# Deep Learning Project Final Report

## CONCORDIA UNIVERSITY

**Sepehr Ghamari**
**Arman Bakhtiari**

April 25, 2024

**Abstract:** This report delves into the application of deep learning techniques for image classification under the constraints of limited data availability, focusing on two distinct challenges: Challenge 1, where pre-trained models cannot be utilized, and Challenge 2, where the use of pre-trained models is permitted. The primary focus is on State-Of-The-Technology (SOT) in few-shot learning (FSL), particularly under the conditions of Challenge 1. The analysis highlights that SOT methods outperform other approaches in Challenge 1, achieving higher accuracy without the benefit of pre-trained models. This underscores the effectiveness of SOT in adapting deep learning techniques to scenarios where data is scarce and conventional model training is restricted.

Draft

## 1. Introduction

Image classification, a fundamental task in computer vision, has experienced a significant leap forward with the integration of deep learning technologies. At the heart of this advancement are extensive datasets such as CIFAR and ImageNet, which are crucial for training deep neural networks (DNNs). CIFAR, with its tens of thousands of labeled images across various categories, provides a solid foundation for developing and testing algorithms. In contrast, ImageNet, with over a million labeled images, has been instrumental in pushing the boundaries of what deep learning can achieve, setting new benchmarks for accuracy in image classification.

The development of pre-trained models represents another cornerstone in the evolution of image classification. Models such as ResNet, Inception, and VGG have been trained on large datasets like ImageNet and are widely used as starting points for many image recognition tasks. By utilizing these pre-trained models, practitioners can employ transfer learning to apply the knowledge gained from one dataset to another. This technique is particularly valuable as it allows for the adaptation of models to new tasks with relatively minor modifications and significantly less data than training from scratch.

Despite these advancements, the challenge of data scarcity remains a major hurdle in specific domains. In fields such as medical imaging or rare species identification, acquiring large volumes of annotated data can be difficult, costly, and time-consuming. This scarcity of data can impede the development of highly accurate models, as deep learning typically relies on large datasets to achieve optimal performance.

Addressing the problem of image classification with limited data is therefore a pivotal focus of this project. The goal is to explore innovative deep learning techniques that are tailored for scenarios where data is not abundantly available. This might include methods such as few-shot learning, synthetic data generation, or advanced data augmentation techniques that enhance the size and variability of training datasets without the need for extensive new data collection. By tackling the issue of limited data, this project aims to broaden the applicability of deep learning in image classification to a wider array of fields, making this powerful technology accessible even when resources are constrained. In this project we mainly put our focus to tackle the problem using few-shot learning techniques.

## 2. Literature Review

Few-shot learning (FSL) is a methodological framework designed to teach machines to understand and generalize from a very limited set of examples—typically much fewer than traditional machine learning models would require. The essence of FSL is to mimic human learning capabilities, which allow for recognizing patterns and making decisions based on minimal exposure to new data. This approach is particularly crucial in domains where data collection is expensive or privacy issues limit the size of datasets.

### 2.1. *LEARNING FROM FEW EXAMPLES: A SUMMARY OF APPROACHES TO FEW-SHOT LEARNING*

Archit Parnami and Minwoo Lee Parnami and Lee, 2022 discuss the various strategies for tackling FSL, dividing the solutions to “Meta-learning” and “non-Meta-learning” approaches (figure1). They emphasize on meta-learning as a central technique. Meta-learning, or ”learning to learn,” trains a model on a variety of learning tasks, allowing it to apply its refined predictive power to new, unseen tasks after only a few examples. This can involve different sub-approaches like metric-based learning, which focuses on distance functions between examples; optimization-based learning, which tweaks the learning algorithm itself to be more efficient with fewer samples; and model-based approaches, which might adjust the model architecture to be more amenable to quick learning from small datasets.

Hybrid methods also play a significant role in advancing FSL by combining elements of these approaches or integrating concepts from related fields like transfer learning, which reuses a pre-trained model on a new, related problem. Such integrations help in addressing the inherent challenges of FSL, such as overfitting and the robustness of models when exposed to novel data under constrained conditions.

Based on the paper we reviewed, we explored several of the outlined models to address our initial challenge. Below is a summary of some of the models we employed.

#### 2.1.1. *MAML*

The fundamental concept behind Model-Agnostic Meta-Learning (MAML) Finn et al., 2017 is to establish effective initial parameters so that new tasks can quickly adapt using one or more steps of gradient descent, even with limited data. These starting parameters are meta-trained across various tasks. Contrary to the LSTM meta-learner, which utilizes two distinct models—a meta-learner and a task-specific learner—MAML employs a unified model. This model uses its parameters as a baseline to facilitate rapid optimization and adjustment to new tasks, achieving the best task-specific parameters.

#### 2.1.2. *ProtoNet*

In Prototypical Networks Snell et al., 2017, a convolutional neural network (CNN) with four layers serves as the embedding function, denoted as $g_{\theta_1}$. The model establishes a class prototype by averaging the embedding vectors of support images that are classified under that particular class:

$$v_c = \frac{1}{|S_c|} \sum_{(x_k, y_k) \in S_c} g_{\theta_1}(x_k) \tag{1}$$

The degree of similarity is determined by the squared Euclidean distance between the embedding of the query image and the class prototypes. The likelihood of each class for the query image is then computed using a softmax function over these negative distances:

$$P(y = c|\hat{x}) = \frac{e^{-d(g_{\theta_1}(\hat{x}), v_c)}}{\sum_{\hat{c} \in C} e^{-d(g_{\theta_1}(\hat{x}), v_{\hat{c}})}} \tag{2}$$

Finally, the loss function, $L$, is calculated as the negative log-likelihood of the correct class label:

$$L(\theta_1) = -\log P_{\theta_1}(y = c|\hat{x}) \tag{3}$$

#### 2.1.3. *CAN*

The Cross Attention Network (CAN) introduces a Cross Attention Module (CAM) Hou et al., 2019 to emphasize the target object regions within the query sample feature, thus yielding more discriminative features. This module achieves attention by generating

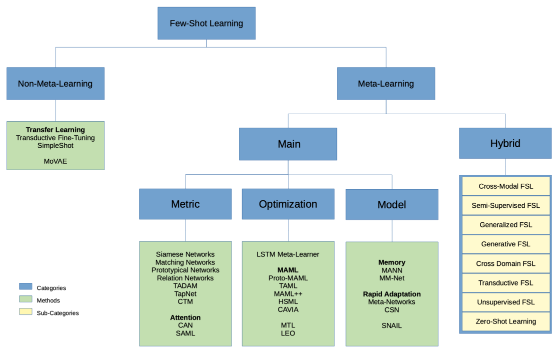

Figure 1: Approaches to FSL are categorized into Meta-Learning-based FSL and Non-Meta-Learning-based FSL Parnami and Lee, 2022.

cross attention maps for each pair of class features and query sample features. By employing correlation estimation and meta fusion, CAM can effectively highlight the target object in test samples, which is particularly beneficial for classes represented by few examples.

Additionally, CAN incorporates a transductive inference algorithm to mitigate the low-data problem. This algorithm uses the entire unlabeled query set to augment the support set iteratively, which allows the class features to become more representative as they are refined with additional pseudo-labeled query samples. The framework of CAN encompasses an embedding module, CAM, and a classification module, working in tandem to ensure that the relevant features of the target objects are accentuated for classification.

#### 2.1.4. *AdaBoost + ResNet*

AdaBoost, which stands for Adaptive Boosting, is a powerful ensemble learning technique that combines multiple weak classifiers to form a strong classifier. In few-shot learning, AdaBoost could be used to iteratively refine the model's performance on a small set of training data Wang et al., 2020. By focusing on the hardest to classify instances, AdaBoost can potentially help to generalize better from few examples by constructing a robust boundary between classes [maghale adaboost]. In this project we used AdaBoost as a classifier on top of a ResNet18 as a feature extractor.

### 2.2. *The Self-Optimal-Transport Feature Transform*

To address the few-shot learning challenges, this study integrates the Self-Optimal-Transport (SOT) feature transform, developed by Daniel Shalam and Simon Korman Shalam and Korman, 2022, which refines feature representation by leveraging the geometry of data distribution. The SOT enhances features of a data instance, facilitating effective matching or grouping tasks by encoding a rich representation of high-order relations between instance features, capturing both direct original similarity and their third-party agreements regarding similarity to other features in the set. Our methodology aims to demonstrate the practical advantages of SOT in improving the generalization capabilities of few-shot models, detailing its theoretical foundation and integration into our few-shot learning framework, with specifics on experimental setup designed to evaluate its impact rigorously.

#### 2.2.1. *Theoretical Basis*

The Self-Optimal-Transport (SOT) feature transform is fundamentally designed to enhance the capabilities of machine learning models by upgrading the feature set of each data instance. This enhancement facilitates superior performance in matching and grouping related tasks by encoding a rich representation of high-order relationships between the features of an instance. The concept of SOT is rooted in optimal transport theory, which provides a principled approach to compare probability distributions or, in this context, feature distributions. Optimal transport seeks to find the most efficient ”transport plan” that morphs one distribution into another while minimizing a certain cost function, typically related to distances between distribution elements.

**Input and Similarity Matrix**: The process begins with an input set of $n$ features, each of dimension $d$. These are arranged in a matrix $V$ of size $n \times d$. A pairwise cosine similarity matrix $S$ is then computed from $V$, where each element $s_{ij}$ represents the cosine similarity between features $i$ and $j$, calculated as follows:

$$s_{ij} = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|}$$

Figure 2 illustrates the initial setup where the input features and the resulting similarity matrix are depicted, emphasizing the transition from individual feature vectors to a structured similarity representation.

**Transport-Plan Matrix via Sinkhorn Iterations**: Utilizing the similarity matrix $S$, the SOT employs the Sinkhorn algorithm to compute the transport-plan matrix $W$. This algorithm iteratively adjusts the matrix to satisfy the doubly stochastic property, where each row and column sums to one. The mathematical expression for this iterative normalization is:

$$W^{(t+1)} = diag(u) \cdot e^{-\lambda C} \cdot diag(v)$$

where $u$ and $v$ are vectors adjusted at each iteration, $C$ is the cost matrix typically calculated as $C = -S$, and $\lambda$ is a regularization parameter. Figure 3 showcases the high-level design of networks that operate on sets of inputs, comparing generic designs to those that incorporate the SOT transform, thereby detailing how the extracted features undergo joint processing by the transform.

**Optimization Problem:**: The optimization aims to solve the entropic regularized optimal transport problem defined as follows:

$$\min_{W \in \Pi} \langle W, C \rangle - \frac{1}{\lambda} H(W)$$

where $H(W) = -\sum_{ij} W_{ij} \log W_{ij}$ is the entropy term, which encourages distribution smoothness and $\lambda$ is the regularization parameter that controls the trade-off between the fidelity of the match and the smoothness of the transport plan. The Sinkhorn algorithm operationalizes this by iteratively adjusting $W$ to satisfy the doubly stochastic conditions, effectively balancing the mass distribution across the transformed features.

This optimization is executed by alternately scaling rows and columns of the matrix $K$, derived from the cost matrix $C$, using vectors $u$ and $v$:

$$W = diag(u) \cdot K \cdot diag(v)$$

where $K = e^{-\lambda C}$, ensuring $W$ remains a doubly stochastic matrix, which means each of its rows and columns sums to one. This iterative adjustment converges to the optimal $W$ that minimizes the cost while adhering to the constraints.

### 2.2.2. *Implementation Details*

The Self-Optimal-Transport (SOT) feature transform was implemented and tested across various machine-learning tasks to assess its efficacy. Specific details include:

**Datasets:** For few-shot classification, the experiments primarily utilized the following benchmark datasets:

- **MiniImagenet**: A popular subset of ImageNet designed for few-shot learning tasks, which includes 100 classes with 600 images each.
- **CIFAR-FS**: A version of CIFAR-100 adapted for few-shot learning that includes 100 classes.
- **CUB**: Focuses on bird species, used for more detailed classification challenges in few-shot learning.

**Pre-training and Fine-tuning:**

- Baseline networks like ProtoNet were adapted and pre-trained.
- Fine-tuning was performed on backbone architectures such as resnet-12 and WRN-28-10, tailored to integrate SOT effectively.

**Hyper-parameters:** Two primary hyperparameters were crucial for optimizing the performance and efficiency of the SOT feature transform:

1. **Number of Sinkhorn Iterations**: Tuned through preliminary experiments to balance between computational efficiency and the accuracy of the transport matrix. Typically, 10 to 20 iterations are sufficient for convergence, though this varies by the complexity and size of the dataset. The number of Sinkhorn iterations for computing the optimal transport plan was fixed at 10.
2. **Entropy Regularization Parameter** $\lambda$: Chosen via cross-validation, where values ranged from 0.01 for finer detail in transport plans, up to 10 for more generalized mappings. The parameter $\lambda$ controls the trade-off between precise

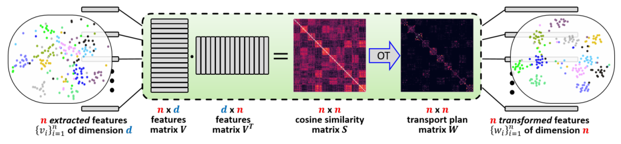
$n$ extracted features $\{v_i\}_{i=1}^n$ of dimension $d$
$n$ x $d$ features matrix $V$
$d$ x $n$ features matrix $V^T$
$n$ x $n$ cosine similarity matrix $S$
OT
$n$ x $n$ transport plan matrix $W$
$n$ transformed features $\{w_i\}_{i=1}^n$ of dimension $n$

Figure 2: The SOT processes $n$ $d$-dimensional features, each visualized as a colored point in the input embedding space, where color indicates the class label. Features are normalized and organized into an $n \times d$ matrix for computing a $n \times n$ cosine similarity matrix $S$. A transport-plan matrix $W$ is then derived from $S$ using several Sinkhorn iterations. The rows of $W$ represent the transformed output features, which are re-embedded to enhance performance in downstream grouping and matching tasks, such as linear classification or clustering algorithms.

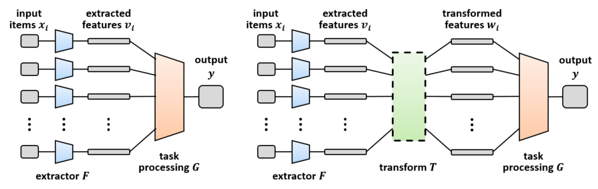

Figure 3: Generic network designs for processing sets of inputs cover architectures suitable for tasks like few-shot classification and clustering. On the left, a typical network processes each input item independently through a common feature extractor $F$, followed by a downstream task module $G$. On the right, a more advanced design involves joint processing of extracted features by a transform $T$, such as our SOT transform or other attention mechanisms. The high-level design of this process is depicted in Figure 1 within the 'green' module.

matching and more diffuse, probabilistic mappings. The entropy regularization parameter λ was set to 0.1.

**Performance Evaluation:** The effectiveness of the SOT-enhanced models was rigorously compared against baseline models and other state-of-the-art few-shot learning methods. The focus was on improvements in classification accuracy and the ability of the SOT to enhance feature discriminability.

## Methodology

For all methods, the CIFAR-10 dataset was employed, comprising 60,000 32x32 color images distributed across 10 classes. During the training phase, models were trained using only two classes at a time, with 25 samples per class. For evaluation, 200 samples per class from the CIFAR-10 test dataset, which were not exposed to the models during training, were utilized.

### *2.3. Challenge 1*

#### *2.3.1. MAML*

**Data Handling:** As MAML is a meta-learning method, each epoch involves considering 5 tasks. In each task, 5 samples per class are used.

**Backbone:** This method employs a simple CNN (referred to as The Net in the TestBed) and a non-pretrained ResNet18 with a dropout layer as the backbone.

**Optimizer:** SGD

**Hyper-parameters:** learning-rate = 0.001. Three learning rates were tested: 0.1, 0.01, and 0.001. The best results were obtained with a learning rate of 0.001.

#### *2.3.2. ProtoNet*

**Data Handling:** A 2-way 5-shot setup is employed. the model was run on 5 tasks, 3 supports, and 2 queries.

**Backbone:** This method employs a simple CNN (referred to as The Net in the TestBed) and a non-pretrained ResNet18 with a dropout layer as the backbone.

**Optimizer:** Adam

**Hyper-parameters:** learning-rate = 0.001. Three learning rates were tested: 0.1, 0.01, and 0.001. The best results were obtained with a learning rate of 0.001.

#### *2.3.3. CAN*

**Data Handling:** 50 samples for training (25 per class), 400 samples for testing(200 per class)

**Backbone:** Simple CNN + CrossAttention Layer + classifier

**Optimizer:** Adam

**Hyper-parameters:** learning-rate = 0.001. Three learning rates were tested: 0.1, 0.01, and 0.001. The best results were obtained with a learning rate of 0.001.

#### *2.3.4. AdaBoost*

**Data Handling:** 50 samples for training (25 per class), 400 samples for testing(200 per class)

**Backbone:** This method uses a basic CNN (referred to as ”The Net” in the TestBed), a non-pretrained ResNet18 with a dropout layer as its backbone, and a pre-trained ResNet50. These three models were tested separately.

**Weak Learner:** Decision Tree

**Hyper-parameters:**n-estimators=50,learning-rate=1.0, random-state=0. Three learning rates were tested: 1, 0.1, and 0.01. The best results were obtained with a learning rate of 1.

#### *2.3.5. SOT*

In adapting the Self-Optimal-Transport (SOT) feature transform to specific project requirements, several modifications were made to the framework originally outlined in the referenced paper. These changes pertain to the choice of datasets, pre-training and fine-tuning approaches, and performance evaluation methodology.

**Datasets:** While the original study employed datasets like MiniImagenet, CIFAR-FS, and CUB for few-shot classification, this project made use of the CIFAR-10 dataset. It is also worth noting that the paper utilized a 5-way-1-shot and 5-way-5-shot setup, whereas this challenge implemented a 2-way-5-shot setup. Also for the training time 20 queries were employed. So in the first part of this project, we tried to convert our raw dataset, to a few-shot-compatible version of CIFAR10.

**Backbone:** Similar to the original paper, two different backbones, ResNet-12 and WRN-28-10, were used from scratch. Notably, a DropBlock was implemented in the ResNet block to enhance the model's generalization.

**Method:** In this project, two different methods were tested: ProtoNet and PTMap. Both utilized SOT as the feature transform block.

**Hyper-parameters:** n-Sinkhorn-iterations = 10, $\lambda$ = 0.1, learning-rate = 0.0002 Four learning rates were tested: 0.01, 0.001, 0.0001, and 0.0002. The best results were obtained with a learning rate of 0.001.

**Performance Evaluation (train and test):** In contrast to the less explicitly defined evaluation parameters in the original study, this project implemented a structured 2-way 5-shot 20-query evaluation setup. The model was trained and evaluated across five different seeds, with the final results reported as the mean and standard deviation of the performances across all seeds. Additionally, four different tests were conducted: both the ProtoNet and PTMap methods were tested using ResNet12 and WRN-28-10 backbones.

### *2.4. Challenge 2*

For the second challenge, we leveraged the flexibility provided by the use of pre-existing models. Specifically, we integrated and analyzed SOT that had previously shown promise during the first challenge. We downloaded and utilized the weights from the publicly available trained model of Daniel Shalam and Simon Korman Shalam and Korman, 2022. Along with that we deployed ResNet and trained an AdaBoost, and a simple linear classifier on top of that separately. The common thread across these models was the implementation of a pre-trained ResNet, sourced from the ImageNet database, which we utilized as a feature extraction mechanism. Subsequently, we focused on

Tabla 1: Performance and Training Time without Pre-training

| Method | Backbone | Test Accuracy | GPU Time |
|---|---|---|---|
| Simple CNN | CNN | 65.81 ± 5.87 | 1.98 ± 0.45 |
| MAML | CNN | 49 ± 0 | 14.15 ± 0.78 |
| MAML | ResNet18 | 56 ± 1.0 | 18.76 ± 0.78 |
| ProtoNet | CNN | 66.25 ± 7.40 | 25.10 ± 0.78 |
| ProtoNet | ResNet18 | 67.5 ± 7.5 | 401.42 ± 8.25 |
| CAN | CNN | 66 ± 12 | 1.82 ± 0.42 |
| ProtoSOT | Resnet12 | **76 ± 0.3** | 7.37 ± 0.31 |
| ProtoSOT | WRN | 68 ± 0.4 | 28.11 ± 0.11 |
| PTMapSOT | WRN | 70 ± 0.4 | 39.22 ± 0.62 |

Tabla 2: Performance and Training Time with Pre-training

| Method | Backbone | Test Accuracy | GPU Time |
|---|---|---|---|
| - | Resnet18 | 92.5 ± 5.26 | 0.13 ± 0.01 |
| - | ResNet50 | **95.75 ± 3.58** | 0.22 ± 0.01 |
| AdaBoost | ResNet18 | 86.75 ± 6.06 | 0.18 ± 0.01 |
| AdaBoost | ResNet50 | 87.44 ± 4.25 | 0.4 ± 0.22 |
| ProtoSOT | ResNet12 | 81 ± 0.5 | 0.12 ± 0.3 |

training each model's classifier using these extracted features. We delved into a comparative analysis between the ResNet18 and ResNet50 versions to understand the impact of model complexity on our few-shot learning task. Additionally, we rigorously measured and documented the training duration for each model to enable a comprehensive evaluation. This temporal analysis was critical, as it provided us with a metric to benchmark against the performance of an AlexNet-based model employed within the established testbed. Through this methodical approach, we aimed to not only achieve high accuracy in classification but also to assess the efficiency and scalability of the models in practice.

## Result

### *2.5. Challenge 1*

In this section, we present the outcomes of applying various analytical methods to our limited dataset. As detailed in earlier sections, different methods with specific setups were utilized to assess their performance and suitability for our data constraints. The findings from these experiments are systematically tabulated in Table 1, which serves as a comprehensive overview of the results obtained. Additionally, we have documented the training times for each method to provide a basis for comparison in terms of computational efficiency. This information is crucial for understanding the trade-offs between method effectiveness and resource utilization.

### *2.6. Challenge 2*

Similar to what we have done in the previous section, here we present the results on achieved accuracies and GPU time using the pre-trained models. The outcomes are shown in the Table 2.

## Conclusion

### *2.7. Challenge 1*

The highest accuracy is achieved by the ProtoSOT with Resnet12 backbone at 76 ± 0.3, which suggests that this combination is most effective among the tested methods for the CIFAR-10 dataset with limited samples. The lowest accuracy is recorded by MAML with a simple CNN backbone at 49 ± 0. The GPU time indicates the computational efficiency of the model during training. The least amount of time is taken by the Simple CNN at 1.98 ± 0.45, and the most is by ProtoNet with ResNet18 at 401.42 ± 8.25. For the few-sample learning task on CIFAR-10 with only 50 samples, the ProtoSOT with a Resnet12 backbone seems to be the most effective approach, achieving the highest accuracy with a fast training time. This suggests that the model is able to extract meaningful features from a very limited dataset and generalize well to a larger test set. On the other hand, methods like MAML with a simple CNN backbone might not be suitable for such tasks as they show poor performance in both accuracy and runtime.

### *2.8. Challenge 2*

The data in Table 2 clearly show that employing pretrained models substantially boosts accuracy levels. Although the linear classifier is quite basic, the combination of ResNet50 as a feature extractor with a linear classifier on top achieves the highest accuracy. Additionally, it is evident that using more complex backbone models like ResNet50 results in higher accuracies. The key point here is to deploy them just as a feature extractor and not train them on the data to avoid overfitting.

## References

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks.

Hou, R., Chang, H., Ma, B., Shan, S., & Chen, X. (2019). Cross attention network for few-shot classification.

Parnami, A., & Lee, M. (2022). Learning from few examples: A summary of approaches to few-shot learning.

Shalam, D., & Korman, S. (2022). The self-optimal-transport feature transform.

Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical networks for few-shot learning.

Wang, W., Zhang, L., Zhang, M., & Wang, Z. (2020). Few shot learning for multi-class classification based on nested ensemble dsvm. *Ad Hoc Networks*, *98*, 102055. https://doi.org/https://doi.org/10.1016/j.adhoc.2019.102055