where samples are clustered with respect to their categories, in order to classify them using a similarity metric.

Following the metric learning paradigm, we propose a methodology that utilizes deep embedding functions to project the samples on the embedding space. To implement these embedding functions, we combine the representation power of vision transformers, a state-of-the-art deep learning architecture, amplified by employing pre-trained self-supervised foundation models. Undoubtedly, a few-shot learning algorithm should harness every bit of available information from the annotated data to be effective under this low data regime. Hence, instead of just incorporating prior knowledge, encoded in the embedding functions parameters, we additionally exploit the information exchange between those functions. Specifically, we conduct a case study that can be summarized in two main questions; (i) Is an exchange of information between the embedding functions beneficial for the problem at hand? (ii) In what way this exchange of information can be established?

In an attempt to answer these questions, we propose three main methods. These are namely ParallelVits, ParallelVits+Encoder, and BlendedVits. ParallelVits method undertakes the role of a performance baseline since it restricts the information flow between the embedding functions, whereas the rest of the methods enable information exchange by leveraging the flexibility of vision transformers architecture. Moreover, several hyper-parameters of the employed meta-learning framework, the neural network architectures, and the aforementioned methods have been put under scrutiny. The evaluation of our method has led to some interesting findings as well as to very promising experimental results, leading to near state-of-the-art performance in the miniImageNet dataset.



# Αλγόριθμοι Βαθιάς Μάθησης για την Κατηγοριοποίηση Εικόνων με λίγα Παραδείγματα

## Περίληψη

Η Βαθιά Μάθηση, ένα επιτυχημένο παρακλάδι της Μηχανικής Μάθησης, στο οποίο έχει συντελεστεί μια άνευ προηγουμένου επανάσταση την τελευταία δεκαετία. Η ισχυρή ιδέα της μάθησης ιεραρχικών αναπαραστάσεων σε συνδυασμό με την πληθώρα δεδομένων, που η ψηφιακή εποχή παρέχει με ευκολία, έχει οδηγήσει σε συναρπαστικά επιτεύγματα σε πολλούς επιστημονικούς τομείς. Ωστόσο, υπάρχουν εφαρμογές όπου δεν είναι διαθέσιμα επαρκή επισημειωμένα δεδομένα εκπαίδευσης, λόγω των περιορισμών της προσιμότητας της ιδιωτικότητας, των δυσκολιών επισημείωσης ή και του απαγορευτικού κόστους δημιουργίας ή χρήσης τους.

Η ανάπτυξη αποτελεσματικών προσεγγίσεων βαθιάς μάθησης, για τέτοια σενάρια μη επαρκών δεδομένων, παραμένει ένα ανοιχτό πρόβλημα. Στην εργασία αυτή εξετάζουμε ένα τέτοιο σενάριο μη επαρκών δεδομένων, για το πρόβλημα της ταξινόμησης εικόνων, πρόβλημα θεμελιώδες για την Υπολογιστική Όραση. Στην βιβλιογραφία αυτό το πρόβλημα αναφέρεται και ως οπτική μάθηση με λίγα παραδείγματα. Συγκεκριμένα, απαιτείται η σωστή ταξινόμηση μη επισημειωμένων εικόνων, παρέχοντας ελάχιστες (ακόμα και μόνο μία) αντιπροσωπευτικές εικόνες για κάθε κατηγορία. Μια συνήθης προσέγγιση, που φέρει την ονομασία μετρική μάθηση, είναι η προβολή των εικόνων σε έναν χώρο στον οποίο οι εικόνες διαμερίζονται ανάλογα με την κατηγορία στην οποία ανήκουν με την χρήση κάποιας μετρικής ομοιότητας.

Υιοθετώντας αυτή την προσέγγιση, προτείνουμε μία μεθοδολογία κατά την οποία η προβολή των εικόνων στον χώρο αναπαραστάσεων γίνεται με τη χρήση συναρτήσεων βαθιάς μάθησης. Συγκεκριμένα, χρησιμοποιούμε την υπερσύγχρονη αρχιτεκτονική βαθιάς μάθησης των μετασχηματιστών όρασης, σε συνδυασμό με την χρήση προεκπαιδευμένων μοντέλων με αυτοεπίβλεψη. Καθώς οι αλγόριθμοι διαθέτουν ελάχιστο αριθμό παραδειγμάτων και προκειμένου να είναι αποτελεσματικοί, θα πρέπει να αξιοποιούν κάθε διαθέσιμη πληροφορία από τα επισημειωμένα δεδομένα. Έτσι, εκτός από την χρήση της πληροφορίας που βρίσκεται αποθηκευμένη στις παραμέτρους των συναρτήσεων βαθιάς μάθησης, η ανταλλαγή της πληροφορίας μεταξύ των ίδιων των συναρτήσεων θα μπορούσε επίσης να αξιοποιηθεί. Με αυτό το σκεπτικό προχωρήσαμε στην διερεύνηση δύο κεντρικών ερωτημάτων: (α) Μπορεί στο συγκεκριμένο πρόβλημα, η ανταλλαγή της πληροφορίας μεταξύ των συναρτήσεων προβολής να συμβάλει θετικά; (β) Με ποιο τρόπο μπορεί να επιτευχθεί αυτή η ανταλλαγή της πληροφορίας;

Σε μια προσπάθεια να απαντήσουμε τα ερωτήματα αυτά, προτείνουμε τρεις μεθόδους. Αυτές είναι οι ParallelVits, η ParallelVits+Encoder και η BlendedVits. Η ParallelVits δεν επιτρέπει την ανταλλαγή της πληροφορίας μεταξύ των συναρτήσεων προβολής αποτελώντας την βάση αναφοράς. Αντιθέτως, οι υπόλοιπες μέθοδοι την επιτρέπουν με διαφορετική προσέγγιση η κάθε μια, εκμεταλλευόμενες την ευελιξία της υιοθετούμενης αρχιτεκτονικής. Επιπλέον, για τις προτεινόμενες μεθόδους, αλλά και για τις συναρτήσεις βαθιάς μάθησης, έχει διεξαχθεί εκτεταμένη αναζήτηση στο πεδίο τιμών διαφόρων υπέρ-παραμέτρων. Η αξιολόγηση των προτεινόμενων μεθόδων

οδήγησε τόσο στην ανακάλυψη σημαντικών ευρημάτων, όσο και σε πολλά υποσχόμενα πειραματικά αποτελέσματα, συγκρίσιμα με αυτά των πρωτοπόρων μεθόδων, στο σύνολο εικόνων του miniImageNet.



## Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επόπτη καθηγητή μου κο Νίκο Κομοντάκη για την καθοδήγηση και την υποστήριξή του καθόλη την διάρκεια της μεταπτυχιακής μου εργασίας. Οι συμβουλές του και οι προτάσεις του ήταν καθοριστικές για την διαμόρφωσή της. Επιπλέον, ιδιαίτερα σημαντική ήταν και η συνεισφορά του Σπύρου Γίδαρη, συνεργάτη του κο Κομοντάκη, με τον οποίο είχαμε πολυάριθμες συζητήσεις για την κατεύθυνση των πειραμάτων. Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένεια μου που με στήριξε με κάθε τρόπο σε όλη την διάρκεια των σπουδών μου.



*στους γονείς μου*



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>5</b>
2.1 Image classification . . . . .	5
2.2 Convolutional Neural Networks . . . . .	6
2.2.1 Overview . . . . .	6
2.2.2 Structure . . . . .	6
2.2.3 Convolution operation . . . . .	6
2.2.4 Notable Properties . . . . .	7
2.3 Vision Transformers . . . . .	7
2.3.1 Overview . . . . .	7
2.3.2 Structure . . . . .	7
2.3.3 Self-Attention . . . . .	8
2.3.4 Notable Properties . . . . .	9
2.4 Foundation models and DINO . . . . .	10
2.5 Few-shot learning . . . . .	11
2.5.1 Definition of FSL . . . . .	11
2.5.2 Core issue in FSL . . . . .	12
2.5.3 Taxonomy . . . . .	13
2.5.3.1 Data . . . . .	13
2.5.3.2 Model . . . . .	13
2.5.3.3 Algorithm . . . . .	14
2.6 Meta-learning . . . . .	14
2.6.1 FSL image classification framework . . . . .	14
2.7 Embedding learning for Image classification . . . . .	15
2.7.1 Overview . . . . .	15
2.7.2 Description . . . . .	15
2.7.3 Related Work . . . . .	16

2.7.3.1	Siamese Nets . . . . .	17
2.7.3.2	Matching Nets . . . . .	17
2.7.3.3	Prototypical Nets . . . . .	18
2.7.3.4	Relation Net . . . . .	19
2.7.4	Related Work based on VITs . . . . .	20
2.7.4.1	Cross Transformers . . . . .	20
2.7.4.2	PMF . . . . .	21
2.7.4.3	Hyperbolic Vision Transformers . . . . .	22
<b>3</b>	<b>Methodology</b>	<b>25</b>
3.1	Overview . . . . .	25
3.2	Meta-learning framework . . . . .	25
3.3	Case Study . . . . .	26
3.4	Embedding Function . . . . .	27
3.5	Proposed Methods . . . . .	28
3.5.1	ParallelVits method . . . . .	28
3.5.2	ParallelVits+Encoder method . . . . .	29
3.5.3	BlendedVits method . . . . .	31
3.6	Methods variations . . . . .	32
3.6.1	Attention Masks . . . . .	32
3.6.2	Cross Attention . . . . .	34
3.6.3	Auxiliary Losses . . . . .	35
3.6.4	Incorporation of image patches . . . . .	36
3.7	Other explored factors . . . . .	37
3.7.1	Similarity Metrics . . . . .	37
3.7.2	Model’s weights state . . . . .	38
3.7.3	Optimal training parameter set . . . . .	38
3.7.4	Artificially increase the number of shots . . . . .	39
3.7.5	Training with more classes . . . . .	39
3.7.6	Fine-tuning at meta-test time . . . . .	39
3.8	Datasets . . . . .	40
3.9	Implementation details . . . . .	40
3.9.1	Network architecture . . . . .	40
3.9.2	Classification Head . . . . .	40
3.9.3	Training details . . . . .	40
3.10	Workflow . . . . .	41
<b>4</b>	<b>Evaluation and Conclusions</b>	<b>43</b>
4.1	Evaluation details . . . . .	43
4.2	Evaluation of the proposed methodology . . . . .	44
4.2.1	Overview . . . . .	44
4.2.2	Main hyper-parameters . . . . .	44
4.2.2.1	Weights states . . . . .	44
4.2.2.2	Similarity Measures . . . . .	45

4.2.2.3	Baseline performance . . . . .	45
4.2.3	<i>BlendedVits</i> . . . . .	45
4.2.3.1	Concatenation block . . . . .	45
4.2.3.2	Attention mask settings . . . . .	46
4.2.3.3	Cross-attention . . . . .	47
4.2.3.4	Image patches . . . . .	47
4.2.4	<i>ParallelVits+Encoder</i> . . . . .	49
4.2.4.1	Encoder blocks . . . . .	49
4.2.4.2	Auxiliary losses . . . . .	50
4.2.4.3	Cross-attention . . . . .	50
4.2.4.4	Image patches . . . . .	51
4.2.5	Conclusions and Best settings . . . . .	51
4.2.6	Other experiments . . . . .	53
4.3	Comparison with other works . . . . .	54
<b>5</b>	<b>Discussion</b>	<b>57</b>
5.1	Summary . . . . .	57
5.2	Future Work . . . . .	58
	<b>Bibliography</b>	<b>59</b>





# List of Tables

3.1	The explored similarity measures. . . . .	38
4.1	The effect of weight states on the proposed methods. . . . .	44
4.2	The effect of similarity choice on our methods. . . . .	45
4.3	The baseline performance of our methods. . . . .	45
4.4	The effect of weight states on the proposed methods. . . . .	46
4.5	Attention mask settings on <i>BlendedVits</i> method using <i>cat block 9</i> . . . . .	47
4.6	Cross-attention in <i>BlendedVits</i> method. . . . .	47
4.7	Cross-attention coupled with mask setting 2 for <i>BlendedVits</i> method. . . . .	47
4.8	The image patches modality combined with <i>BlendedVits</i> method. . . . .	48
4.9	The image patches modality combined with attention mask setting 4. . . . .	49
4.10	The effect of pre-trained blocks selection on the <i>ParallelVits+Encoder</i> method. . . . .	49
4.11	The effect of auxiliary losses on the <i>ParallelVits+Encoder</i> method. . . . .	50
4.12	Cross-attention in <i>ParallelVits+Encoder</i> method. . . . .	50
4.13	The image patches modality combined with <i>ParallelVits+Encoder</i> method. . . . .	51
4.14	Our best settings on MiniImageNet’s validation set. . . . .	52
4.15	The effect of sampling more classes on our methods performance. . . . .	53
4.16	The effect of more samples per category (i.e. shots) on the baseline performance of our methods. . . . .	53
4.17	Our best settings on MiniImageNet’s test set. . . . .	54
4.18	Comparison of our best settings with other related works. . . . .	55



# List of Figures

1.1	Representation learning. Taken from [20]. . . . .	2
2.1	VIT (left side) and Encoder block (right side). Taken from [76]. . .	8
2.2	Scaled dot-product attention (left side) and multi-head attention (right side). Taken from [34]. . . . .	10
2.3	Self-distillation with no labels. Taken from [73]. . . . .	11
2.4	Self-attention from a vision transformer trained with no supervision. Taken from [73]. . . . .	12
2.5	Task-invariant (left side) and hybrid (right side) embedding models. Taken from [66]. . . . .	15
2.6	Siamese Nets: Training and testing procedure. Taken from [14]. . .	16
2.7	Matching Nets. Taken from [24]. . . . .	17
2.8	ProtoNets: few-shot (left side) and zero-shot (right side) settings. Taken from [31]. . . . .	18
2.9	Relation Net. Taken from [43]. . . . .	19
2.10	Cross Transformers. Taken from [59]. . . . .	20
2.11	PMF. Taken from [85]. . . . .	21
2.12	Hyperbolic Vision Transformers. Taken from [83]. . . . .	22
3.1	An example of 2-way 3-shot tasks during training and testing phases. For each task two classes are sampled for both sets and three samples per class for the support sets. . . . .	26
3.2	Task-invariant (left) vs Hybrid embedding learning (right). The support and query sets are embedded with the embedding functions $g$ and $f$ respectively. Then, the query sample is classified to the category of the support sample which is closer in the embedding space. At the right side there is a bidirectional arrow that indicates the information flow between $g$ and $f$ in the hybrid paradigm. . . .	27

3.3	A simplified illustration of the embedding generation using ViT. An image is split into patches which are linearly projected using a fully connected layer for each patch. Then a randomly initialized representation token is added to the sequence of the linear projected patches. Afterwards, the feature vector is refined through the ViT blocks and finally the resulted representation token corresponds to the image embedding. . . . .	28
3.4	<i>ParallelVits</i> method: an outline. The support and query sets are embedded independently and then the similarity scores between them are computed for the classification process to proceed. . . . .	29
3.5	<i>ParallelVits</i> method: an example. The representation tokens are vectors in the embedding space. The query samples are classified to the classes of support tokens that are closer in the embedding space.	29
3.6	<i>ParallelVits+Encoder</i> method: an outline. The support and query samples are initially embedded independently by the two illustrated ViTs. Then the resulting embeddings are processed by a transformer encoder that outputs the contextualized embeddings. Finally, the similarities between them are calculated. . . . .	30
3.7	<i>ParallelVits+Encoder</i> method: an example. From the ViTs output the patch tokens (painted in gray) are discarded. The representation tokens are concatenated in the token dimension and then are fed to the transformer encoder. Finally, query samples are classified to the categories of support tokens that are closer in the embedding space.	30
3.8	<i>BlendedVits</i> method: an outline. The support and query sets are embedded independently up to a point. After that, support and query representation tokens are jointly refined. The process proceeds with the calculation of similarity scores for the classification of the query samples. . . . .	31
3.9	<i>BlendedVits</i> method: an example. Both support and query sets are embedded independently up to a point, using the ViT architecture. Then the support representation tokens are concatenated to each query feature vector at the cat block of the query network. After the concatenation each query is jointly processed with all the support representation tokens. At the end of the process, each query sample is classified with respect to the support representation tokens that it was contextualized with. . . . .	32
3.10	<i>BlendedVits</i> method: During the concatenation phase. The support representation tokens are denoted by $s_0$ and $s_1$ , the query representation token by $q_0$ and the query patch tokens by $q_p$ . The resulted feature vector is denoted by $X$ . . . . .	33

3.11	Mask settings (1-3). This figure displays how the masked attention map is multiplied by the vector $V$ to compute the new feature vector $Z$ . The zeroes in the masked attention map are the entries that have been masked using the corresponding setting. Note, that each row of the masked attention map sums to one. . . . .	34
3.12	Attention connectivity for support representation tokens (i.e. $s_0$ and $s_1$ ) using mask setting 1. Each support representation token attends all the tokens of the value vector $V$ except other support representation tokens. Precisely, the support representation tokens do not attend each other. . . . .	34
3.13	Cross attention to the query patch tokens example. The self-attention operation (denoted as SA) is applied to the feature vector $X$ resulting in a feature vector $Z$ where the query patch tokens $q_p$ are reverted to their initial state. . . . .	35
3.14	<i>ParallelVits+Encoder</i> method with auxiliary losses (setting 1). The auxiliary losses are computed over the similarity scores of support and query representation tokens before and between the encoder blocks (these are denoted by $B_0$ and $B_1$ respectively). . . . .	36
3.15	<i>ParallelVits+Encoder</i> method with auxiliary losses (setting 2). The auxiliary losses are computed over similarity scores of support and query representation tokens that have been encoded at a different level. . . . .	36
3.16	<i>BlendedVits</i> method: Incorporation of pooled patch tokens during the concatenation phase. The support representation tokens are denoted by $s_0$ and $s_1$ , the support pooled patch tokens by $s_{p0}$ and $s_{p1}$ , the query representation token by $q_0$ and the query patch tokens by $q_p$ . . . . .	37
3.17	<i>ParallelVits+Encoder</i> method: Incorporation of pooled patch tokens during the concatenation phase. The support representation tokens are denoted by $s_0$ and $s_1$ , the support pooled patch tokens by $s_{p0}$ and $s_{p1}$ and the query representation tokens by $q_0$ and $q_1$ . . . . .	37
3.18	The proposed workflow illustration. The meta-learning method is denoted by $M$ . The support and query sets that form the training and test tasks are denoted by $S$ and $Q$ respectively. . . . .	42
4.1	The mask setting 4. Areas painted in gray are masked. In particular, attention connectivity is restricted in a way that interaction involving support pooled patch tokens is only allowed between them and query patch tokens. . . . .	48



# Chapter 1

## Introduction

Deep learning (DL) is a thriving field of Machine Learning (ML) that has revolutionized Artificial Intelligence (AI) over the last decade. Having survived two AI winters and with its origins dating back at least to the invention of McCulloch-Pitts neuron (1943) [1], it seems that is finally here to stay. Recent advances, have enabled applications such as ChatGPT [87], an interactive conversational agent and DALLE-2 [88], a model capable of generating photo-realistic images from text, to catch even AI practitioners off guard, by demonstrating breathtaking capabilities.

The success story of DL can be merely attributed to a key idea, that is the hierarchical representation learning [20]. In particular, instead of constraining the model to discover only the mapping between a fixed representation of the input space and a target variable, in DL the model has to discover an input space representation as well. Extracting such high-level and abstract features from raw data is a challenging problem. However, providing that the task can be broken into a hierarchy of problems, DL learns high-level features by building on low-level ones in a hierarchical manner (illustrated in figure 1.1). Interestingly enough, automatically learnt representations are more effective than manually created ones but they come at a price; that is the necessity for large scale datasets, such as ImageNet [8] in computer vision.

In DL field, human brain has been a significant source of inspiration. On the one hand, it is proof by example that intelligent behavior is possible, thus imitating its mechanisms might be a good starting point [2], and on the other hand, it sets the scientific milestones to be pursued, since intelligent behaviour is the end goal. An intrinsic characteristic of human brain is to learn and generalize from few examples. Especially at an early age, humans can learn new tasks rapidly by utilizing little to no information. To illustrate that point, given a single image depicting an object, one is able to recognise objects of that particular category in other unseen images. Even more interestingly, a person can also use that single image to make an educated guess about images of unknown categories [14].

Although mainstream ML approaches have been quite successful in several large scale source tasks, they fail to demonstrate those abilities in the limited

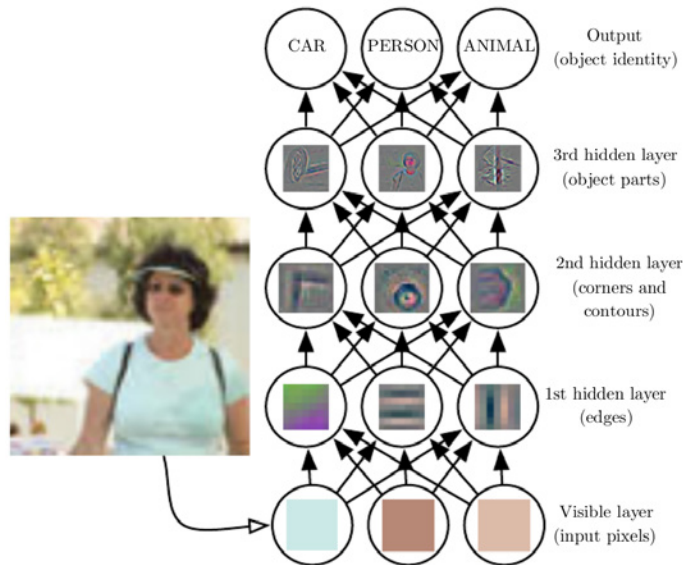


Figure 1.1: Representation learning. Taken from [20].

data regime. That partly, comes from the difficulty of learning a large number of parameters from few examples, that most likely leads to over-fitting [24]. The need to overcome those limitations provided fertile ground for the development of few-shot learning (FSL) which is the ML paradigm that attempts to imitate this brain inspired behaviour.

Sufficiently large annotated training data can be hard or impossible to be acquired due to inherent constraints of the task at hand. For example, in medical applications, annotation costs and privacy regulations do not allow for conventional ML to be applied effectively [67]. As a result, FSL serves as a suitable candidate for such applications. Specific examples of that kind, can be found in medical imaging such as chest x-ray pathology detection [81], in drug discovery applications, where making meaningful predictions is required [26], even in decoding and decoupling the signals of the human brain [71].

Furthermore, use cases of FSL also exist in time-constrained applications, where the extensive retraining of an ML model can be a prohibitive bottleneck. To name a few, the imitation of human actions in robotics [28], recommendation systems [33], visual tracking of arbitrary targets [53] and image retrieval [32] in computer vision, are some of the many applications that exploit FSL's advantages.

This line of work has been conducted in the context of the FSL image classification problem. In that problem, an  $N$ -way  $K$ -shot task is provided as input, comprised of two sets of images. These are the support and query sets that contain samples from the same  $N$  categories. The support set consists of  $K$  annotated samples per class, whereas the query set consists of  $Q$  non-annotated ones. The ultimate goal is to classify query samples into the correct categories.



To tackle the FSL classification problem, inspired by related works [85, 24, 31], variations of the metric-learning paradigm have been employed. This is an approach, where the support and query sets are projected into a space, in a way that samples are clustered with respect to their categories. Having projected both sets on the embedding space, a similarity measure is utilized to classify them.

As an embedding function, a vision transformer (ViT) [76] is utilized, which processes images as a sequence of image patches. Founded on the self-attention mechanism, ViT is a flexible architecture known for its state-of-the-art results. As an initialization of the model’s parameters, we leverage a self-supervised model that has been pre-trained using the DINO methodology [73]. We follow that approach for two main reasons. First, since ViT is a data-hungry architecture, employing a pre-trained model saves a lot of training time and requires less data. Secondly, it has been shown that especially self-supervised ViTs are superior feature map extractors [73].

In addition, we adopt a commonly used meta-learning framework, introduced by Vinyals et. al [24]. The key idea, is for training (aka meta-train) to mimic the test setting (aka meta-test); making the training objective consistent with the testing one. As a result, training is comprised of few-shot tasks as well. This approach has been found by Vinyals et. al [24] to minimise the meta-learning algorithm’s generalization error.

The focus of the conducted experiments, is to explore whether information exchange between the embedding functions is beneficial for the problem at hand. In that regard, we have proposed three main methods. These are namely *ParallelVits*, *ParallelVits+Encoder*, and *BlendedVits*. More precisely, *ParallelVits* restricts the information flow between the embedding functions, whereas the other proposed methods enable it through the attention mechanism in the ViT architecture. Additionally, we have conducted a grid search for several hyper-parameters related to the proposed methods, the meta-learning framework, and the employed neural network architecture.

The rest of the thesis is organized as follows. The second chapter 2, provides some essential information regarding the problem at hand and presents related works found in literature. In the third chapter 3, the employed methodology is presented, discussing our proposed methods and their variations. We continue with the fourth chapter 4, where we evaluate our methods and comment on the acquired results. Finally, we conclude with the fifth chapter 5, where we provide a summary of our conclusions and discuss probable future work directions.



## Chapter 2

# Related Work

In this chapter, we begin by the description of the image classification problem discussion on the different approaches that have been employed over time to provide an adequate solution. Then, we move to the presentation of some essential neural network architectures, namely Convolutional Neural Networks and Vision Transformers that have transformed the computer vision field. Additionally, we describe the few-shot learning discipline; providing the related definitions and its taxonomy. Moreover, we refer to the meta-learning perspective of few-shot learning paradigm and we emphasize on metric-learning approach. In the end, we present seminal related works following that approach and works in that field that employ the notorious vision transformer architecture.

### 2.1 Image classification

From the perspective of ML, classification is the problem where a computer program is asked to specify which of  $K$  categories some input belongs to [20]. More specifically, the algorithm is required to learn a mapping function  $f : \mathbb{R}^n \rightarrow \{1, \dots, k\}$  that given an input sample  $x$  should output the correct class  $y = f(x)$  or a probability distribution over the possible categories. Image classification, is a classification problem where the input is an image which typically contains only one object. There is a variety of datasets for image classification, some notable are ImageNet [8], CIFAR-10/100 [9] and MNIST [6].

Traditionally, image classification was resolved by the incorporation of computer vision feature detection algorithms and conventional machine learning models. However, these methods did not generalize well and as a consequence they could not be transferred to other domains [63]. In 2012, the introduction of AlexNet [11], a deep convolutional neural network (CNN) [6], has revolutionized computer vision discipline and popularized the use of deep learning [20]. CNNs were dominating the field for almost a decade until a more recent architecture known as vision transformers [76] has pushed the baseline even further for several computer vision tasks, including image classification.

Although VITs outperform CNNs in various occasions, that does not suggest that they should be regarded as a panacea. Specifically, there are some works [70, 89, 79] that make a comparison between those architectures in terms of generalization and robustness. These works agree on that CNNs have greater inductive bias that leads to better performance in a lower data regime. Nevertheless, VITs demonstrate better generalization on out-of-distribution samples.

## 2.2 Convolutional Neural Networks

### 2.2.1 Overview

Even though CNNs are not part of the proposed methodology, a concise description is provided here due to their major impact in the computer vision field and their wide use in related work. As CNNs are around for more than a decade, their structure has undergone a lot of tweaks and modifications to improve them in several aspects. Some notable works in that direction are AlexNet [11], VGG [16], NiN [13], Inception [18], ResNet [19], DenseNet [30], Wide Residual Nets [35] and so on.

### 2.2.2 Structure

Despite, the architectural upgrades the conventional architecture of CNNs comprise of the following components: (i) *convolutional layers*, (ii) *pooling layers*, and (iii) *fully connected layers*.

In *convolutional layers* (i), the input is convolved with a learnable kernel that has smaller spatial size and is able to compute local features in an efficient way. As we go deeper in the architecture the feature dimension of the layer increases. *Pooling layers* (ii) essentially reduce the spatial dimensions of the input signal making the representation approximately invariant to small translations of the input. Finally, *fully connected layers* (iii) are located at the end of the network, where the spatial dimensions of the input signal have been diminished and their main role is to convert the 2D feature maps into a 1D feature vector.

### 2.2.3 Convolution operation

The basic element of CNNs is the convolution operation, a specialized kind of linear operation, which in discrete time can be written as follows:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \quad (2.1)$$

In the equation 2.1,  $x$  is regarded as the input signal that is convolved with a learnable kernel  $w$ .

### 2.2.4 Notable Properties

According to [20, 44], the key properties which contribute to CNN architecture’s success, are namely: (i)*sparse interactions*, (ii)*parameter sharing* and (iii)*equivariant representations*.

*Sparse interactions* (i) essentially mean that the kernel is allowed to be smaller than the input. Hence, local features can be learnt and at the same time, as fewer operations are required, the algorithm is more computationally efficient. *Parameter sharing* (ii) implies that the same set of parameters (i.e. the same learnable kernel) is convolved with the input, as opposed to fully-connected layers where for each connection a different parameter is used. Having only one kernel for all positions of the input, substantially decreases the total parameters of the model and leads to better generalization. In addition, *equivariant representations* (iii) arise due to parameter sharing in the convolution mechanism. Equivariance in mathematical terms means  $f(g(x)) = g(f(x))$ . In other words,  $f$  is equivariant to  $g$  if the order of application does not change the result of the composite function. The convolution layer has equivariance to translation. That is to say, that if the input is translated, so does the output.

## 2.3 Vision Transformers

### 2.3.1 Overview

Transformer architectures originate from the natural language processing (NLP) field, where transformers constitute the de facto standard [68]. The first work to propose a transformer architecture founded on the self-attention mechanism for machine translation, a well-researched NLP task, is due to Vaswani et. al [34]. This work was followed by other significant advances, such as BERT [47] and GPT [54] that demonstrated how self-supervision and transfer learning can be leveraged effectively. After the revolution that NLP discipline has witnessed, computer vision researches have tried to incorporate transformer-like architectures to exploit the potential of self-attention in image processing. Both standalone and CNN-aided approaches have been attempted in that regard. However the first pure-transformer architecture for image recognition at scale was proposed by Dosovitskiy et. al [76], which is competitive to (or even better than) state-of-the-art CNNs. Since then, several variations of that model [92, 86] have been proposed to essentially tackle two main issues of the VIT architecture. These are the absence of strong inductive biases, such as translation equivariance and locality, as opposed to CNNs, and the necessity for more data that comes from it [92].

### 2.3.2 Structure

The vanilla VIT architecture [76] retains only the encoder part of the encoder-decoder structured NLP transformer [68]. This encoder can be broken down in encoder blocks that are connected in series. The encoder blocks encapsulate a

self-attention mechanism and a feed-forward network (FFN) which are connected residually and their inputs are normalized. To adapt NLP transformers to images, another step has to be taken. That is, the introduction of a projection layer that takes the input image, splits it into patches and eventually produces a sequence of patch embeddings. Moreover, a representation token is placed along the image embeddings that serves the purpose of image feature representation. Additionally, in order for the spatial structure of the image to be preserved, a position embedding is added to each of the tokens. Finally, a multi-layer perceptron (MLP) is being placed at the top in order to classify the image with respect to the resulted image representation. For illustration purposes, the general layout of the ViT architecture and the internals of the transformer encoder's block are displayed in figure 2.1.

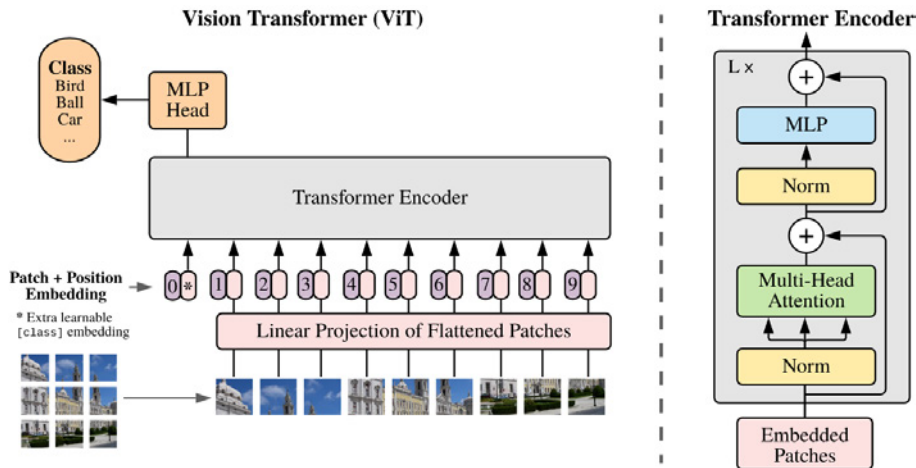


Figure 2.1: ViT (left side) and Encoder block (right side). Taken from [76].

### 2.3.3 Self-Attention

The fundamental element of transformers is termed *self-attention*. The key idea to self-attention is to obtain a weighting scheme for the input by multiplying the input by itself. In transformers, self-attention is parameterized with learnable weights. These are namely, key weights  $W^K \in \mathbb{R}^{d \times d_k}$ , query weights  $W^Q \in \mathbb{R}^{d \times d_k}$  and value weights  $W^V \in \mathbb{R}^{d \times d_v}$ . During a forward pass, given an input sequence  $X = (x_1, x_2, \dots, x_n) \in \mathbb{R}^{n \times d}$ , where  $d$  is the embedding dimension, three matrices are calculated by using the respective weights. These are  $K = XW^K$ ,  $Q = XW^Q$  and  $V = XW^V$ . At the end, the result of the self-attention layer is computed as follows<sup>1</sup>:

<sup>1</sup>The scaling factor  $\sqrt{d_k}$  is being used for keeping the product  $Q \times K^T$  sufficiently small, to avoid that resulting in extremely small gradients [34].

$$Z = \underbrace{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)}_{\text{attention map}} \times V \quad (2.2)$$

As it is apparent from the equation 2.2, for a given entry in the sequence, self-attention computes the dot-product of the query with all the keys, which is then normalized, using the softmax function, to obtain the attention scores. These attention scores are then participate as weights in a weighted sum of the values to eventually yield the self-attention result for that particular sequence entry. An illustration of the process can be seen in figure 2.2.

In practice, a variant of the aforementioned mechanism is utilized which is called *multi-head attention* (displayed in figure 2.2). As the name suggests, it breaks the feature dimension down to  $d_v = d_k = d/h$ , where  $h$  stands for the number of attention heads. For each head the self-attention is computed as shown in equation 2.2 and then the heads are concatenated and projected using a weight matrix  $W^O \in \mathbb{R}^{h*d_k*d}$  as follows:

$$Z = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.3)$$

In addition, to restrict attention to some of the entries of the input sequence an *attention mask* is being used. The attention mask is a matrix  $M \in \mathcal{R}^{d_k \times d_k}$  full of zeroes that in order to exclude a particular entry of the attention map a  $-\infty$  is being placed at the corresponding position in the attention mask. Afterwards, this mask is added as shown in equation 2.4 before softmax is applied.

$$Z = \underbrace{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)}_{\text{attention map}} \times V \quad (2.4)$$

### 2.3.4 Notable Properties

It has been shown empirically [57] that multi-head self-attention is a more generic operation than convolution, if sufficient parameters are provided. Furthermore, both local and global features can be computed [52], without having the strong image-specific inductive bias of CNNs [76], which also leads to better generalization. Moreover, filters are computed dynamically from the learnable parameters as opposed to convolutions, making transformers more robust towards adversarial attacks [86]. Last but not least, transformers have shown excellent scalability [76]. That can partly be attributed to multi-head attention implementation which is optimized for parallelization [86].

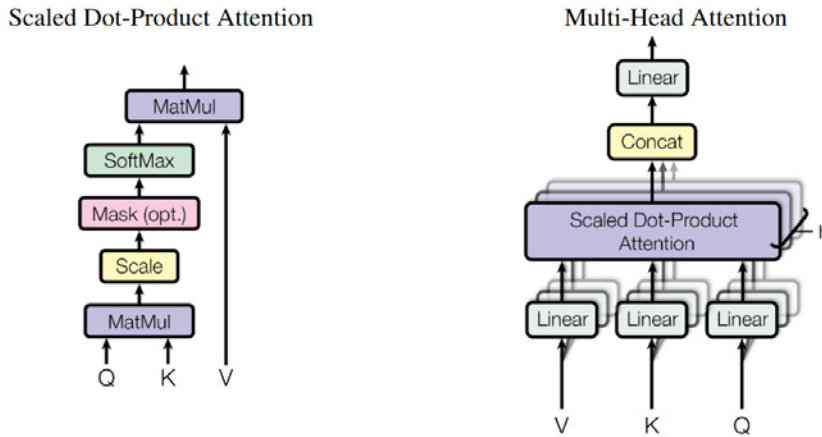


Figure 2.2: Scaled dot-product attention (left side) and multi-head attention (right side). Taken from [34].

## 2.4 Foundation models and DINO

According to Bommasani et. al [82], a foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adopted to a wide range of downstream tasks. Foundation models have two main characteristics, namely *emergence* and *homogenization*. *Emergence* means that the behaviour of the model is induced implicitly rather than explicitly introduced. For example, GPT-3 [54] with 175 billion parameters have demonstrated capabilities, such as in-context learning, without being specifically trained for it. In addition, *homogenization* can be understood as the incorporation of the same set of methodologies to tackle multiple problems. As an example, almost the same transformer architecture is leveraged both in computer vision and NLP disciplines. Those characteristics have made foundation models very attractive for industry, as fine-tuning a model requires fewer resources than training it from scratch especially for data hungry models as transformers [76]. Additionally, self-supervised pre-training, which most of these foundation models have gone through, has been shown to result in better feature representations than those acquired by supervised pre-training [72, 61, 62].

Every self-supervision methodology is based on a pretext task so that target labels (aka pseudo labels) can be extracted without requiring human annotation. In computer vision, for example, that could be the prediction of geometrical transformation, such as rotation [37], the prediction of the color in an image [25] or the prediction of a whole patch [22]. Nevertheless, the method introduced in the next paragraph does not require any pseudo labels at all.

In this work, a more recent self-supervision methodology is being adopted, termed as DINO [73]. DINO stands for self-distillation with **no** labels. As a



knowledge distillation method, it has a student and a teacher network with identical architectures (aka co-distillation). However, neither the teacher nor the student model have undergone any pre-training prior to the method. A forward pass of a single training step proceeds as follows. The models are provided with a different augmentation of an image (e.g. a random crop), then the teacher’s output (i.e. a  $k$ -dimensional feature vector) is centered using a mean computed over the batch. Afterwards, both the outputs of the models are passed through a temperature softmax (i.e. a softmax with a scaling factor) and finally the similarity of those predictions is measured using the standard cross-entropy loss. The backward pass includes only the student network, since the teacher network’s parameters are updated only through an exponential moving average (EMA) of the student network’s parameters. An illustration of the process can be seen in figure 2.3.

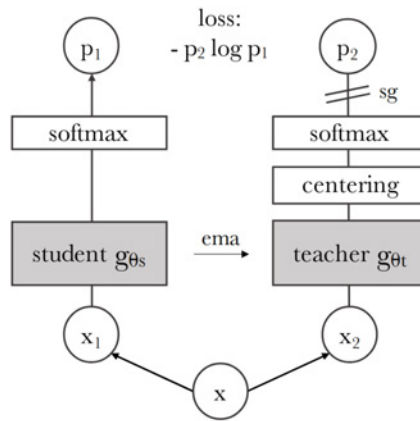


Figure 2.3: Self-distillation with no labels. Taken from [73].

Despite this self-supervision methodology is architecture agnostic and can be applied with a CNN as well, the authors make the following observations. Firstly, the feature maps of the last layer of the self-supervised ViT contain explicit information about the semantic segmentation of an image, as it is visible in figure 2.4. Secondly, these features, without any further processing, are excellent KNN classifiers. Interestingly enough, both of these statements do not hold for either supervised ViTs or CNNs. These observations are indicative of why self-supervised ViTs are better feature extractors.

## 2.5 Few-shot learning

### 2.5.1 Definition of FSL

Inspired from the, not necessarily human, brain ability to learn through limited examples, FSL is a well-studied field of ML and has received a lot of attention from ML researchers. In general, it is comprised of algorithms that leverage prior knowledge or introduce some bias to account for the weak supervision signal. To

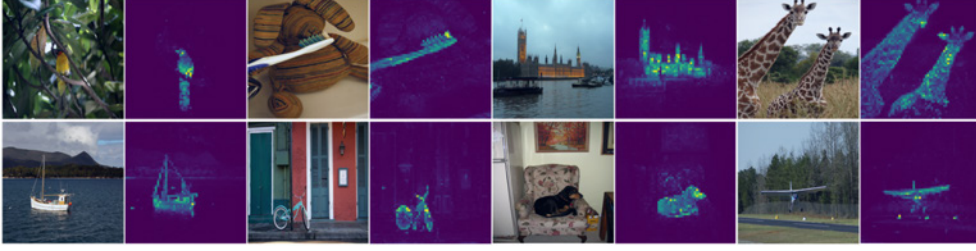


Figure 2.4: Self-attention from a vision transformer trained with no supervision. Taken from [73].

formally state the FSL problem definition the recall of the widely accepted ML definition [5], is deemed necessary.

**Definition 1** *A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .*

To provide an example, in the chess learning problem, the task is the well-known board game, the experience corresponds to the number of games the system has seen, during training, and the performance measure might be the win-ratio of the system against opponents. Based on the aforementioned definition a highly-cited research survey [66] suggests the following one:

**Definition 2** *Few-shot Learning (FSL) is a type of machine learning problems (specified by  $E$ ,  $T$ , and  $P$ ), where  $E$  contains only a limited number of examples with supervised information for the target  $T$ .*

### 2.5.2 Core issue in FSL

Unsurprisingly, the limited training examples, that serve as the system’s experience, render the empirical risk minimizer unreliable, transforming FSL into a more demanding problem than the original one [94, 66]. To illustrate that point, let us recall that a loss function  $\mathcal{L}$ , in supervised learning, indicates how close a prediction, acquired using a hypothesis  $h \in \mathcal{H}$ , is to ground truth (i.e. the correct prediction) of a specified task. In particular, one can go further and calculate both the expected and empirical risks.

$$\mathcal{R}(h) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(h(x_i), y_i) = \mathbb{E}[\mathcal{L}(h(x), y)] \quad (2.5)$$

There are three hypothesis related to this reasoning. Firstly, the optimal hypothesis  $\hat{h}$  that minimizes the expected risk, which is not constrained to the hypothesis space  $\mathcal{H}$ . Secondly, the best approximation of the  $\hat{h}$  in  $\mathcal{H}$ , denoted by  $h^*$ , and finally, the best hypothesis in  $\mathcal{H}$  obtained by empirical risk minimization,

denoted by  $h_N$  ( $N$  corresponds to the number of training examples). Now, the decomposition of the excess error can be written as:

$$\mathbb{E}[\mathcal{R}(h_N) - \mathcal{R}(\hat{h})] = \underbrace{\mathbb{E}[\mathcal{R}(h^*) - \mathcal{R}(\hat{h})]}_{\text{1st term}} + \underbrace{\mathbb{E}[\mathcal{R}(h_N) - \mathcal{R}(h^*)]}_{\text{2nd term}} \quad (2.6)$$

Although the above reasoning is simplified by assuming that the aforementioned hypothesis functions are unique and by constraining the scope of the problems to supervised only, the following conclusion can be drawn. The excess error, in general, depends on the hypothesis space  $\mathcal{H}$  (1st term), which is determined by the model at hand and on the number of samples  $N$  (2nd term), in the training set. Thus, having less samples for the empirical risk minimization makes the FSL problem more demanding.

### 2.5.3 Taxonomy

FSL is a well-studied field that encompasses a variety of methods. Recent surveys in the field, such as [66, 84, 94], have not reached to a consensus on FSL taxonomy. As a result, the most accepted one [66], on the basis of citations count at the time of this writing, will be adopted. The rationale behind this particular taxonomy is that all FSL methods use prior knowledge to compensate for the limited data regime which they are constraint to operate in. Hence, a taxonomy should be founded on the grounds this prior knowledge is incorporated. In that regard, authors divide these approaches into three perspectives. These are namely *data*, *model* and *algorithm*. Although, an exhaustive, in-depth, exploration of those perspectives would be very interesting, for the purpose of this thesis, only a concise description of the main ideas is provided here.

#### 2.5.3.1 Data

The *data* perspective is comprised of data augmentation methods that increase the number of samples in the training dataset. Apart from that, the augmented samples make the algorithm invariant to a variety of transformations, such as translation [36, 23], flipping [40], rotation [40, 23], scaling [45, 15] and so on. The augmentation policy is selected carefully to exploit inherent properties of the task at hand [66]. As a consequence, these methods are not easily transferable to other tasks without manually selecting a different set of augmentations. To alleviate that, there is a line of work [46, 60] that proposes algorithms which can acquire augmenting policies in an automatic way.

#### 2.5.3.2 Model

Methods belonging to the *model* perspective leverage prior knowledge to constraint the hypothesis space  $\mathcal{H}$ , so that the search for the best hypothesis  $h$  would be feasible with limited samples. Under this category fall multi-task learning [91, 4],

embedding learning [43, 24, 31], learning with external memory [21, 12, 17] and generative modeling [48, 41].

### 2.5.3.3 Algorithm

The *algorithm* perspective includes methods that utilize prior knowledge to search for a parametrization  $\theta$  of the best hypothesis  $h^* \in \mathcal{H}$ . Assuming that a conventional optimization method, such as stochastic gradient descent is difficult to converge when insufficient samples are supplied, these methods leverage prior knowledge to provide a better parameter initialization [27, 38] or to train a meta-learner that outputs the parameter updates [29].

## 2.6 Meta-learning

Almost all FSL methods pose their problem as a meta-learning problem to incorporate prior knowledge. The key idea in meta-learning [7], although general and abstract, is to *learn how to learn*. This implies that there are two learners, an inner one (aka learner) and an outer one (aka meta-learner). The former solves a specific task (aka base task), such as image classification, whereas the latter interferes with the former so that an outer objective will be met. That might be generalization improvement or learning speed of the inner learner [84]. More formally, meta-learning improves the progress  $P$  on a new task  $T$  by leveraging the experience  $E$  obtained by the task and meta-knowledge extracted from other related tasks by a meta-learner. The parameters of the meta-learner are optimized in order to minimize the error across all learners. This broad description of meta-learning encompasses many conventional outer algorithms such as grid hyper-parameter search, cross-validation, early stopping, and so on.

### 2.6.1 FSL image classification framework

A very common meta-learning framework for FSL image classification, first introduced by Vinyals et. al [24], is to make the training and testing objectives consistent. In particular, the dataset is split into two disjoint sets of categories. These are, the base categories (aka train categories) and the novel categories (aka test categories). During the *training phase* (aka meta-train), tasks (aka episodes)  $T$  are formed from base categories only. Specifically, each task  $T$  is composed by a support set  $S$  (i.e. the set of labeled samples) and a query set  $Q$  (i.e. the set of unlabeled samples). Those sets contain samples of the same categories, and in each set the same number of samples per category is used. The meta-learner is then trained to minimize the error predicting the labels of the query samples conditioned on the support set  $S^2$ . Since the training phase mimics the testing phase, the only difference during meta-test is that now tasks  $T'$  are formed from

<sup>2</sup>Only the labels of the support samples are available to meta-learner, as this would also be the case in the test setting.

novel categories only. The described procedure is a form of meta-learning since the training procedure explicitly *learns to learn* from a given support set  $S$  to minimise a loss over a query set  $Q$ . Reasonably, as  $T'$  diverges from  $T$ , that is when the domain shift increases between train and test categories, it is expected for the performance to decrease [24].

The convention is that a FSL task is described by the number of sampled classes (i.e.  $N$ -way) and the samples per class in the support set (i.e.  $K$ -way). Usual settings for the aforementioned framework are the 5-way 1-shot tasks and 5-way 5-shot tasks. These settings serve as benchmarks, when they are combined with a specific dataset, for the comparison of several FSL methods. There is a variety of FSL image classification datasets, some notable are mini-imageNet [24, 90], CifarFS [39], tiered-imageNet [42], meta-dataset [65] and CUB [10].

## 2.7 Embedding learning for Image classification

### 2.7.1 Overview

*Embedding learning* (aka metric-learning) is a very common approach towards FSL image classification; categorised under the *Model* perspective of FSL (explained in 2.5.3.2). Furthermore, it usually follows the meta-learning framework proposed by Vinyals et. al [24] (described in 2.6.1). In this section, a description of embedding learning will be provided along with its variations, namely task-invariant, task-specific and hybrid. Furthermore, since under this paradigm falls the proposed methodology some closely related works will be presented.

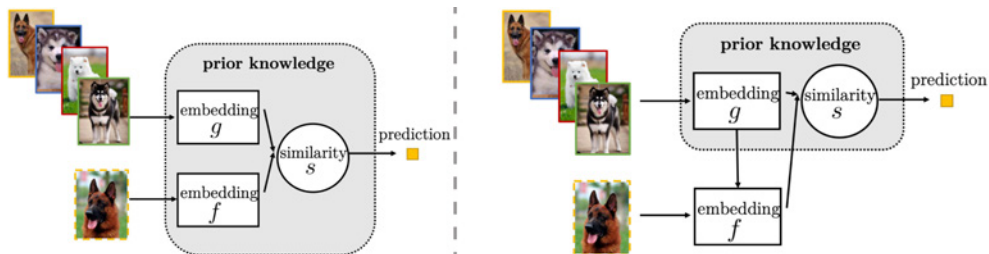


Figure 2.5: Task-invariant (left side) and hybrid (right side) embedding models. Taken from [66].

### 2.7.2 Description

In order to reduce the size of the hypothesis space  $\mathcal{H}$ , embedding learning utilizes a function (aka embedding function) to map each sample  $x \in \mathbb{R}^d$  to a space  $\mathcal{Z}$  of a lower dimension (aka embedding space). In that space, samples of the same category are clustered together with respect to a similarity measure  $s(\cdot, \cdot)$ . Due to the use of that similarity measure, these methods are also referred to as metric-learning based methods.

More precisely, embedding learning involves two embedding functions  $g : \mathbb{R}^d \rightarrow \mathcal{Z}$  and  $f : \mathbb{R}^d \rightarrow \mathcal{Z}$ , where  $g$  is used to embed the support samples  $x_s$  and  $f$  is used to embed the query samples  $x_q$ . The classification of the query samples takes place by comparing the similarities between embedded support samples  $g(x_s)$  and embedded query samples  $f(x_q)$ . The query sample will be eventually classified to the category of the support sample that is located closer in the embedding space.

Depending on the implementation, the similarity measure  $s(\cdot, \cdot)$  can be learnable or fixed. Moreover, the embedding functions can be either dependent or independent. In addition, embedding learning is further classified based on the knowledge that is being leveraged in order for the embedding functions to be learnt. Specifically, there is *task-specific*, *task-invariant* and *hybrid* embedding learning. Task-specific embedding models learn an embedding function utilizing information only from the task at hand. On the other hand, task-invariant embedding models learn an embedding function from a sufficiently large dataset, independent to the task. Finally, hybrid embedding models are a combination of the aforementioned paradigms, where a general embedding function is refined by incorporating task specific information. For illustration purposes, figure 2.5 displays the task-invariant and hybrid embedding models.

### 2.7.3 Related Work

Some of the most influential and seminal works towards embedding learning image classification, arranged in chronological order, are chosen to be described here. These are namely, Siamese Nets [14], Matching Nets [24], Prototypical Nets [31] and Relation Net [43].



Figure 2.6: Siamese Nets: Training and testing procedure. Taken from [14].

### 2.7.3.1 Siamese Nets

Siamese Nets were first introduced in the early 1990s [3] as a solution to the verification problem. The key characteristics of the siamese networks is that they have identical architecture and their parameters are shared (aka tied parameters). In addition, they accept two distinct inputs and output two distinct high-level feature representations, along which a metric is being computed. The idea is that two similar images would result to a high score of similarity, since the parameters of the models are tied and the representations of the samples in the feature space are expected to be close. Koch et. al [14] demonstrated that after training Siamese Nets on the verification problem; acquiring good feature representations, these features can generalize to the one-shot classification task. In particular, to adapt the model to the one-shot problem’s formulation, given a query sample, they create all the pairs between the query sample and the available support samples. Afterwards, they feed those pairs to the Siamese Nets and they classify the query sample to the class of the support sample participating in the most similar pair. An illustration of the process can be seen in figure 2.6.

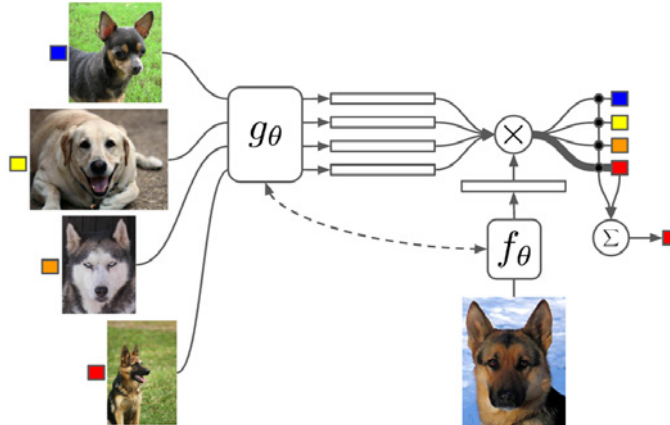


Figure 2.7: Matching Nets. Taken from [24].

### 2.7.3.2 Matching Nets

Matching Nets [24] is an impactful work in the FSL image classification field as it introduces one of the most commonly used meta-learning frameworks (described in 2.6.1). Apart from that, its novelty resides also in the modeling level. In particular, for a given query sample  $\hat{x}$  and a support set  $S = \{(x_i, y_i)\}_{i=1}^k$ , they embed both the query sample and the support samples in  $S$  using the embedding functions  $f$  and  $g$  respectively. Afterwards, they calculate the cosine similarity<sup>3</sup> of the query with all the support samples in  $S$  which they later normalize using softmax over all similarities.

<sup>3</sup>in equation 2.7 cosine similarity is denoted as  $c(\cdot)$ .

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^k e^{c(f(\hat{x}), g(x_j))}} \quad (2.7)$$

The resulting probability distribution over all possible categories of query sample label  $\hat{y}$  is obtained as follows <sup>4</sup>:

$$P(\hat{y}|\hat{x}, S) = \sum_{i=1}^k a(\hat{x}, x_i) y_i \quad (2.8)$$

Hence, in the one-shot case that would be:

$$P(\hat{y}|\hat{x}, S) = \begin{bmatrix} a(\hat{x}, x_0) \\ \vdots \\ a(\hat{x}, x_k) \end{bmatrix} \quad (2.9)$$

Whereas, when more than one sample per class is utilized, the probability for each category will be a sum over the attention kernels  $a(\hat{x}, x_i)$  formed from samples of that particular category<sup>5</sup>.

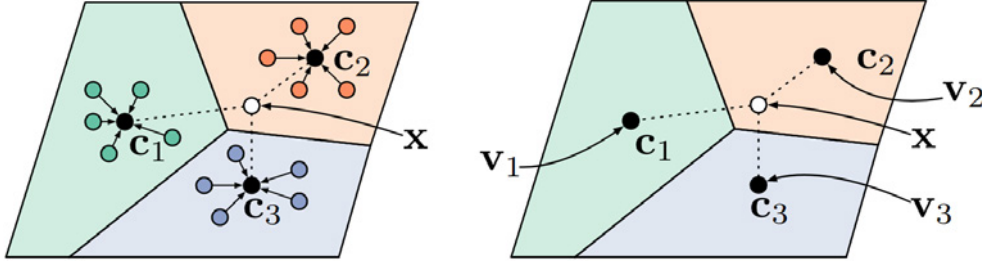


Figure 2.8: ProtoNets: few-shot (left side) and zero-shot (right side) settings. Taken from [31].

### 2.7.3.3 Prototypical Nets

Prototypical Nets [31] take the idea of Matching Nets [24], for few-shot learning tasks, one step further. That is done by introducing a simple inductive bias which presumes that a single prototype exists, for each category, where embedded samples, belonging to that category, cluster around it. These prototypes are called centroids; an example of that can be seen in figure 2.8. To capitalize on that, the proposed classification method first embeds the support samples using an embedding function (i.e. a neural network) and then computes the centroids for each category. Afterwards, the classification proceeds by assigning the query sample to the category of the nearest centroid.

<sup>4</sup>note that  $y_i$  is a one-hot vector.

<sup>5</sup>that probability will be well defined since the softmax is computed over all similarities.



In more detail, centroids are calculated by taking the mean over the support samples embeddings of the same class. After that, classification takes place by computing the normalized similarity between query samples and centroids quite similarly to the Matching Nets methodology. Consequently, for the one-shot case where only one support sample is given for each category ProtoNets are equivalent to MatchingNets.

An observation made by the authors regarding the similarity metric is that euclidean distance is better suited for their problem formulation rather than cosine distance. They presume that this can be attributed to the fact that euclidean distance belongs to the class of Bregman divergences where cosine distance does not. Another outcome of their work, related to the episode composition, was the observation that when they formed training episodes by sampling more classes compared to the test setting led to better results.

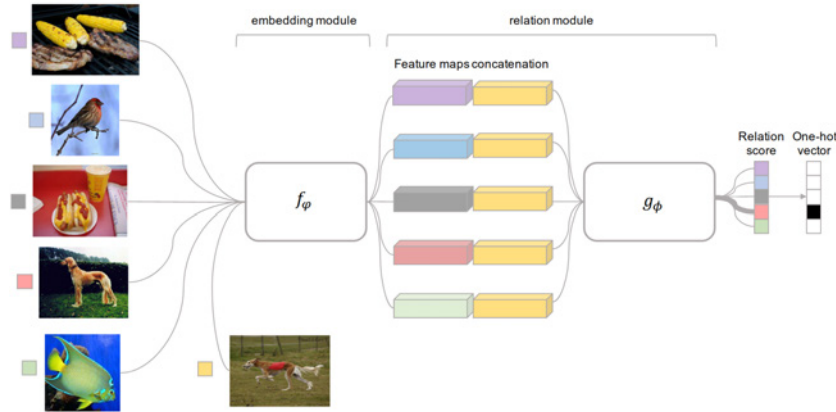


Figure 2.9: Relation Net. Taken from [43].

#### 2.7.3.4 Relation Net

Relation Net [43] proposes an extension to the aforementioned works [24, 31] by incorporating a learnable non-linear comparator as a replacement for the fixed metric employed by previous works. As it is illustrated in figure 2.9, the proposed method, provided a query sample and a support set, embeds both of them using an embedding function, and then concatenates the embeddings in a pairwise manner, so that each pair is formed by the query embedding and one of the support embeddings. In addition, it employs a relation module (i.e. learnable non-linear comparator), that is implemented as a deep neural network, and undertakes the role of assigning each pair a similarity score. Then, the query sample is classified to the category of the support sample in the pair that scored the highest similarity value.

Authors base their approach on the conjecture that having both embedding function and metric be learnable, provides greater flexibility to their model as

opposed to previous models where the metric is fixed. They also claim that a fixed metric is constrained on comparing embeddings in an element-wise manner and usually relies on the assumption of linear separability of the provided features. As a result, it is more difficult for those methods to generalize in inadequately discriminative representations.

### 2.7.4 Related Work based on ViTs

Vision transformers have seen limited use in FSL with respect to embedding learning for image classification, as also pointed out by [85]. Nevertheless, there are some works that either utilize them as robust feature extractors [85, 83] or as a manner to incorporate task-specific information by essentially exploiting the attention mechanism [59]. These works are going to be described here.

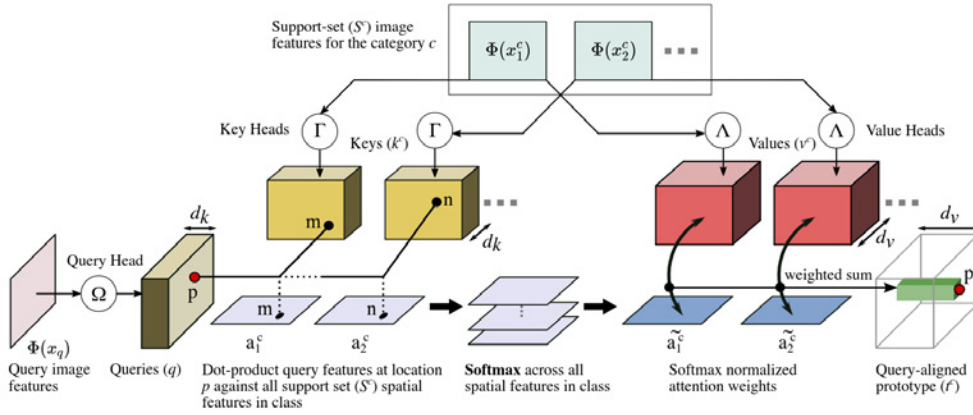


Figure 2.10: Cross Transformers. Taken from [59].

#### 2.7.4.1 Cross Transformers

Cross Transformers [59] is a closely related work that its novelty is twofold. To begin with, the authors formulate contrastive learning; specifically the SimCLR framework [55], in a way that is compatible with the meta-learning framework proposed by Vinyals et. al [24] (described in 2.6.1). That is done to avoid a problem they call *supervision collapse*. Moreover, they propose a novel transformer architecture, termed as CrossTransformer, which they use on top of a CNN, capable of harnessing the limited labeled samples during the test setting to accurately classify the query samples.

The term *supervision collapse* is used to describe the phenomenon where a classification model, during training, learn class specific patterns that are effective in order to cluster samples of the same class, minimizing other ways that samples might relate to each other. However, the latter may be useful in out-of-distribution tasks (i.e. different classes or different domain); thus *supervision collapse* impairs the model's ability to generalize. That problem is inherent to previous works, such

as ProtoNets [31], that are trained using a classification objective. To circumvent this limitation, they employ contrastive learning which, as they show, result in superior feature representations immune by design to the aforementioned problem.

As per their proposed pipeline, given a few-shot task and a query sample  $x_q$ , they provide as input the  $x_q$  along with all the support samples of a class  $c$ . The goal is to create query-aligned prototypes **for each class** by aligning the query feature representation  $\Phi(x_q)$  with the feature representations of the support samples  $\Phi(x_i^c)$ . At first, all the samples are embedded using a CNN. The employed CNN has its last pool layer dropped to preserve the spatial dimensions of feature representations. Afterwards, the cross transformer architecture is applied. The diagram in figure 2.10 illustrates how the cross-attention, from the query features  $\Phi(x_q)$  to the support features  $\Phi(x_i^c)$ , takes place. Specifically, it illustrates the process for a **particular spatial position**  $p$  of  $\Phi(x_q)$ . The depicted  $\Omega$ ,  $\Gamma$  and  $\Lambda$  are the attention weights for the calculation of the attention queries, keys and values, respectively. After computing the query-aligned prototypes  $t^c$  for each class the classification of the query sample  $x_q$  happens by aggregating euclidean distances between aligned local features from the above prototypes and corresponding query image values  $w_p = \Lambda\Phi(x_q)_p$ . Consequently, the proposed pipeline is able to make local part-based comparisons and to account for spatial alignment in a manner which is more agnostic to the underlying classes.

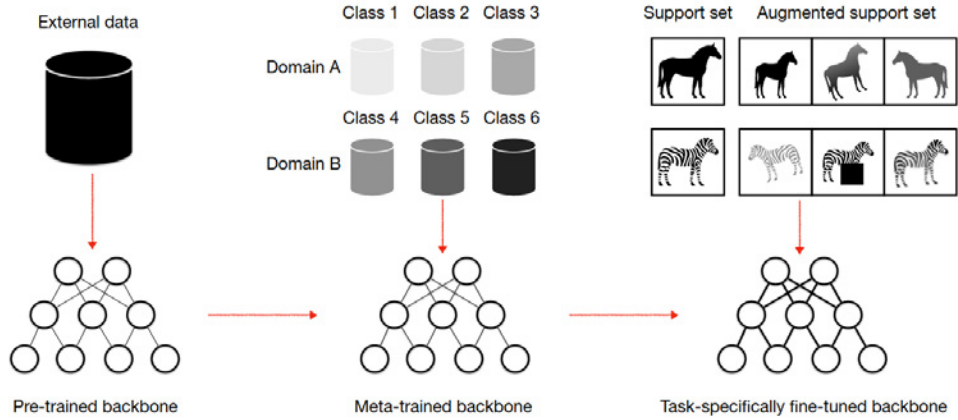


Figure 2.11: PMF. Taken from [85].

#### 2.7.4.2 PMF

PMF [85], which stands for **P**re-train, **M**eta-Learn, **F**ine-Tune, is a recent work which proposes a three stage simple pipeline to push the state-of-the-art even further. The novelty of PMF lies in the combination of its components. To begin with, PMF employs a ViT architecture [76] which is pre-trained in a self-supervised manner using DINO [73]. As for the meta-learner, PMF utilizes the Prototypical Nets [31] work. Consequently, the embedding function that is used in Prototypical

Nets is substituted with the aforementioned pre-trained ViT. Finally, the third stage of the pipeline is fine-tuning which will be described in more detail.

Fine-tuning procedure takes place right before meta-test time for a number of optimization steps. In each step, support samples from novel categories are sampled and augmented using several transformations (e.g. rotation, Gaussian blur, color jittering, etc.). Then, the fine-tune task is for the model to classify the augmented samples (i.e. fake query samples). The application of meta-test time fine-tuning is beneficial when the domain shift between meta-train and meta-test is large. To illustrate that point, the authors perform experiments where the source dataset (i.e. meta-train dataset) differs from the target dataset (i.e. meta-test dataset).

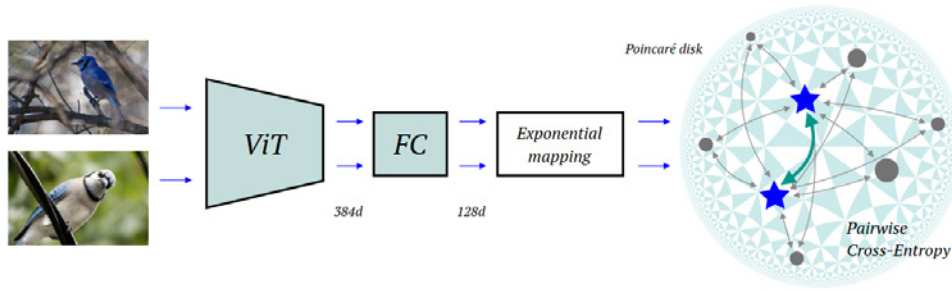


Figure 2.12: Hyperbolic Vision Transformers. Taken from [83].

### 2.7.4.3 Hyperbolic Vision Transformers

Hyperbolic Vision Transformers [83] emphasize on how meta-learners can learn highly-discriminative models so that embeddings of the same category will be clustered together, whereas those of different category will be pushed apart. To encourage the meta-learner to result in such models the authors propose to embed samples in the hyperbolic space and especially to the Poincare disc [93]. Apart from that, similarly to aforementioned works, they employ a vision transformer [76] as the model’s underlying architecture which is pre-trained with DINO [73]. In addition, they propose a form of contrastive learning, during the meta-train phase, using a pairwise cross-entropy loss based on hyperbolic distances.

The intuition behind hyperbolic embeddings is that hyperbolic spaces are intrinsically suited for hierarchical data. Moreover, hyperbolic spaces without sacrificing the model’s accuracy or representational power, have the ability to use low dimensional manifolds for embeddings. One reason is that in hyperbolic spaces the volume of an object scales exponentially with respect to each diameter, whereas in euclidean space scales polynomially.

The proposed pipeline is illustrated in figure 2.12. In particular, given two samples of the same category (aka positives) they use a pre-trained ViT to extract the feature representations of the samples which they later project using a fully

connected layer. Afterwards, they map those projected features onto the Poincare disc. The blue stars, depicted in the figure, denote the mapped representations of the samples that are clustered together, whereas the gray circles represent other samples from the same batch that belong to different categories (aka negatives). Finally, the displayed arrows indicate the pair-wise cross entropy loss.



## Chapter 3

# Methodology

### 3.1 Overview

In this line of work, we tackle the FSL image classification problem. That is accomplished by following the embedding learning (aka metric-learning) paradigm under which an embedding function projects the samples in a well partitioned embedding space; enabling classification with a fixed similarity measure to be possible. To train our deep embedding functions we employ the meta-learning framework proposed by Vinyals et. al [24], where the training setting mimics the testing one. The main focus of this work is to understand whether and how the information flow between the embedding functions is beneficial in solving the problem at hand. In that regard, we have proposed three main architectures and variations. In addition, we have explored other factors to improve the overall performance of our methods.

This chapter is organised in the following way. We begin by explaining the adopted meta-learning framework, then we elaborate on the case study of this work and the underlying intuition. Afterwards we proceed by describing the employed embedding function and its properties and we continue by presenting our main architectures, their variations and other explored factors. Finally, we conclude this chapter by demonstrating the overall workflow.

### 3.2 Meta-learning framework

The meta-learning framework proposed in the seminal work of Matching Networks [24] has been the most common framework for few-shot learning in recent years. As explained in 2.6.1, it is based on a simple machine learning principle: test and train conditions must match. Specifically, the dataset is split into train and test categories that are used in meta-train (i.e. the training phase) and meta-test (i.e. the testing phase) respectively. Each phase is comprised of tasks (aka episodes). A task consists of a support set (i.e. the set of annotated samples) and a query set (i.e. the set of non-annotated samples). Each of those sets contain samples

of the same  $N$  randomly sampled categories. Within a set, the same number of samples per category are randomly selected. Meanwhile, the number of samples per category can differ between the support and query sets <sup>1</sup>. Given an  $N$ -way  $K$ -shot task, which stands for  $N$  sampled classes and  $K$  samples per class in the support set, a meta-learner is tasked to specify the classes of the samples belonging to the query set with respect to the samples in the support set. To further clarify how tasks are formed we provide an illustration in figure 3.1.

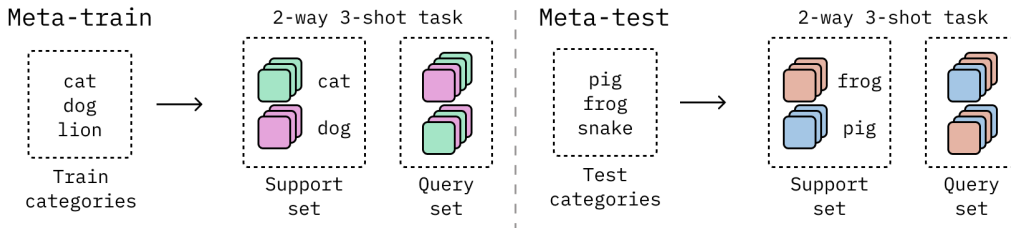


Figure 3.1: An example of 2-way 3-shot tasks during training and testing phases. For each task two classes are sampled for both sets and three samples per class for the support sets.

### 3.3 Case Study

Due to the limited data regime at the testing phase, where only few examples are provided to classify samples of unseen categories during the training phase, the employed meta-learning algorithm should be capable of harnessing every bit of available information. As Doersch et. al proposed with the Cross-Transformers architecture [59], we hypothesise that information flow between the embedding functions could transform the resulting embeddings appropriately, leading to a better partitioning of the embedding space, where the classification process with a fixed metric will be easier to tackle. In particular, the ideal partitioning of the embedding space occurs when embeddings of samples belonging to the same category are clustered together, whereas embeddings of samples of different categories are pushed apart.

In that regard, we explore the full spectrum from task-invariant to hybrid embedding learning (aka metric-learning). As described in 2.7.2, in task-invariant embedding learning the embeddings of support and query samples are generated independently and the information that the algorithm exploits has been encoded in the embedding function’s parameters during the training phase. Conversely, in hybrid embedding learning both task-invariant and task-specific information is leveraged. More specifically, the generation process of support and query embeddings not only leverages the encoded information in the embedding functions’

<sup>1</sup>For example, in figure 3.1 there are four samples per category in the query set and three samples per category in the support set.



parameters, but also information that is exchanged between them. An illustration of the task-invariant and hybrid embedding learning can be seen in figure 3.2.

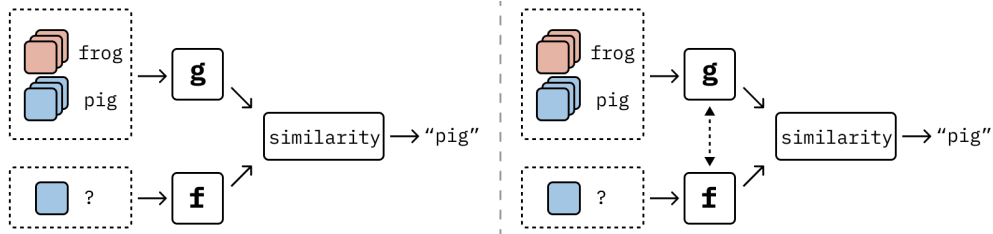


Figure 3.2: Task-invariant (left) vs Hybrid embedding learning (right). The support and query sets are embedded with the embedding functions  $g$  and  $f$  respectively. Then, the query sample is classified to the category of the support sample which is closer in the embedding space. At the right side there is a bidirectional arrow that indicates the information flow between  $g$  and  $f$  in the hybrid paradigm.

Nevertheless, our exploration does not end there. Information flow can be established in several ways and to multiple directions. For example, information can flow from support to query embedding function, meaning that the projection of query embeddings will be constraint on the provided information of the support set. Moreover, information flow can be bidirectional. That implies that both query and support embeddings exploit information from each other to be projected in the embedding space. To that end we have designed three main methods and variations of them in an attempt to cover as many scenarios as possible.

To conclude this section, our case study can be summarised in the following two questions: (i) Does information flow between the embedding functions is beneficial for the problem at hand? (ii) In what way this information flow can be established?

### 3.4 Embedding Function

In metric-learning, an embedding function is required to project the samples in the embedding space. Here, the role of the embedding function is undertaken by a vision transformer architecture [76] (illustrated in 2.3). More specifically, we employ a self-supervised pre-trained ViT that has been trained with the DINO [73] methodology (presented in 2.4), following the works of [85, 83]. The choice of this particular architecture is mainly based on its desirable properties and state-of-the-art results.

First of all, contrary to CNN’s hierarchical design pattern, ViT has an isotropic architecture. Isotropic neural networks have the property that all of the weights and intermediate features have identical dimensionality. That comes in handy when there is a need for cross-layer parameter sharing. Another useful property, that is inherited by the attention mechanism, is the shape independence. In other words, at any point in the network one can introduce more tokens without the need to re-train or adjust any other hyper-parameters of the architecture. Hence,

concatenating tokens at any point in the network becomes trivial. Furthermore, as discussed in 2.3 and 2.4, ViTs and specifically self-supervised ones are excellent feature extractors without having the strong image-specific inductive bias of CNNs, which leads to better generalization.

It is worth mentioning, that the resulted embedding for a particular sample corresponds to a ViT’s representation token (referred in 2.3). To further clarify that, we provide an illustration of the embedding generation using ViT in figure 3.3. Moreover, the information flow between the embedding functions is mainly manifested using the transformer’s attention mechanism. That is done by exploiting its flexibility and concatenating tokens along the feature vector as we are going to describe in detail in the sections to come.

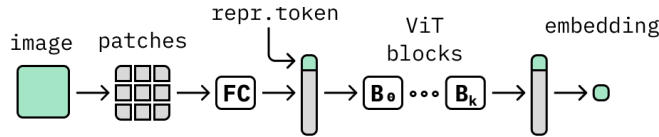


Figure 3.3: A simplified illustration of the embedding generation using ViT. An image is split into patches which are linearly projected using a fully connected layer for each patch. Then a randomly initialized representation token is added to the sequence of the linear projected patches. Afterwards, the feature vector is refined through the ViT blocks and finally the resulted representation token corresponds to the image embedding.

## 3.5 Proposed Methods

### 3.5.1 ParallelVits method

*ParallelVits* is the simplest method that we employ. It is comprised of two embedding functions (i.e.  $g$  and  $f$ ) that project samples independently on the embedding space. That is to say, that there is no information flow between support and query sets during the generation of the embeddings. The samples are embedded by leveraging only prior knowledge encoded in the embedding function’s parameters. In other words, this is a task-invariant method.

More concretely, given a task consisting of a support set  $X_s$  and a query set  $X_q$ , *ParallelVits* embeds each sample independently (i.e.  $g(X_s)$  and  $f(X_q)$ ). A similarity metric  $s(\cdot, \cdot)$  is then used to calculate the similarity scores between query and support embeddings (i.e.  $s(g(X_s), f(X_q))$ ). Finally, the support samples are classified to the categories of the query samples that are closer in the embedding space. To further clarify this method we provide an outline and an example in figures 3.4 and 3.5, where  $ViT_0$  and  $ViT_1$  serve as the embedding functions  $g$  and  $f$  respectively.

In addition, it should be noted that the *ParallelVits* method is very similar to Matching Nets [24] and Prototypical Nets [31] in the one-shot case (i.e. for

tasks where one sample per class is provided). Furthermore, in our case study, this method undertakes the role of the performance baseline. To demonstrate that methods exploiting task-specific information are beneficial, they have to surpass the *ParallelVits* performance.

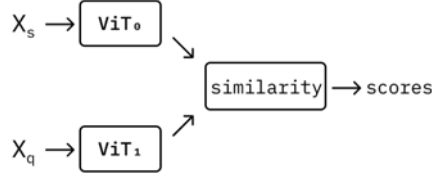


Figure 3.4: *ParallelVits* method: an outline. The support and query sets are embedded independently and then the similarity scores between them are computed for the classification process to proceed.

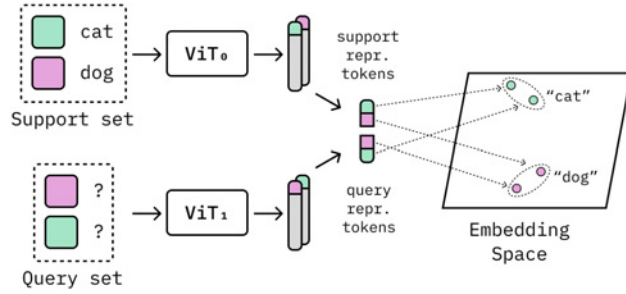


Figure 3.5: *ParallelVits* method: an example. The representation tokens are vectors in the embedding space. The query samples are classified to the classes of support tokens that are closer in the embedding space.

### 3.5.2 ParallelVits+Encoder method

Although *ParallelVits* is an effective method it does not incorporate task-specific information. To alleviate that, we incorporate a transformer encoder in the *ParallelVits* architecture resulting in a new hybrid method that we term as *ParallelVits plus Encoder*. More precisely, instead of having two embedding functions (i.e.  $f$  and  $g$ ), we introduce a third one (i.e.  $h$ ) that is applied on top of the other two and enables the unrestricted flow of information in all directions. Consequently, the generation of embeddings leverage prior information encoded in the trained parameters of the embedding functions and additionally exploit information between and within the support and query sets. An illustration of the outline of the method is provided in figure 3.6.

Given a task consisting of a support set  $X_s$  and a query set  $X_q$ , the method begins by embedding each sample independently (i.e.  $g(X_s)$  and  $f(X_q)$ ) as in

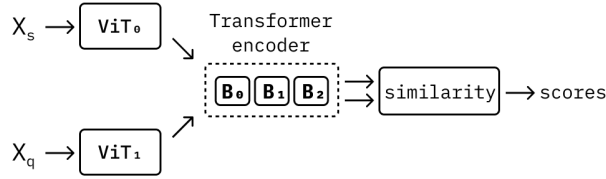


Figure 3.6: *ParallelVits+Encoder* method: an outline. The support and query samples are initially embedded independently by the two illustrated ViTs. Then the resulting embeddings are processed by a transformer encoder that outputs the contextualized embeddings. Finally, the similarities between them are calculated.

the *ParallelVits* method. Additionally, the generated embeddings are concatenated and then provided to a subsequent embedding function to be jointly refined (i.e.  $h(c(g(X_s), f(X_q)))$ , where  $c(\cdot, \cdot)$  is the concatenation operation.). Finally, the similarity scores are calculated between the contextualized support and query embeddings in order for the classification process of the query samples to proceed. An example that illustrates how the concatenation process takes place can be found in 3.7.

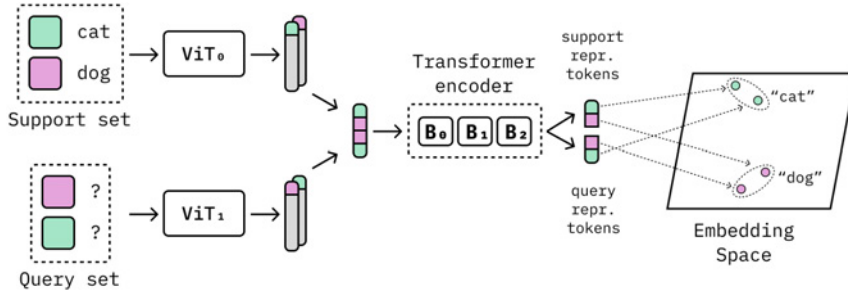


Figure 3.7: *ParallelVits+Encoder* method: an example. From the ViTs output the patch tokens (painted in gray) are discarded. The representation tokens are concatenated in the token dimension and then are fed to the transformer encoder. Finally, query samples are classified to the categories of support tokens that are closer in the embedding space.

From the implementation’s perspective, the additional embedding function  $h$ , that we call transformer encoder, is comprised of transformer encoder blocks (described in 2.3) which are being initialized using the employed pre-trained ViT. The actual number of those encoder blocks has been chosen empirically. Furthermore, this transformer encoder is position agnostic as positional tokens have not been added. In other words, it process the representation tokens as a set. Finally, another detail regarding the encoder is that a randomly initialized representation token is not supplied as it happens with ViT, since the goal is to refine the provided ones.

### 3.5.3 BlendedVits method

*ParallelVits plus Encoder* method allows information to be exchanged between representation tokens through a transformer encoder after they have been separately encoded by the employed ViT architectures. Nevertheless, since different kind of features are built along a multi-layer neural network, from low-level ones such as edges to high-level ones like object parts, there could be some merit to earlier information exchange. That is precisely the rationale behind *BlendedVits* method. More specifically, *BlendedVits* method consists of two ViTs, similarly to the *ParallelVits* method. However, it allows query and support representation tokens, after a specified query network’s block, to be jointly encoded. Hence, this method (outlined in figure 3.8) is categorised as a hybrid one since it incorporates both task specific and task invariant information to project the samples on the embedding space.

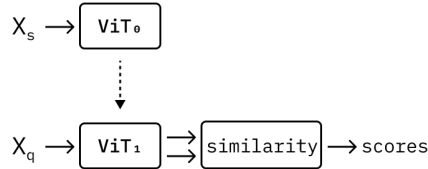


Figure 3.8: *BlendedVits* method: an outline. The support and query sets are embedded independently up to a point. After that, support and query representation tokens are jointly refined. The process proceeds with the calculation of similarity scores for the classification of the query samples.

More concretely, given a task consisting of a support set  $X_s$  and a query set  $X_q$ , *BlendedVits* embeds each of them independently up to a point. The support set is encoded through a specified number of layers of the support network and then is concatenated to each query feature vector at a specified query network’s block (we call it the *cat block*). In that manner, after the cat block, each query will be processed along with all the support representation tokens corresponding to the support set. Finally, the contextualized queries are classified with respect to the support tokens that have been contextualised with. An illustration of that process can be seen in figure 3.9.

It is worth mentioning that both the concatenation block in query network (i.e. *cat block*) and the support network’s block, from which support representation tokens are originated, are hyper-parameters of the method. In the provided illustration the support representation tokens of the last support network’s layer are used. However, experiments have been conducted where support representation tokens of previous blocks have been incorporated as well.

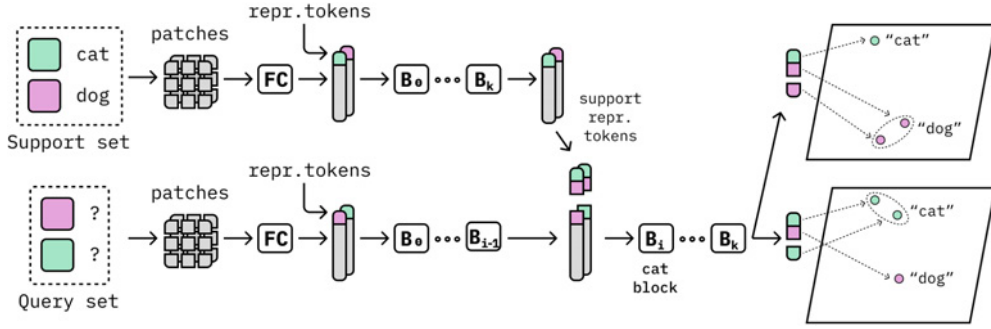


Figure 3.9: *BlendedVits* method: an example. Both support and query sets are embedded independently up to a point, using the ViT architecture. Then the support representation tokens are concatenated to each query feature vector at the cat block of the query network. After the concatenation each query is jointly processed with all the support representation tokens. At the end of the process, each query sample is classified with respect to the support representation tokens that it was contextualized with.

## 3.6 Methods variations

### 3.6.1 Attention Masks

Information flow between support and query sets can be achieved through both *ParallelVits+Encoder* and *BlendedVits* methods. However, to enable a fine-grained control over the information exchange between tokens and restrict any undesirable attention connectivity we incorporate the modality of attention masks. The rationale is that information propagation between some of the support and query tokens might be harmful to the performance of the methods. In that regard, we have explored several masking settings in search of the most beneficial attention connectivity.

To present these settings we should first briefly revisit the inner workings of the attention mechanism discussed in 2.3.3. The attention mechanism takes as input a feature vector  $X$  and linearly projects it, using three different learnable weight matrices, generating three versions of itself, namely Query  $Q$ , Key  $K$  and Value  $V$ . Then the multiplication of  $QK^T$  takes place to provide a weighting scheme for the Values vector  $V$ . That weighting scheme, when is normalized, using the softmax function, is called the *attention map*. Our attention mask  $M$  is added directly to the  $QK^T$  product before softmax operation takes place. The equation 3.1 shows how the new feature vector  $Z$  is calculated by applying the masked attention mechanism.<sup>2</sup>

<sup>2</sup>The scaling factor  $\sqrt{d_k}$  is being used for keeping the product  $Q \times K^T$  sufficiently small, to avoid that resulting in extremely small gradients [34].

$$Z = \underbrace{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)}_{\text{attention map}} \times V \quad (3.1)$$

The mask  $M$  is a two dimensional array that consists of two elements, these are  $\{0, -\infty\}$ . Since the mask is added to the product  $QK^T$  the elements that are added with 0 will not change and elements added with  $-\infty$  will get infinitely small. Thus, when the softmax operation is applied<sup>3</sup> the weights corresponding to the infinitely small elements will be zero and hence the corresponding elements of the Value vector  $V$  will be masked.

The masking settings are applied to the *BlendedVits* method after the concatenation of the support representation tokens to the query feature vector. Consequently, a query feature vector will consist of the support representation tokens, the query representation token and the query patch tokens. To present our mask settings we consider a 2-way 1-shot task illustrated in 3.10.

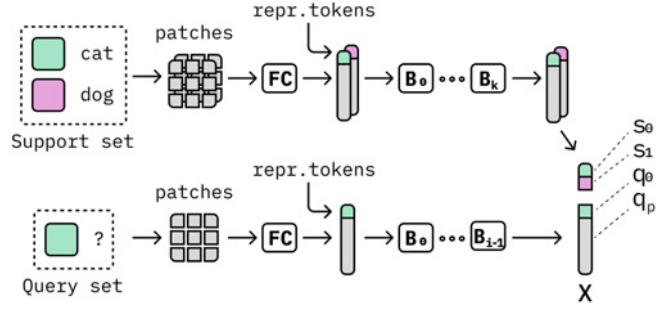


Figure 3.10: *BlendedVits* method: During the concatenation phase. The support representation tokens are denoted by  $s_0$  and  $s_1$ , the query representation token by  $q_0$  and the query patch tokens by  $q_p$ . The resulted feature vector is denoted by  $X$ .

After the concatenation, the subsequent attention block will process the feature vector  $X$  using the the masked attention operation (equation 3.1). It will first compute the product  $QK^T$ , add the mask  $M$  and apply the softmax operation. Then the new feature vector  $Z$  will be calculated using the masked attention map and the Value vector  $V$ . For our three proposed mask settings an illustration of that step is depicted in figure 3.11.

The first attention mask setting (i.e. setting 1) is designed to restrict attention for support representation tokens only to themselves. Consequently, by applying this mask, a support representation token cannot attend to other tokens in the feature vector. The second attention mask setting (i.e. setting 2) that was explored is a super-set of the aforementioned mask setting. Not only it restricts the attention between support representation tokens, but also does not allow the query representation token to attend support representation tokens and vice versa. Last

<sup>3</sup>softmax is applied in a row-wise manner, thus every row will sum to one.

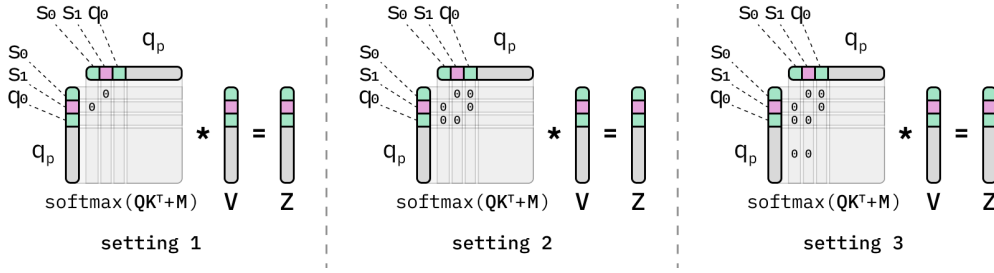


Figure 3.11: Mask settings (1-3). This figure displays how the masked attention map is multiplied by the vector  $V$  to compute the new feature vector  $Z$ . The zeroes in the masked attention map are the entries that have been masked using the corresponding setting. Note, that each row of the masked attention map sums to one.

but not least, the third attention mask setting (i.e. setting 3) is a super-set of the above mask settings and additionally does not allow query patch tokens to attend the support representation tokens.

To clarify further our mask settings we demonstrate the attention connectivity of the support representation tokens (i.e.  $s_0$  and  $s_1$ ) for mask setting 1 in figure 3.12.

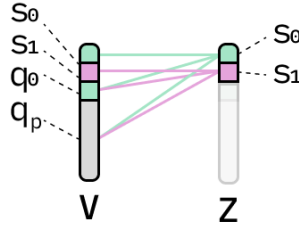


Figure 3.12: Attention connectivity for support representation tokens (i.e.  $s_0$  and  $s_1$ ) using mask setting 1. Each support representation token attends all the tokens of the value vector  $V$  except other support representation tokens. Precisely, the support representation tokens do not attend each other.

### 3.6.2 Cross Attention

Cross-attention (aka encoder-decoder attention) is a type of attention that is found in the conventional NLP transformer [34]. This attention type has the characteristic of not modifying its target (i.e. the tokens that it attends to). In other words it provides a way to facilitate one-way information flow between the support and query sets.

We have experimented with cross-attention both in the *BlendedVits* and *ParallelVits+Encoder* methods. Specifically, in the *BlendedVits* method we have applied cross-attention from query patch tokens to support representation tokens and vice



versa. Furthermore, in *ParallelVits+Encoder* cross-attention has been applied between support and query representation tokens in both directions as well.

Implementation wise, cross-attention can be simulated by performing the self-attention operation and then by reverting the attended tokens in the new feature vector to the previous state (i.e. before self-attention operation takes place.). Schematically, that can be seen in figure 3.13.

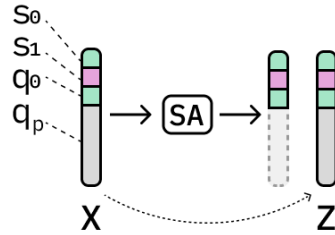


Figure 3.13: Cross attention to the query patch tokens example. The self-attention operation (denoted as SA) is applied to the feature vector  $X$  resulting in a feature vector  $Z$  where the query patch tokens  $q_p$  are reverted to their initial state.

### 3.6.3 Auxiliary Losses

Auxiliary losses is an additional modality that we apply in combination with the *ParallelVits+Encoder* method in an attempt to enhance the supervision signal. In that regard, we employ two different approaches. Both of them compute additional losses that eventually are aggregated by summation in a cumulative loss. This loss is then utilized to update the weights of the model.

The first approach (i.e. setting 1) computes similarities at different points in the network. Specifically, in addition to the default loss, it computes losses before and between the encoder blocks of the extra transformer encoder. On the other hand, the second approach (i.e. setting 2) calculates similarities between representation tokens that have been processed from the transformer encoder (we call them contextualized) and representation tokens that have not (we call them initial). In particular, the auxiliary losses are computed over the following pairs of representation tokens: (i) initial support with contextualized query tokens and (ii) initial query with contextualized support tokens. Both of the aforementioned methods are illustrated in figures 3.14 and 3.15 respectively.

An issue that arises by employing this modality, is that we end up having three different similarity score matrices (i.e. one for each loss), from which we should pick one to return as the model's prediction. In that regard, we have explored either returning one those matrices or aggregating those three by mean or max reduction.

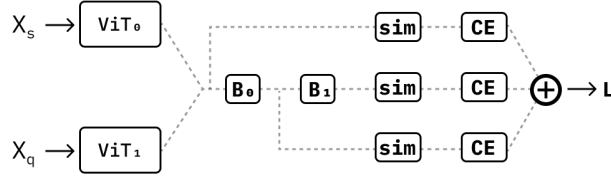


Figure 3.14: *ParallelVits+Encoder* method with auxiliary losses (setting 1). The auxiliary losses are computed over the similarity scores of support and query representation tokens before and between the encoder blocks (these are denoted by  $B_0$  and  $B_1$  respectively.).

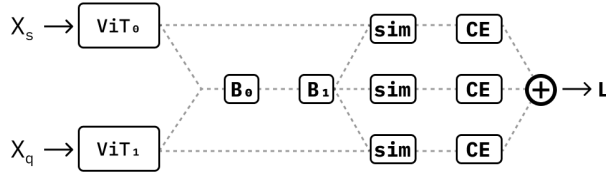


Figure 3.15: *ParallelVits+Encoder* method with auxiliary losses (setting 2). The auxiliary losses are computed over similarity scores of support and query representation tokens that have been encoded at a different level.

### 3.6.4 Incorporation of image patches

Broadly speaking, ViT architecture utilizes a randomly initialized token (i.e. representation token) that its sole purpose is to be refined through the attention blocks along with patch tokens and to eventually encapsulate information needed to minimize the objective’s loss (described in 2.3). In that regard, in both *BlendedVits* and *ParallelVits+Encoder* methods, we concatenate representation tokens to enable information exchange. However, that approach has the following shortcomings. First, by concatenating only representation tokens we might lose some information of patch tokens that has not being encoded yet. Secondly, the interaction of patch tokens with foreign representation tokens (i.e. representation tokens of the other feature vectors) might be beneficial. To explore those intuitions, we have implemented the modality of incorporating image patches.

In particular, for the *BlendedVits* method instead of concatenating only the support representation tokens to the query network, the support patch tokens of the support set are concatenated as well. Due to the fact that the complexity of attention mechanism is quadratic with respect to the number of participating tokens, this is done by applying an average pooling operation over the support patch tokens before the concatenation takes place. Consequently, the concatenation in the *BlendedVits* method proceeds as can be seen in figure 3.16. Moreover, In the *ParallelVits+Encoder* method, pooled support or query image patch tokens are incorporated in a similar manner (illustrated in figure 3.17) and interact with all support and query representation tokens.

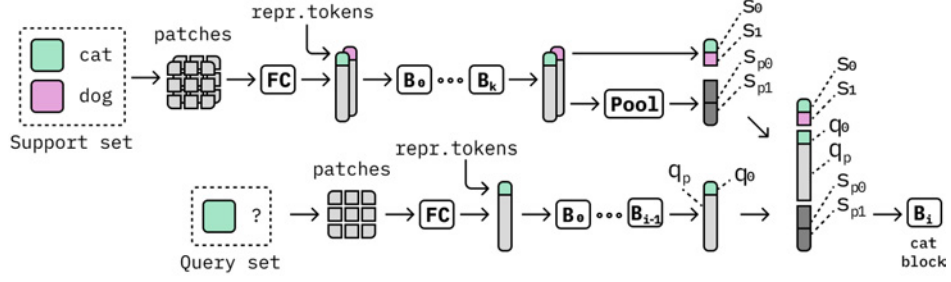


Figure 3.16: *BlendedVits* method: Incorporation of pooled patch tokens during the concatenation phase. The support representation tokens are denoted by  $s_0$  and  $s_1$ , the support pooled patch tokens by  $s_{p0}$  and  $s_{p1}$ , the query representation token by  $q_0$  and the query patch tokens by  $q_p$ .

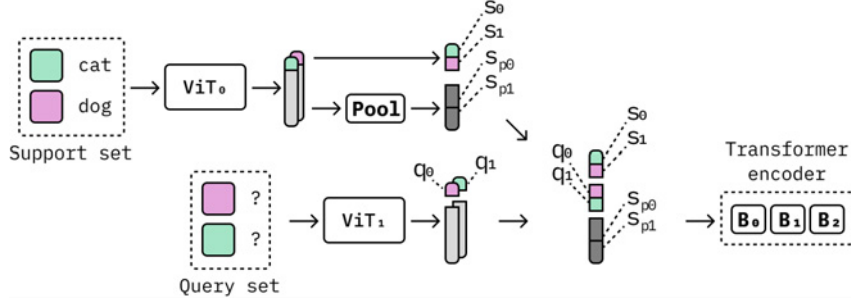


Figure 3.17: *ParallelVits+Encoder* method: Incorporation of pooled patch tokens during the concatenation phase. The support representation tokens are denoted by  $s_0$  and  $s_1$ , the support pooled patch tokens by  $s_{p0}$  and  $s_{p1}$  and the query representation tokens by  $q_0$  and  $q_1$ .

### 3.7 Other explored factors

In addition to our case study we have also explored other techniques to increase the overall performance of the proposed methods. Some of these techniques are described here.

#### 3.7.1 Similarity Metrics

Since metric-learning needs a metric to compute the distance between the acquired embeddings we have experimented with three different similarity metrics to evaluate their performance and choose the best performing one. In particular, we have explored cosine similarity, dot product similarity and euclidean similarity. Each of those measures has each own interpretation. *Euclidean similarity* measures the negative distance between the ends of the vectors, *Cosine similarity* measures the cosine of the angle  $\theta$  between the embeddings and finally the *Dot product similarity* is similar to cosine similarity but takes into consideration the

length of the vectors as well. For clarification purposes, we provide the formulas for these similarity measures given two embedding vectors  $a = [a_1 \ a_2 \ \dots \ a_n]^T$  and  $b = [b_1 \ b_2 \ \dots \ b_n]^T$  in table 3.1.

Similarity	Formula
Euclidean	$-\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$
Cosine	$\frac{a^T b}{ a  b }$
Dot Product	$a_1 b_1 + a_2 b_2 + \dots + a_n b_n =  a  b \cos(\theta)$

Table 3.1: The explored similarity measures.

### 3.7.2 Model’s weights state

Inspired from the impactful work of Siamese Nets [14] we have explored three states for the parameters of the employed VITs that undertake the role of embedding functions. Specifically, we have explored three settings, these are namely shared weights, independent weights and frozen support network’s weights. In the shared weights state, the two embedding functions are sharing their parameters or in other words they are identical (i.e.  $f \equiv g$ ). In the second setting, where the weights are independent, the embedding functions are allowed to be different (i.e.  $f \neq g$ ). Finally, the third setting, where the parameters of the support network are frozen, the embedding functions differ as well. However, in that setting the support network is constrained to preserve its initial weights state, that is the state of the employed pre-trained VIT.

### 3.7.3 Optimal training parameter set

A group of experiments has been dedicated to explore whether training only a subset of the network’s parameters has any positive effect on the acquired performance of the methods (i.e. the classification accuracy). This experiment group was inspired by the observation that when the support and query networks share their parameters the classification accuracy increases. More specifically, our intentions were to explore whether the observed increase in the performance was due to the smaller number of parameters that were trained. Some of the explored settings, under this experiment group, were to train all the parameters of the model not related to the attention mechanism, to train all the parameters of the model instead of those in the FFN located in the transformer blocks, to train only those parameters in the self-attention mechanism and to train only those parameters included in the normalization layers of the transformer blocks.

### 3.7.4 Artificially increase the number of shots

An FSL task is specified by the number of sampled classes and the number of samples per class (aka shots). As the number of shots increases the task becomes more trivial and consequently the performance of the method increases as well. Thus, an idea is to artificially increase the number of shots of a task. For example, for a one-shot task it would be ideal if we could use an algorithm to generate more samples based on the provided samples. In that regard, we have employed two approaches. Firstly, we attempted to increase the number of shots by augmenting the given samples. In particular, we have used a set of augmentations. These are random crop followed by a horizontal flip, colour jittering followed by colour dropping and Gaussian blur<sup>4</sup>. By applying those augmentations to the given samples we artificially increased the number of shots. As a second approach, instead of augmenting the provided samples, we utilized support representation tokens from previous layers of the support network. More precisely, in the similarities calculation we also included the support representation tokens of the last  $k$  layers of the support network. In both of the aforementioned approaches, the extra support representation tokens are handled similarly to more than one shot case (described in 3.9.2).

### 3.7.5 Training with more classes

Some works, such as [75] suggest that meta-learning is mostly effective over whole-classification training when novel classes are similar to base classes. Specifically, by sampling more categories during the meta-train phase they imply that the learnt feature representations demonstrate better class transferability. In that regard, we performed a group of experiments where during the training phase more classes were sampled. For example in a five-way one-shot task, during training, 10 classes were sampled instead of five.

### 3.7.6 Fine-tuning at meta-test time

As Vinyals et. al [24] and Hu et. al [85] have pointed out, fine tuning can be beneficial for FSL algorithms. Specifically, fine-tuning during the meta-test phase can significantly increase the performance especially when out-of-distribution data are included in the test set. For that reason, we have employed fine-tuning at meta-test time. This modality is implemented by augmenting support samples from the test set using an augmentation policy similar to 3.7.4. The augmented samples undertake the role of the query samples to perform a few optimization steps and then the classification of the actual query samples takes place.

---

<sup>4</sup>This augmentation policy was adopted by the SimCLR paper [55].

## 3.8 Datasets

Throughout this work most of the experiments have been conducted on the mini-ImageNet dataset. This dataset has been put together by Vinyals et. al [24] and contains 60K images of 100 categories. Thus, there are 600 images per category with spatial dimensions of 84x84. Ravi et. al [90] have proposed a category split of this dataset that is adopted by a lot of works and is used to benchmark FSL algorithms. In particular, they split the data to 64, 16, and 20 categories for training, validation and testing, respectively. Furthermore, we have also conducted some experiments on the CIFAR100 [9] dataset. This dataset contains 100 categories with each of them containing 600 images as well. We have manually split this dataset into a 80 categories for training and 20 categories for testing.

## 3.9 Implementation details

### 3.9.1 Network architecture

The employed network architecture for this work is a ViT-S/16 with 21M parameters. Hence, it is comprised of 12 blocks, each multi-head attention module has 6 heads and the feature dimension is 384. Additionally, the expected image size is 224x224 and the patch size is 16. For images of smaller size a scale up operation takes place, using bi-linear interpolation. It is worth mentioning, that we drop the MLP network that is placed at the top of the ViT architecture and plays the role of the classifier. That is because we are interested only for the feature representation vectors.

### 3.9.2 Classification Head

The classification is done by employing a fixed measure of similarity (e.g. cosine similarity). There are two cases which need to be addressed for the classification of query samples, with respect to the support set. Firstly, in the one-shot setting, where one sample per class is provided, the query sample is classified in the category of the nearest support sample in the embedding space, similarly to both Matching Nets and Prototypical Nets [31]. Secondly, when more than one sample per category is provided, the similarities between all support-query pairs are computed and then aggregated per category (e.g. by taking the maximum similarity score). The resulted representative similarity scores per category are used to classify the query sample.

### 3.9.3 Training details

Each method is comprised of two ViT architectures. As described above, we incorporate three weight states (i.e. shared weights, frozen support weights and independent weights). In either case, the computational graph includes both of the architectures and each method is trained in an end-to-end fashion. The most

important hyper-parameter for training neural networks is the learning rate [20]. A learning rate has been chosen by conducting a coarse grid search for our methods. From the learning rate’s ( $lr$ ) domain  $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$  we have found that  $lr = 10^{-6}$  is the most suitable having a batch size<sup>5</sup> (i.e. number of tasks processed in parallel) equal to one. Greater learning rates resulted in degrading model’s performance, probably because they harm the already learnt representations of the pre-trained model. With smaller learning rates we observed that after the first few episodes the performance of the model was not increasing. A possible explanation could be that the optimization procedure is trapped in some local minima.

As per the optimizer, we utilize the AdamW optimizer [51] which is one of the state-of-the-art optimizers for training neural networks, coupled with a cosine scheduler to achieve faster convergence. In addition, we have found that 16K episodes were enough to achieve convergence given that the best validation performance is recorded before reaching the maximum number of episodes, having a minimum margin of 960 episodes and a median margin of 5600 episodes. The resulted model checkpoint which we use for evaluation is that with the higher validation accuracy.

In addition, the classification loss which is utilized is the standard cross-entropy loss preceded by a softmax operation. In particular, the similarities calculated between a query and the support embeddings are passed through the softmax to be converted to probabilities and then the cross-entropy loss is computed.

### 3.10 Workflow

The proposed workflow consists of three phases. These are namely, pre-train phase, meta-train phase and meta-test phase. In pre-train phase the feature backbone, that is used as an initialization for the employed embedding functions, is pre-trained using the DINO methodology described in 2.4 on external data. In the meta-train phase one of the proposed meta-learning methods is being trained using the meta-learning framework, explained in 2.6.1, on the training categories of the given dataset. Finally, the meta-trained method is evaluated on the test categories of the dataset. To put all of our framework’s components together schematically, we provide a diagram (3.18) that illustrates the general workflow.

---

<sup>5</sup>We use batch size = 1 throughout this work.

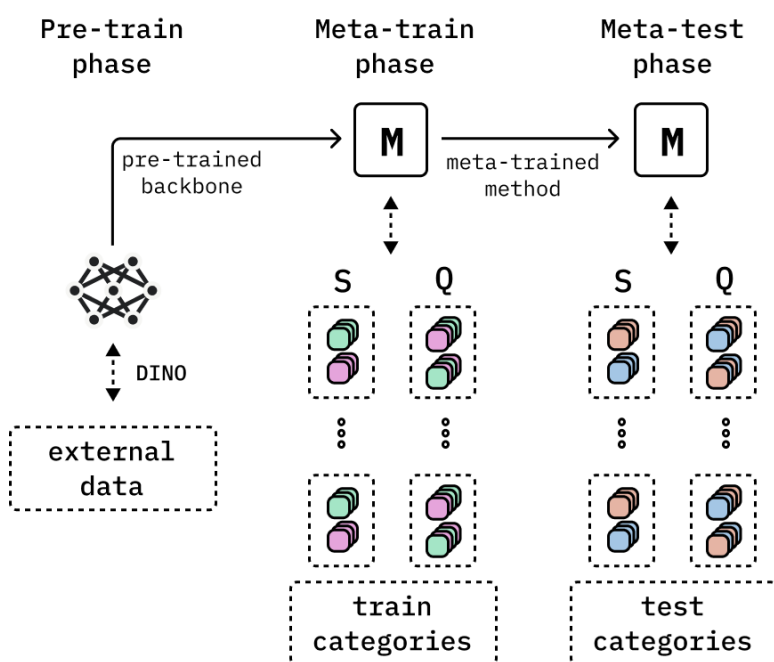


Figure 3.18: The proposed workflow illustration. The meta-learning method is denoted by  $M$ . The support and query sets that form the training and test tasks are denoted by  $S$  and  $Q$  respectively.



## Chapter 4

# Evaluation and Conclusions

In this chapter, we present the results of the conducted experiments, discuss our conclusions and compare our best-performing methods with related works in the literature. Specifically, in the first section we elaborate on the evaluation details. In the second section, we demonstrate how several architectural choices impact the classification performance, summing up with our conclusions and the presentation of the best settings. Finally, we conclude the chapter by comparing our best settings with other works.

### 4.1 Evaluation details

Throughout the evaluation process, we use a single evaluation metric. In particular, we utilize classification accuracy defined as the ratio of correct classifications made over the total classifications attempted. Although a single evaluation metric might not witness the full story, comparisons between several settings become straightforward.

During the evaluation procedure, we randomly sample two thousand episodes of the specified task format (e.g. 5way-1shot task) averaging the classification accuracy over them. In addition, during model selection, we evaluate the experiment settings on the validation set. Having concluded to our best models, we additionally evaluate them on the test set to compare them with other works. Specifically for the test set evaluation we run each evaluation five times and report the average performance to further minimize the variation in the acquired results.

It is worth mentioning, that the majority of the performed experiments have been conducted under the 5way-1shot setting, a usual benchmark for FSL algorithms, and thus enabling the comparison of our proposed methodology with other works. Nevertheless, we perform some experiments for 5way-5shot tasks to assert the normal behavior of our methods (i.e. as shots increase the task becomes easier, and the methods' performance increases.)

## 4.2 Evaluation of the proposed methodology

### 4.2.1 Overview

This section begins by presenting the selection of two main hyper-parameters that we adapt throughout the evaluation. These are the employed similarity metric, which is used to classify the query samples using the corresponding embedding vectors, and the weight states (e.g. shared parameters) of the ViT models that undertake the role of the embedding functions. The performance of our methods, under those hyper-parameters, is regarded as a baseline performance, which is used to evaluate the benefits of the explored modalities. We continue by presenting separately how our method variations affect the *BlendedVits* and *ParallelVits+Encoder* methods baseline performance. In addition, we present our best settings and we conclude by demonstrating experiments conducted to assert the normal behaviour of our methods and other attempts to increase their performance.

### 4.2.2 Main hyper-parameters

#### 4.2.2.1 Weights states

As described in 3.7.2, we have explored three settings for the parameters of the embedding functions (i.e. ViT models). These are namely shared weights, independent weights and frozen support network’s weights. The effect that these settings exert, on the performance of our methods, can be seen in table 4.1.

Shared weights seem to demonstrate the best performance across all the methods, meanwhile independent weights follow with a narrow margin. Finally, frozen support weights obtain the worst performance. Interestingly enough, although independent weights result in greater model’s capacity and flexibility, we do not observe over-fitting. We attribute that fact to the adopted meta-learning framework. In addition, frozen support weights performance indicates that it is necessary for the support network parameters to be updated and that the flexibility provided by only the trainable query network is not sufficient. Since shared weights achieve the best performance for all our methods, in the experiments to come, we use them by default unless stated otherwise.

Weights	<i>BlendedVits</i> <sup>1</sup>	<i>ParallelVits</i>	<i>ParallelVits+Enc</i>
Frozen support	88.19%	89.34%	88.69%
Independent	91.24%	91.26%	90.69%
Shared	<b>91.43%</b>	<b>92.25%</b>	<b>91.99%</b>

Table 4.1: The effect of weight states on the proposed methods.

<sup>1</sup>The initial version of *BlendedVits* concatenates the support representation tokens to the first block (cat block = 0).

#### 4.2.2.2 Similarity Measures

Similarity measures have been utilized to calculate similarity scores between the embedding vectors and eventually classify the query samples, as described in 3.7.1. Specifically, *Cosine*, *Dot product* and *Euclidean* similarities have been explored for all our methods. The effect of those similarity measures is illustrated in table (4.2).

Apparently, *Euclidean* similarity is the clear winner across all methods, meanwhile, *Dot product* similarity and *Cosine* similarity follow. For the *BlendedVits* method, *Dot product* obtains better results than *Cosine* similarity. However, for the *ParallelVits* and *ParallelVits+Enc* methods, *Cosine* similarity performs better. It is worth noting that in related works [85, 31, 24] there is no consensus on the choice of similarity measure. However, the choice is vital and is usually selected empirically. Throughout this work, we adapt *Euclidean* similarity as the default similarity measure.

Similarity	<i>BlendedVits</i>	<i>ParallelVits</i>	<i>ParallelVits+Enc</i>
Cosine	88.74%	89.93%	90.65%
Dot Product	90.29%	89.43%	90.51%
Euclidean	<b>91.43%</b>	<b>92.25%</b>	<b>91.99%</b>

Table 4.2: The effect of similarity choice on our methods.

#### 4.2.2.3 Baseline performance

To demonstrate the impact of the employed modalities (described in 3.6) the initial methods performance (i.e. baseline performance) should be presented first. In that way, it is feasible to conclude whether the applied modifications affect positively or negatively the performance of the method at hand. The following table 4.3 summarises the baseline performance for the proposed methods.

<i>BlendedVits</i>	<i>ParallelVits</i>	<i>ParallelVits+Enc</i>
<b>91.43%</b>	<b>92.25%</b>	<b>91.99%</b>

Table 4.3: The baseline performance of our methods.

### 4.2.3 *BlendedVits*

#### 4.2.3.1 Concatenation block

In *BlendedVits* method we have experimented with the *cat block* hyper-parameter as mentioned in 3.5.3. That is the block where the support representation tokens, of the last layer of the support network, are concatenated in the query network. In the initial version of *BlendedVits* method *cat block* was set to the first block of ViT network. In those experiments we perform a grid search for the *cat block*

hyper-parameter for each weight state to see if a better choice for the *cat block* can be made. We therefore present the best performance and the corresponding *cat block* in table 4.4. From the results, we observe that for each weight state the best performance is achieved with a different choice of *cat block*. Another conclusion, is that there is a general tendency for *cat blocks* deeper in the query network to result in better performance. That is expected since features learnt by the last layers of the network are more similar to those extracted from the last layer of the support network.

Additionally, we have conducted experiments where instead of concatenating the last version of support representation tokens we concatenate the support representation tokens of the respective layer. That was done under the assumption that respective blocks output similar features. However, the results of those experiments did not show any improvement probably because of the fewer transformer blocks, that were available, for support representation tokens to be embedded properly.

Weights	Best performance	cat block <sup>2</sup>
Shared	91.78%	9
Independent	91.24%	11
Frozen support	88.30%	11

Table 4.4: The effect of weight states on the proposed methods.

#### 4.2.3.2 Attention mask settings

One of our most effective modalities is the use of attention masks, described in 3.6.1. This modality applies to the *BlendedVits* method and consists of three main settings. The first setting (i.e. setting 1) restricts support representation tokens from attending each other. The second setting (i.e. setting 2) is a super-set of the first, meanwhile it restricts attention between support representation tokens and the query representation token. Finally, the third setting (i.e. setting 3) is also a super-set of the aforementioned settings but does not allow query patch tokens to attend support representation tokens. The effect of those settings on the baseline performance of the *BlendedVits* method can be seen in table 4.5. The results show that setting 2 is the most beneficial. Setting 3 follows and setting 1 does not positively affect the performance of the model. The main conclusion is that when support representation tokens attend the query representation token, a decrease is observed in method’s performance. Additionally, since setting 3 has worse performance than setting 2, we claim that the attention of query patch tokens to support representation tokens is beneficial.

<sup>2</sup>The provided numbers correspond to block indexes in [0,11].

<sup>3</sup>Relative to *BlendedVits* with cat block 9.

Settings	Accuracy	Relative increase <sup>3</sup>
setting 1	91.73%	-0.05
setting 3	92.13%	+0.35
setting 2	92.31%	<b>+0.53</b>

Table 4.5: Attention mask settings on *BlendedVits* method using *cat* block 9.

#### 4.2.3.3 Cross-attention

Cross-attention falls under the attention connectivity experiments and is extensively described in 3.6.2. The main purpose of cross-attention is to modify the processed feature vector without changing the attention target. In other words, it uses the attention target as a constraint on the other tokens. Cross-attention has been performed in two directions. These are support representation tokens attending to query patch tokens and vice versa. We demonstrate the results of those experiments in table 4.6. At first glance, cross-attention modality seems to not positively affect the performance of the method. However, given the promising results of the attention masking we conducted experiments where those modalities are combined. Specifically, in table 4.7, we utilize the second attention mask along with the cross-attention modality. In that manner, query representation token does not attend support representation tokens. The results indicate, that the combination of those modalities increase the baseline performance when support representation tokens undertake the role of the attention target.

Target	Accuracy	Relative increase <sup>4</sup>
support repr.	91.72%	<b>-0.06</b>
query patches	88.25%	-3.53

Table 4.6: Cross-attention in *BlendedVits* method.

Target	Accuracy	Relative increase
support repr.	92.24%	<b>+0.46</b>
query patches	91.47%	-0.31

Table 4.7: Cross-attention coupled with mask setting 2 for *BlendedVits* method.

#### 4.2.3.4 Image patches

The incorporation of image patches is another modality that we apply to *BlendedVits* method as described in 3.6.4. In this method we additionally include the support pooled patch tokens when support representation tokens are concatenated

<sup>4</sup>Relative to the performance of the method without cross-attention.

to the query feature vector. In particular, we have experimented with several sizes of pooled patch tokens, these are 4, 8 and 16. The results of this modality are presented in table 4.8.

As it is clear from the presented results, *BlendedVits* method does not seem to benefit from this modality. Moreover, a more significant decrease in performance, can be observed, as the size of pooled patch tokens increases. We attribute the negative effect of this modality to the attention connectivity between pooled patch tokens and representation tokens.

In that regard, we have designed a generalization of mask setting 2, depicted in 4.1, that restricts the attention connectivity from the support pooled patch tokens to both representation token types (i.e. setting 4). The results of this combination of modalities can be observed in the table 4.9. The combination of those modalities seems to positively affect the performance of *BlendedVits* method. Nevertheless, we still observe that an increase in the size of pooled patch tokens results in worse performance.

Size	Accuracy	Relative increase
4	91.71%	<b>-0.07</b>
8	91.71%	-0.07
16	91.51%	-0.27

Table 4.8: The image patches modality combined with *BlendedVits* method.

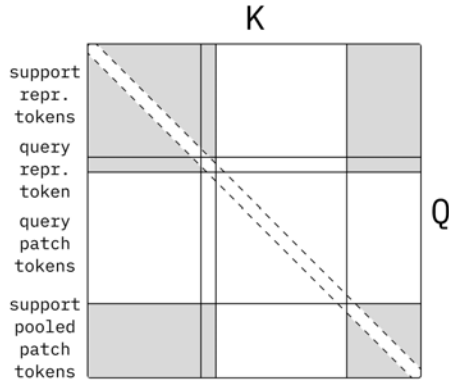


Figure 4.1: The mask setting 4. Areas painted in gray are masked. In particular, attention connectivity is restricted in a way that interaction involving support pooled patch tokens is only allowed between them and query patch tokens.

Size	Accuracy	Relative increase
4	92.23%	<b>+0.45</b>
8	92.16%	+0.38

Table 4.9: The image patches modality combined with attention mask setting 4.

#### 4.2.4 *ParallelVits+Encoder*

##### 4.2.4.1 Encoder blocks

The encoder of the *ParallelVits+Encoder* method consists of transformer encoder blocks. Those blocks have been initialized from the employed pre-trained ViT. To choose which blocks to incorporate we have conducted some experiments. Specifically, we explore block combinations which are located deeper in ViT’s architecture since they have trained to extract high-level features. The rationale behind this choice, is that those blocks’ parameters need fewer optimization steps than other blocks, early in the architecture, to provide a useful representation. The table 4.10 displays the explored combinations and the corresponding classification accuracy. From the acquired results, we observe three general trends. First, blocks that are closer to the last layer of the architecture obtain better results. Second, it seems that combinations of two blocks are more effective than using a group of three blocks or a single block. Finally, by employing the same block twice we do not see any improvement. Throughout the rest of the experimentation with *ParallelVits+Encoder* method, we use by default the combination of the 11th and 12th blocks (these correspond to indices 10,11).

ViT blocks <sup>5</sup>	Accuracy
10	91.71%
8, 9, 10	91.73%
11	91.74%
8, 9	91.77%
9, 10, 11	91.80%
9, 10	91.92%
10, 11	91.99%
11, 11	<b>91.99%</b>

Table 4.10: The effect of pre-trained blocks selection on the *ParallelVits+Encoder* method.

---

<sup>5</sup>The provided numbers correspond to block indexes in [0,11].

#### 4.2.4.2 Auxiliary losses

Auxiliary losses have been integrated to the *ParallelVits+Encoder* method as mentioned in 3.6.3. In particular, we employ two types of auxiliary losses. The first setting computes the auxiliary losses before and between the encoder blocks, meanwhile the second setting computes auxiliary losses over embeddings that have not been encoded yet by the extra blocks and those that have. The table 4.11, demonstrates the impact of those settings on the performance of *ParallelVits+Encoder* method.

It is apparent that setting 2 is more beneficial than setting 1 on the *ParallelVits+Encoder*. Furthermore, it is worth mentioning that the aggregation of similarity matrices, which arise from the multiple loss calculations, has been done by mean and max reductions or by selecting one of the similarity matrices, as described in 3.6.3. Interestingly enough, we have found empirically that the latter yields the best results for both settings.

Settings	Accuracy	Relative increase <sup>6</sup>
setting 1	92.06%	+0.07
setting 2	92.13%	<b>+0.14</b>

Table 4.11: The effect of auxiliary losses on the *ParallelVits+Encoder* method.

#### 4.2.4.3 Cross-attention

Similarly to *BlendedVits* method, we have incorporated cross-attention modality with *ParallelVits+Encoder* as well. Nevertheless, here is applied in a different context. Specifically, cross-attention takes place over all the concatenated representation tokens. In table 4.12, we present the results for two attention targets, these are the support representation tokens and the query representation tokens.

We can observe, from the presented results, that cross-attention modality does not significantly affect the performance of *ParallelVits+Encoder* method’s. Additionally, we observe that when query representation tokens serve as the attention target the decrease in performance is more significant.

Target	Accuracy	Relative increase <sup>7</sup>
support repr.	92.01%	<b>+0.02</b>
query repr.	91.66%	-0.33

Table 4.12: Cross-attention in *ParallelVits+Encoder* method.

<sup>6</sup>Relative to *ParallelVits+Encoder* accuracy without applying this modality.

<sup>7</sup>Relative to the performance of the method without cross-attention.



#### 4.2.4.4 Image patches

In *ParallelVits+Encoder* we have also incorporated image patches as described in 3.6.4. Similarly to *BlendedVits* method, we additionally include the average pooled support patch tokens when support and query representation tokens are concatenated together. In particular, we have experimented with several sizes of pooled patch tokens, these are 4, 8 and 16. The results of this modality are presented in table 4.13. In correspondence with the results obtained for *BlendedVits* method, we observe that image patches modality does not affect positively the performance of *ParallelVits+Encoder* method. In addition, as the size of pooled patch tokens increases, the performance of the method drops. Hence, image patches modality is not included in *ParallelVits+Encoder*'s best settings.

Size	Accuracy	Relative increase
4	91.94%	<b>-0.05</b>
8	91.90%	-0.09
16	91.75%	-0.24

Table 4.13: The image patches modality combined with *ParallelVits+Encoder* method.

#### 4.2.5 Conclusions and Best settings

Having experimented with all our main methods and their variations we summarise our conclusions. In the *BlendedVits* method we enable information exchange between support and query sets by concatenating the support representation tokens to each of the query feature vectors. Specifically, we concatenate the support representation tokens, of a block in the support network, to another block in the query network (i.e. cat block). In that regard we have found that the best performance is acquired using the last block of the support network, for fetching the support representation tokens, and the 10th block (i.e. cat block = 9) of the query network to concatenate them.

Furthermore, we have discovered that by restricting the attention connectivity between support representation tokens and the query representation token we obtained better results. Although that could indicate that information flow between support and query networks is not beneficial, the third proposed attention mask, where the query patches are restricted from paying attention to the representation support tokens as well, results in worse performance. That leads to the conclusion, that the interaction between the support representation tokens and query patch tokens is useful.

Moreover, by employing the cross-attention modality, which guarantees that the attention target will not be modified, we acquired another positive result when we combined it with the attention mask setting. Nevertheless, it is not clear

whether this result is due to the combination of cross-attention with the attention mask modality or due to the attention mask modality alone.

Finally, the incorporation of pooled support image patches coupled with the generalization of the second attention mask (i.e. setting 4) obtains better results than the *BlendedVits* performance baseline. That result might indicate that information stored in support image patches is useful when combined in the query feature vector.

Regarding *ParallelVits+Encoder* method, its purpose is to enable information flow between support and query sets by the concatenation of all the representation tokens. In contrast with, *BlendedVits* method, image patches are not incorporated at all. Its inferior performance, with respect to the *BlendedVits* method, might outline their significance in the process of information exchange.

Specifically for the *ParallelVits+Encoder* method, we have explored several choices for the the encoder blocks. We have found that the last blocks of the employed pre-trained ViT obtain the best performance and particular when they are combined in pairs. After examining several modalities, we have found that only the auxiliary losses result in better performance than the baseline.

Finally, for *ParallelVits*, which is a method where no exchange of information flow is allowed between support and query sets, we have not proposed any variation, since the purpose of the explored variation was to restrict or enhance the way information is exchanged. A conclusion regarding *ParallelVits* method; despite its simplicity, and by only incorporating prior information, is that it can perform on par with the other proposed methods.

In conclusion, we provide a summary of our best settings in table 4.14. The best validation accuracy is achieved by the *BlendedVits* method coupled with the second attention mask. The *ParallelVits* method is located in the second place of our results table, meanwhile other two variations of the *BlendedVits* follow. These are the cross-attention setting combined with the second mask setting, and the image patches incorporated with the mask setting 4. At the bottom of the table, the best *ParallelVits+Encoder* setting is located with the auxiliary losses.

Method	Variation(s)	Accuracy
BlendedVits	mask setting 2	<b>92.31%</b>
ParallelVits	N/A	92.25%
BlendedVits	cross-attention, mask setting 2	92.25%
BlendedVits	image patches, mask setting 4	92.23%
ParallelVits+Encoder	auxiliary losses	92.13%

Table 4.14: Our best settings on MiniImageNet’s validation set.

### 4.2.6 Other experiments

We have also conducted experiments in order to assert the normal behaviour of our methods. For those experiments, we have used our methods without any variation. Specifically, we have tested our methods on tasks where more categories are sampled. These are namely the 8way-1shot, the 11way-1shot and the 14way-1shot tasks. The table 4.15 depicts the performance of our methods on those tasks. Furthermore, we have evaluated our methods on the 5way-5shot task. The results of this evaluation are presented in table 4.16.

From the acquired results, it is clear that as the number of sampled categories increases the performance of our methods decreases. That is expected since the classification task becomes more demanding. For the five-shot experiments, since more annotated samples are provided, our methods perform better on those tasks. Interestingly enough, we observe that the order of our methods, with respect to the classification accuracy, is not constant. In particular, in the five-shot experiments, *BlendedVits* is ranked first in the results table. Meanwhile, when the number of sampled categories increases, *BlendedVits* has the worst performance.

Task	BlendedVits	ParallelVits	ParallelVits+Encoder
08way-1shot	87.80%	88.88%	89.33%
11way-1shot	84.97%	86.83%	86.69%
14way-1shot	81.81%	83.86%	83.79%

Table 4.15: The effect of sampling more classes on our methods performance.

Method	Accuracy
BlendedVits	<b>96.96%</b>
ParallelVits	96.88%
ParallelVits+Encoder	96.58%

Table 4.16: The effect of more samples per category (i.e. shots) on the baseline performance of our methods.

In addition to the aforementioned experiments, we have also made several attempts to increase the performance of the proposed methods. As described in 3.7, these are the training of a subset of models’ parameters, the artificial increase of the number of shots, the incorporation of more classes during the training phase and the models fine-tuning at the meta-test phase.

Unfortunately, those experiments did not result in an improvement to any of our methods. For fine-tuning at meta-test phase, we hypothesize, that the underlining cause is the small shift in the data distribution between train and test categories of miniImageNet dataset. In particular, as Hu et. al [85] have demonstrated, fine-tuning at meta-test time is beneficial only when a large shift in the distribution of the data has occurred. As per the other experiments, we

do not provide an explanation regarding the lack of improvement in our proposed methods, since more extensive experimentation is required.

### 4.3 Comparison with other works

In addition to comparisons made within our proposed methodology, we contrast our methods with other state-of-the-art related works. The comparison is established on the five-way one-shot task using the miniImageNet test set. It is important to mention that our results are not directly comparable to much of those works in both terms of architecture (e.g. number of model parameters) and prior knowledge (i.e. we use a pre-trained model). However, this comparison provides an insight of where our methods stand with respect to both impactful and recent works. Our best settings' results on the test set are provided in table 4.17, meanwhile the comparison with related works is presented in table<sup>8</sup>.

Method	Variation(s)	Accuracy
BlendedVits	cross-attention, mask setting 2	<b>93.51%</b>
ParallelVits	N/A	93.35%
BlendedVits	mask setting 2	93.26%
BlendedVits	image patches, mask setting 4	93.21%
ParallelVits+Encoder	auxiliary losses	93.07%

Table 4.17: Our best settings on MiniImageNet's test set.

---

<sup>8</sup>The table's contents have been adopted from [85].

Method	Backbone	Accuracy	Published in
Baseline++ [56]	CNN-4-64	48.2%	ICLR, 2019
ProtoNet [31]	CNN-4-64	55.5%	Nips, 2017
MetaOpt-SVM [49]	ResNet12	61.4%	CVPR, 2019
ProtoNet [58]	WRN-28-10	62.9%	ICCV, 2019
RS-FSL [69]	ResNet12	65.3%	BMVC, 2021
Fine-tuning [58]	WRN-28-10	65.7%	ICLR, 2020
Meta-Baseline [75]	ResNet12	68.6%	ICCV, 2021
SIB [77]	WRN-28-10	70.0%	ICLR, 2020
PLCM [78]	ResNet12	70.1%	ICCV, 2021
LST [50]	ResNet12	70.1%	NeurIPS, 2019
ProtoNet [74]	AMDIM ResNet	76.8%	ICASSP, 2021
EPNet+SSL [64]	WRN-28-10	79.2%	ECCV, 2020
CNAPS+FETI [80]	ResNet18	79.9%	WACV, 2022
PT-MAP [77]	RN-28-10	82.9%	ICANN, 2021
ParallelVits+Enc	ViT-small	<b>93.1%</b>	-
ParallelVits	ViT-small	<b>93.3%</b>	-
BlendedVits	ViT-small	<b>93.5%</b>	-
PMF [85]	ViT-base	95.3%	CVPR, 2022

Table 4.18: Comparison of our best settings with other related works.



# Chapter 5

## Discussion

### 5.1 Summary

In this work we have tackled the FSL image classification problem, using the notorious ViT architecture amplified with a self-supervised model pre-trained with DINO methodology. Specifically, we have proposed and evaluated three main methods and their variations. Each method has been designed to fulfill a different purpose. We began from *ParallelVits* method, which restricts the information exchange between the embedding functions, taking the role of the performance baseline in our case study. We continued by proposing two additional methods that in contrast with *ParallelVits*, they allow such exchange, in mainly two different ways.

Through experimental evaluation, we have found that our methods achieve near state-of-the-art performance on the five-way one-shot benchmark on miniImageNet. Additionally, we have acquired some interesting insights on how information flow can be incorporated in a more beneficial manner, using the attention mechanism of the transformer architecture.

Regarding our case study, even though, the results indicate that our methods incorporating information flow, perform equally or even surpass the performance achieved by the performance baseline (i.e. *ParallelVits*), we cannot ignore that this is happening with a narrow margin. One explanation could be that the achieved classification accuracy is already too high, rendering the benefits of information exchange hardly noticeable. Another explanation could be that a different approach should be followed in order to make the most out of information exchange, between the embedding functions. In either case there is plenty of follow-up work that could be conducted in this context to strengthen our conclusions and explore other techniques of information exchange.

## 5.2 Future Work

Our work could be extended in multiple directions. Those directions could be related to a more robust evaluation of our methods, an exploration of other modalities to incorporate information flow between the embedding functions, and an evaluation on other setting and forms of few-shot learning.

First of all, our experimentation has mostly taken place on the miniImageNet dataset. To alleviate that, a first direction would be to evaluate our methods in other datasets such as CifarFS [39], tiered-imageNet [42], meta-dataset [65] and CUB [10]. That would allow us to evaluate already implemented modalities, such as fine-tuning at the meta-test time, that require a greater distribution shift between train and test data.

Additionally, we could also observe how our methods perform on relevant but more demanding forms of FSL. Cross-domain FSL is where the training phase differs from the test one, not only in the categories utilized, but also in the underlying image distribution. Another popular form of FSL, we could explore, is the fine-grained FSL. In that form, categories are very similar to each other; making the problem more challenging. For instance, a candidate dataset for fine-grained FSL is the CUB [10] dataset, comprised of bird images only.

Furthermore, in our evaluation we enforce some hyper-parameters to be constant (e.g. the weight state). That is done for the sake of avoiding exponential growth in the number of experiments to be conducted, since unlimited resources are not available. However, this approach could hide an underlying trend or correlation. In that regard, another direction would be to conduct experiments under different values of those hyper-parameters, seeking unobserved findings.

Finally, an additional direction would be to compare our methods and their variations in other FSL settings. For example, the five-way five-shot setting is a usual benchmark for few-shot algorithms comparison, in literature. Apart from that, having more samples available per class, would enable for other approaches to be followed. For example, in the related work of Cross Transformers [59], they generate query aligned prototypes for each class by taking advantage of the multiple shots in the task setting.



# Bibliography

- [1] Warren S. McCulloch and Walter Pitts.  
“A logical calculus of the ideas immanent in nervous activity.”  
In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1, 1943), pp. 115–133.  
ISSN: 1522-9602. DOI: 10.1007/BF02478259.  
URL: <https://doi.org/10.1007/BF02478259> (visited on 02/18/2023).
- [2] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.”  
In: *Psychological Review* 65.6 (1958), pp. 386–408.  
ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519.  
URL: <http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519> (visited on 02/18/2023).
- [3] Jane Bromley et al.  
“Signature Verification using a ”Siamese” Time Delay Neural Network.”  
In: *Advances in Neural Information Processing Systems*. Vol. 6.  
Morgan-Kaufmann, 1993. URL: <https://proceedings.neurips.cc/paper/1993/hash/288cc0ff022877bd3df94bc9360b9c5d-Abstract.html>  
(visited on 03/01/2023).
- [4] Rich Caruana. “Multitask Learning.”  
In: *Machine Learning* 28.1 (July 1, 1997), pp. 41–75. ISSN: 1573-0565.  
DOI: 10.1023/A:1007379606734. URL:  
<https://doi.org/10.1023/A:1007379606734> (visited on 02/24/2023).
- [5] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997. 414 pp.  
ISBN: 0070428077.
- [6] Y. Lecun et al. “Gradient-based learning applied to document recognition.”  
In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.  
ISSN: 1558-2256. DOI: 10.1109/5.726791.
- [7] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell.  
“Learning to Learn Using Gradient Descent.”  
In: *Artificial Neural Networks — ICANN 2001*.  
Ed. by Georg Dorffner, Horst Bischof, and Kurt Hornik.  
Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001,  
pp. 87–94. ISBN: 9783540446682. DOI: 10.1007/3-540-44668-0\_13.

- [8] Jia Deng et al. “ImageNet: A large-scale hierarchical image database.” In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. June 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.
- [9] Krizhevsky, Alex and Geoffrey Hinton. “Learning multiple layers of features from tiny images.” In: (2009). URL: <http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf>.
- [10] Catherine Wah et al. *The Caltech-UCSD Birds-200-2011 Dataset*. July 2011. URL: <https://resolver.caltech.edu/CaltechAUTHORS:201111026-120541847> (visited on 02/16/2023).
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. URL: <https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html> (visited on 02/25/2023).
- [12] Alex Graves, Greg Wayne, and Ivo Danihelka. *Neural Turing Machines*. Dec. 10, 2014. DOI: 10.48550/arXiv.1410.5401. arXiv: 1410.5401[cs]. URL: <http://arxiv.org/abs/1410.5401> (visited on 02/21/2023).
- [13] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. Mar. 4, 2014. DOI: 10.48550/arXiv.1312.4400. arXiv: 1312.4400[cs]. URL: <http://arxiv.org/abs/1312.4400> (visited on 02/25/2023).
- [14] Gregory Koch. “Siamese neural networks for one-shot image recognition.” In: *ICML deep learning workshop*. 32nd International Conference on Machine Learning (ICML 2015). Vol. vol. 2. 2015. URL: <http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf>.
- [15] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. “Human-level concept learning through probabilistic program induction.” In: *Science* 350.6266 (Dec. 11, 2015), pp. 1332–1338. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aab3050. URL: <https://www.sciencemag.org/lookup/doi/10.1126/science.aab3050> (visited on 02/15/2023).
- [16] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556. arXiv: 1409.1556[cs]. URL: <http://arxiv.org/abs/1409.1556> (visited on 02/25/2023).

- [17] Sainbayar Sukhbaatar et al. “End-To-End Memory Networks.” In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: <https://proceedings.neurips.cc/paper/2015/hash/8fb21ee7a2207526da55a679f0332de2-Abstract.html> (visited on 02/21/2023).
- [18] Christian Szegedy et al. “Going Deeper With Convolutions.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9. URL: [https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2015/html/Szegedy\\_Going\\_Deeper\\_With\\_2015\\_CVPR\\_paper.html](https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html) (visited on 02/25/2023).
- [19] Kaiming He et al. “Deep Residual Learning for Image Recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2016/html/He\\_Deep\\_Residual\\_Learning\\_CVPR\\_2016\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html) (visited on 02/25/2023).
- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [21] Alexander Miller et al. *Key-Value Memory Networks for Directly Reading Documents*. Oct. 10, 2016. DOI: 10.48550/arXiv.1606.03126. arXiv: 1606.03126[cs]. URL: <http://arxiv.org/abs/1606.03126> (visited on 02/21/2023).
- [22] Deepak Pathak et al. “Context Encoders: Feature Learning by Inpainting.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2536–2544. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2016/html/Pathak\\_Context\\_Encoders\\_Feature\\_CVPR\\_2016\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2016/html/Pathak_Context_Encoders_Feature_CVPR_2016_paper.html) (visited on 02/27/2023).
- [23] Adam Santoro et al. “Meta-Learning with Memory-Augmented Neural Networks.” In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, June 11, 2016, pp. 1842–1850. URL: <https://proceedings.mlr.press/v48/santoro16.html> (visited on 02/21/2023).
- [24] Oriol Vinyals et al. “Matching Networks for One Shot Learning.” In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html> (visited on 02/15/2023).

- [25] Richard Zhang, Phillip Isola, and Alexei A. Efros. “Colorful Image Colorization.” In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 649–666. ISBN: 9783319464879. DOI: 10.1007/978-3-319-46487-9\_40.
- [26] Han Altae-Tran et al. “Low Data Drug Discovery with One-Shot Learning.” In: *ACS Central Science* 3.4 (Apr. 26, 2017), pp. 283–293. ISSN: 2374-7943, 2374-7951. DOI: 10.1021/acscentsci.6b00367. URL: <https://pubs.acs.org/doi/10.1021/acscentsci.6b00367> (visited on 02/17/2023).
- [27] Sergi Caelles et al. “One-Shot Video Object Segmentation.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 221–230. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2017/html/Caelles\\_One-Shot\\_Video\\_Object\\_CVPR\\_2017\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2017/html/Caelles_One-Shot_Video_Object_CVPR_2017_paper.html) (visited on 02/25/2023).
- [28] Yan Duan et al. “One-Shot Imitation Learning.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/ba3866600c3540f67c1e9575e213be0a-Abstract.html> (visited on 02/17/2023).
- [29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 17, 2017, pp. 1126–1135. URL: <https://proceedings.mlr.press/v70/finn17a.html> (visited on 02/15/2023).
- [30] Gao Huang et al. “Densely Connected Convolutional Networks.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, pp. 4700–4708. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2017/html/Huang\\_Densely\\_Connected\\_Convolutional\\_CVPR\\_2017\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html) (visited on 02/25/2023).
- [31] Jake Snell, Kevin Swersky, and Richard Zemel. “Prototypical Networks for Few-shot Learning.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html> (visited on 02/15/2023).

- [32] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. “Few-Shot Learning Through an Information Retrieval Lens.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/01e9565cecc4e989123f9620c1d09c09-Abstract.html> (visited on 02/17/2023).
- [33] Manasi Vartak et al. “A Meta-Learning Perspective on Cold-Start Recommendations for Items.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/51e6d6e679953c6311757004d8cbbba9-Abstract.html> (visited on 02/17/2023).
- [34] Ashish Vaswani et al. “Attention is All you Need.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html> (visited on 02/20/2023).
- [35] Sergey Zagoruyko and Nikos Komodakis. *Wide Residual Networks*. June 14, 2017. DOI: 10.48550/arXiv.1605.07146. arXiv: 1605.07146[cs]. URL: <http://arxiv.org/abs/1605.07146> (visited on 02/25/2023).
- [36] Sagie Benaim and Lior Wolf. “One-Shot Unsupervised Cross Domain Translation.” In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/062ddb6c727310e76b6200b7c71f63b5-Abstract.html> (visited on 02/21/2023).
- [37] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*. Mar. 20, 2018. DOI: 10.48550/arXiv.1803.07728. arXiv: 1803.07728[cs]. URL: <http://arxiv.org/abs/1803.07728> (visited on 02/15/2023).
- [38] Rohit Keshari et al. “Learning Structure and Strength of CNN Filters for Small Sample Size Training.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 9349–9358. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2018/html/Keshari\\_Learning\\_Structure\\_and\\_CVPR\\_2018\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2018/html/Keshari_Learning_Structure_and_CVPR_2018_paper.html) (visited on 02/25/2023).
- [39] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. “TADAM: Task dependent adaptive metric for improved few-shot learning.” In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/>

- paper/2018/hash/66808e327dc79d135ba18e051673d906-Abstract.html (visited on 02/20/2023).
- [40] Hang Qi, Matthew Brown, and David G. Lowe. “Low-Shot Learning With Imprinted Weights.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 5822–5830. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2018/html/Qi\\_Low-Shot\\_Learning\\_With\\_CVPR\\_2018\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2018/html/Qi_Low-Shot_Learning_With_CVPR_2018_paper.html) (visited on 02/21/2023).
- [41] Scott Reed et al. *Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions*. Feb. 28, 2018. DOI: 10.48550/arXiv.1710.10304. arXiv: 1710.10304[cs]. URL: <http://arxiv.org/abs/1710.10304> (visited on 02/24/2023).
- [42] Mengye Ren et al. *Meta-Learning for Semi-Supervised Few-Shot Classification*. Mar. 1, 2018. DOI: 10.48550/arXiv.1803.00676. arXiv: 1803.00676[cs,stat]. URL: <http://arxiv.org/abs/1803.00676> (visited on 02/15/2023).
- [43] Flood Sung et al. “Learning to Compare: Relation Network for Few-Shot Learning.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, pp. 1199–1208. URL: [https://openaccess.thecvf.com/content\\_cvpr\\_2018/html/Sung\\_Learning\\_to\\_Compare\\_CVPR\\_2018\\_paper.html](https://openaccess.thecvf.com/content_cvpr_2018/html/Sung_Learning_to_Compare_CVPR_2018_paper.html) (visited on 02/15/2023).
- [44] Athanasios Voulodimos et al. “Deep Learning for Computer Vision: A Brief Review.” In: *Computational Intelligence and Neuroscience* 2018 (Feb. 1, 2018), e7068349. ISSN: 1687-5265. DOI: 10.1155/2018/7068349. URL: <https://www.hindawi.com/journals/cin/2018/7068349/> (visited on 02/26/2023).
- [45] Yabin Zhang, Hui Tang, and Kui Jia. “Fine-Grained Visual Categorization using Meta-Learning Optimization with Sample Selection of Auxiliary Data.” In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, pp. 233–248. URL: [https://openaccess.thecvf.com/content\\_ECCV\\_2018/html/Yabin\\_Zhang\\_Fine-Grained\\_Visual\\_Categorization\\_ECCV\\_2018\\_paper.html](https://openaccess.thecvf.com/content_ECCV_2018/html/Yabin_Zhang_Fine-Grained_Visual_Categorization_ECCV_2018_paper.html) (visited on 02/21/2023).
- [46] Ekin D. Cubuk et al. “AutoAugment: Learning Augmentation Strategies From Data.” In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, pp. 113–123. URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/](https://openaccess.thecvf.com/content_CVPR_2019/html/)

- Cubuk\_AutoAugment\_Learning-Augmentation\_Strategies\_From\_Data\_CVPR\_2019\_paper.html (visited on 02/21/2023).
- [47] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. May 24, 2019.  
DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs].  
URL: <http://arxiv.org/abs/1810.04805> (visited on 02/26/2023).
- [48] Jonathan Gordon et al.  
*Meta-Learning Probabilistic Inference For Prediction*. Aug. 6, 2019.  
DOI: 10.48550/arXiv.1805.09921. arXiv: 1805.09921[cs,stat].  
URL: <http://arxiv.org/abs/1805.09921> (visited on 02/24/2023).
- [49] Kwonjoon Lee et al.  
“Meta-Learning With Differentiable Convex Optimization.” In:  
Proceedings of the IEEE/CVF Conference on Computer Vision and  
Pattern Recognition. 2019, pp. 10657–10665. URL:  
[https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Lee\\_Meta-Learning\\_With\\_Differentiable\\_Convex\\_Optimization\\_CVPR\\_2019\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2019/html/Lee_Meta-Learning_With_Differentiable_Convex_Optimization_CVPR_2019_paper.html) (visited on 02/15/2023).
- [50] Xinzhe Li et al.  
“Learning to Self-Train for Semi-Supervised Few-Shot Classification.”  
In: *Advances in Neural Information Processing Systems*. Vol. 32.  
Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/bf25356fd2a6e038f1a3a59c26687e80-Abstract.html>  
(visited on 03/16/2023).
- [51] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*.  
Jan. 4, 2019. DOI: 10.48550/arXiv.1711.05101.  
arXiv: 1711.05101[cs,math].  
URL: <http://arxiv.org/abs/1711.05101> (visited on 02/15/2023).
- [52] Jorge Pérez, Javier Marinković, and Pablo Barceló.  
*On the Turing Completeness of Modern Neural Network Architectures*.  
Jan. 10, 2019. DOI: 10.48550/arXiv.1901.03429.  
arXiv: 1901.03429[cs,stat].  
URL: <http://arxiv.org/abs/1901.03429> (visited on 02/27/2023).
- [53] Zhipeng Zhang and Houwen Peng.  
“Deeper and Wider Siamese Networks for Real-Time Visual Tracking.” In:  
Proceedings of the IEEE/CVF Conference on Computer Vision and  
Pattern Recognition. 2019, pp. 4591–4600.  
URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Zhang\\_Deep\\_and\\_Wider\\_Siamese\\_Networks\\_for\\_Real-Time\\_Visual\\_Tracking\\_CVPR\\_2019\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2019/html/Zhang_Deep_and_Wider_Siamese_Networks_for_Real-Time_Visual_Tracking_CVPR_2019_paper.html) (visited on 02/17/2023).

- [54] Tom Brown et al. “Language Models are Few-Shot Learners.” In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.  
URL: <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html> (visited on 02/19/2023).
- [55] Ting Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations.” In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 1597–1607.  
URL: <https://proceedings.mlr.press/v119/chen20j.html> (visited on 02/15/2023).
- [56] Wei-Yu Chen et al. *A Closer Look at Few-shot Classification*. Jan. 12, 2020. DOI: 10.48550/arXiv.1904.04232. arXiv: 1904.04232[cs].  
URL: <http://arxiv.org/abs/1904.04232> (visited on 02/15/2023).
- [57] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. *On the Relationship between Self-Attention and Convolutional Layers*. Jan. 10, 2020. DOI: 10.48550/arXiv.1911.03584. arXiv: 1911.03584[cs,stat].  
URL: <http://arxiv.org/abs/1911.03584> (visited on 02/27/2023).
- [58] Guneet S. Dhillon et al. *A Baseline for Few-Shot Image Classification*. Oct. 21, 2020. DOI: 10.48550/arXiv.1909.02729. arXiv: 1909.02729[cs,stat].  
URL: <http://arxiv.org/abs/1909.02729> (visited on 02/15/2023).
- [59] Carl Doersch, Ankush Gupta, and Andrew Zisserman. “CrossTransformers: spatially-aware few-shot transfer.” In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 21981–21993.  
URL: <https://proceedings.neurips.cc/paper/2020/hash/fa28c6cdf8dd6f41a657c3d7caa5c709-Abstract.html> (visited on 02/15/2023).
- [60] Ryuichiro Hataya et al. “Faster AutoAugment: Learning Augmentation Strategies Using Backpropagation.” In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 1–16. ISBN: 9783030585952. DOI: 10.1007/978-3-030-58595-2\_1.
- [61] Kaiming He et al. “Momentum Contrast for Unsupervised Visual Representation Learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.  
URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2020/html/He\\_](https://openaccess.thecvf.com/content_CVPR_2020/html/He_)



- Momentum\_Contrast\_for\_Unsupervised\_Visual\_Representation\_Learning\_CVPR\_2020\_paper.html (visited on 02/27/2023).
- [62] Ishan Misra and Laurens van der Maaten. “Self-Supervised Learning of Pretext-Invariant Representations.” In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 6707–6717. URL: [https://openaccess.thecvf.com/content\\_CVPR\\_2020/html/Misra\\_Self-Supervised\\_Learning\\_of\\_Pretext-Invariant\\_Representations\\_CVPR\\_2020\\_paper.html](https://openaccess.thecvf.com/content_CVPR_2020/html/Misra_Self-Supervised_Learning_of_Pretext-Invariant_Representations_CVPR_2020_paper.html) (visited on 02/27/2023).
- [63] Niall O’Mahony et al. “Deep Learning vs. Traditional Computer Vision.” In: *Advances in Computer Vision*. Ed. by Kohei Arai and Supriya Kapoor. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020, pp. 128–144. ISBN: 9783030177959. DOI: 10.1007/978-3-030-17795-9\_10.
- [64] Pau Rodríguez et al. “Embedding Propagation: Smoother Manifold for Few-Shot Classification.” In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 121–138. ISBN: 9783030585747. DOI: 10.1007/978-3-030-58574-7\_8.
- [65] Eleni Triantafillou et al. *Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples*. Apr. 8, 2020. DOI: 10.48550/arXiv.1903.03096. arXiv: 1903.03096[cs,stat]. URL: <http://arxiv.org/abs/1903.03096> (visited on 02/15/2023).
- [66] Yaqing Wang et al. “Generalizing from a Few Examples: A Survey on Few-shot Learning.” In: *ACM Computing Surveys* 53.3 (June 12, 2020), 63:1–63:34. ISSN: 0360-0300. DOI: 10.1145/3386252. URL: <https://doi.org/10.1145/3386252> (visited on 02/15/2023).
- [67] Martin J. Willeminck et al. “Preparing Medical Imaging Data for Machine Learning.” In: *Radiology* 295.1 (Apr. 2020), pp. 4–15. ISSN: 0033-8419, 1527-1315. DOI: 10.1148/radiol.2020192224. URL: <http://pubs.rsna.org/doi/10.1148/radiol.2020192224> (visited on 02/18/2023).
- [68] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing.” In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

- URL: <https://aclanthology.org/2020.emnlp-demos.6> (visited on 02/26/2023).
- [69] Mohamed Afham et al. *Rich Semantics Improve Few-shot Learning*. Nov. 12, 2021. DOI: 10.48550/arXiv.2104.12709. arXiv: 2104.12709[cs]. URL: <http://arxiv.org/abs/2104.12709> (visited on 03/16/2023).
- [70] Yutong Bai et al. “Are Transformers more robust than CNNs?” In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 26831–26843. URL: <https://proceedings.neurips.cc/paper/2021/hash/e19347e1c3ca0c0b97de5fb3b690855a-Abstract.html> (visited on 02/25/2023).
- [71] Myriam Bontonou et al. “Few-Shot Decoding of Brain Activation Maps.” In: *2021 29th European Signal Processing Conference (EUSIPCO)*. 2021 29th European Signal Processing Conference (EUSIPCO). ISSN: 2076-1465. Aug. 2021, pp. 1326–1330. DOI: 10.23919/EUSIPC054536.2021.9616158.
- [72] Mathilde Caron. “Self-supervised learning of deep visual representations.” PhD thesis. Université Grenoble Alpes [2020-....], Dec. 9, 2021. URL: <https://theses.hal.science/tel-03675254> (visited on 02/27/2023).
- [73] Mathilde Caron et al. “Emerging Properties in Self-Supervised Vision Transformers.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660. URL: [https://openaccess.thecvf.com/content/ICCV2021/html/Caron\\_Emerging\\_Properties\\_in\\_Self-Supervised\\_Vision\\_Transformers\\_ICCV\\_2021\\_paper.html](https://openaccess.thecvf.com/content/ICCV2021/html/Caron_Emerging_Properties_in_Self-Supervised_Vision_Transformers_ICCV_2021_paper.html) (visited on 02/15/2023).
- [74] Da Chen et al. “Self-Supervised Learning for Few-Shot Image Classification.” In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). ISSN: 2379-190X. June 2021, pp. 1745–1749. DOI: 10.1109/ICASSP39728.2021.9413783.
- [75] Yinbo Chen et al. “Meta-Baseline: Exploring Simple Meta-Learning for Few-Shot Learning.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9062–9071. URL: [https://openaccess.thecvf.com/content/ICCV2021/html/Chen\\_Meta-Baseline\\_Exploring\\_Simple\\_Meta-Learning\\_for\\_Few-Shot\\_Learning\\_ICCV\\_2021\\_paper.html](https://openaccess.thecvf.com/content/ICCV2021/html/Chen_Meta-Baseline_Exploring_Simple_Meta-Learning_for_Few-Shot_Learning_ICCV_2021_paper.html) (visited on 02/15/2023).

- [76] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021.  
DOI: 10.48550/arXiv.2010.11929. arXiv: 2010.11929[cs].  
URL: <http://arxiv.org/abs/2010.11929> (visited on 02/15/2023).
- [77] Yuqing Hu, Vincent Gripon, and Stéphane Pateux. “Leveraging the Feature Distribution in Transfer-Based Few-Shot Learning.”  
In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Ed. by Igor Farkaš et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 487–499.  
ISBN: 9783030863401. DOI: 10.1007/978-3-030-86340-1\_39.
- [78] Kai Huang et al.  
“Pseudo-Loss Confidence Metric for Semi-Supervised Few-Shot Learning.”  
In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021, pp. 8671–8680. URL: [https://openaccess.thecvf.com/content/ICCV2021/html/Huang\\_Pseudo-Loss\\_Confidence\\_Metric\\_for\\_Semi-Supervised\\_Few-Shot\\_Learning\\_ICCV\\_2021\\_paper.html](https://openaccess.thecvf.com/content/ICCV2021/html/Huang_Pseudo-Loss_Confidence_Metric_for_Semi-Supervised_Few-Shot_Learning_ICCV_2021_paper.html) (visited on 03/16/2023).
- [79] Muhammad Muzammal Naseer et al.  
“Intriguing Properties of Vision Transformers.”  
In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 23296–23308.  
URL: <https://proceedings.neurips.cc/paper/2021/hash/c404a5adbf90e09631678b13b05d9d7a-Abstract.html> (visited on 02/25/2023).
- [80] Peyman Bateni et al.  
“Enhancing Few-Shot Image Classification With Unlabelled Examples.” In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022, pp. 2796–2805.  
URL: [https://openaccess.thecvf.com/content/WACV2022/html/Bateni\\_Enhancing\\_Few-Shot\\_Image\\_Classification\\_With\\_Unlabelled\\_Examples\\_WACV\\_2022\\_paper.html](https://openaccess.thecvf.com/content/WACV2022/html/Bateni_Enhancing_Few-Shot_Image_Classification_With_Unlabelled_Examples_WACV_2022_paper.html) (visited on 03/16/2023).
- [81] Ananth Reddy Bhimireddy et al. *Few-Shot Transfer Learning to improve Chest X-Ray pathology detection using limited triplets*. Apr. 16, 2022.  
DOI: 10.48550/arXiv.2204.07824. arXiv: 2204.07824[cs,eess].  
URL: <http://arxiv.org/abs/2204.07824> (visited on 02/18/2023).
- [82] Rishi Bommasani et al.  
*On the Opportunities and Risks of Foundation Models*. July 12, 2022.  
DOI: 10.48550/arXiv.2108.07258. arXiv: 2108.07258[cs].  
URL: <http://arxiv.org/abs/2108.07258> (visited on 02/15/2023).

- [83] Aleksandr Ermolov et al. “Hyperbolic Vision Transformers: Combining Improvements in Metric Learning.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7409–7419.  
URL: [https://openaccess.thecvf.com/content/CVPR2022/html/Ermolov\\_Hyperbolic\\_Vision\\_Transformers\\_Combining\\_Improvements\\_in\\_Metric\\_Learning\\_CVPR\\_2022\\_paper.html](https://openaccess.thecvf.com/content/CVPR2022/html/Ermolov_Hyperbolic_Vision_Transformers_Combining_Improvements_in_Metric_Learning_CVPR_2022_paper.html) (visited on 03/01/2023).
- [84] Timothy Hospedales et al. “Meta-Learning in Neural Networks: A Survey.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9 (Sept. 2022), pp. 5149–5169. ISSN: 1939-3539.  
DOI: 10.1109/TPAMI.2021.3079209.
- [85] Shell Xu Hu et al. “Pushing the Limits of Simple Pipelines for Few-Shot Learning: External Data and Fine-Tuning Make a Difference.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9068–9077.  
URL: [https://openaccess.thecvf.com/content/CVPR2022/html/Hu\\_Pushing\\_the\\_Limits\\_of\\_Simple\\_Pipelines\\_for\\_Few-Shot\\_Learning\\_External\\_CVPR\\_2022\\_paper.html](https://openaccess.thecvf.com/content/CVPR2022/html/Hu_Pushing_the_Limits_of_Simple_Pipelines_for_Few-Shot_Learning_External_CVPR_2022_paper.html) (visited on 02/15/2023).
- [86] Salman Khan et al. “Transformers in Vision: A Survey.” In: *ACM Computing Surveys* 54.10 (Sept. 13, 2022), 200:1–200:41. ISSN: 0360-0300. DOI: 10.1145/3505244.  
URL: <https://doi.org/10.1145/3505244> (visited on 02/26/2023).
- [87] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue*. OpenAI. Nov. 30, 2022.  
URL: <https://openai.com/blog/chatgpt/> (visited on 02/18/2023).
- [88] OpenAI. *DALL·E 2*. OpenAI. 2022.  
URL: <https://openai.com/dall-e-2/> (visited on 02/18/2023).
- [89] Sayak Paul and Pin-Yu Chen. “Vision Transformers Are Robust Learners.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.2 (June 28, 2022), pp. 2071–2081. ISSN: 2374-3468.  
DOI: 10.1609/aaai.v36i2.20103.  
URL: <https://ojs.aaai.org/index.php/AAAI/article/view/20103> (visited on 02/25/2023).
- [90] Sachin Ravi and Hugo Larochelle. “Optimization as a Model for Few-Shot Learning.” In: *International Conference on Learning Representations*. July 21, 2022. URL: <https://openreview.net/forum?id=rJY0-Kc11> (visited on 02/15/2023).
- [91] Yu Zhang and Qiang Yang. “A Survey on Multi-Task Learning.” In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (Dec. 2022), pp. 5586–5609. ISSN: 1558-2191. DOI: 10.1109/TKDE.2021.3070203.

- [92] Kai Han et al. “A Survey on Vision Transformer.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (Jan. 2023), pp. 87–110. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3152247.
- [93] *Poincaré disk model*. In: *Wikipedia*. Page Version ID: 1142291709. Mar. 1, 2023. URL: [https://en.wikipedia.org/w/index.php?title=Poincar%C3%A9\\_disk\\_model&oldid=1142291709](https://en.wikipedia.org/w/index.php?title=Poincar%C3%A9_disk_model&oldid=1142291709) (visited on 03/03/2023).
- [94] Yisheng Song et al. “A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities.” In: *ACM Computing Surveys* (Feb. 4, 2023). Just Accepted. ISSN: 0360-0300. DOI: 10.1145/3582688. URL: <https://doi.org/10.1145/3582688> (visited on 02/16/2023).