Breakxit: Sequential Memory Preserving Approach for Few-Shot Image Classification

Susim Mukul Roy B20AI043 | Piyush Arora B20CS086 | Navlika Singh B20AI025

1. Introduction and Problem Statement

Few-shot image classification aims to use very limited support samples to transfer the classifier from base training classes to novel test classes [5, 15, 16] which meets the requirement in application scenarios when training data is scarce. Generally, there are two approaches for mitigating the domain gap,i.e.,domain generalization, and domain adaptation. Domain generalization improves the inherent generalization ability of the learned feature and directly applies it to novel domains without further tuning. In contrast, the domain adaptation uses samples from the novel domain to fine-tune the already-learned feature. For few shot image classification, the domain generalization approach [8, 19]is more explored than the domain adaptation approach [7], because limited support samples hardly provide reliable clues for domain adaptation.

In our project, we explore the feature extractor backbone to understand the cross-domain connection stored in it by introducing memory augmented propagation(MAP) and Matching Feature Hierarchy(MFH) modules to obtain a better embedding which was shown by [17] to be a very significant factor in classification. You can find our code at Github.

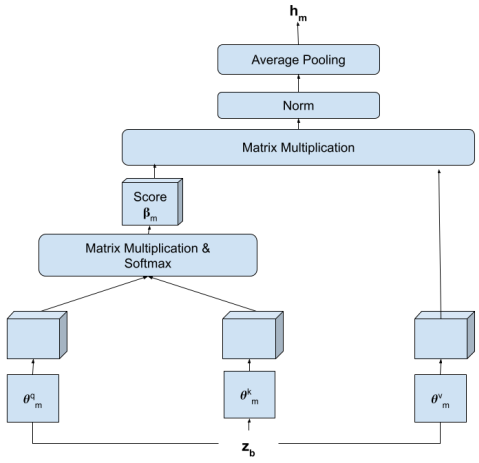

Figure 1. A diagramatic representation of the Matching Feature Hierarchy(**MFH**) module. A single mapper **m** at block **b** extracts embedding $\mathbf{h}_m$ using an attention mechanism containing query θ^q_m and key θ^k_m to build attention scores β_m, with self-attention inferred using value θ^v_m and score β_m

2. Related Work

2.1. Few Shot Image Classification

Few-shot image classification aims at classifying images from novel categories with limited labeled samples. There are two few-shot learning schemes: 1) Meta-learning and 2)Transfer learning. Meta-learning [3,6,15,16] focus on applying metric learning to simulate few-shot scenarios in the training phase and upgrading the training optimizer. Transfer learning [23], on the other hand, solves the problem in a transductive way by re-training a new classifier or adapter for the novel domain while fixing other parts of the network. [3, 7] point out that the meta learning-based methods under perform the transfer learning methods on the cross-domain scenario. In this project, we follow the standard transfer learning routine.

2.2. Domain Generalization & Adaptation

Domain generalization [22] aims to improve cross domain performance without needing to access the target domain. Generalization can be achieved by data augmentation and generation [11, 21], domain-invariant representation learning and feature disentanglement, and general training strategies. In contrast to this, we look at [17] which proposes to combine the different meta-datasets and train a feature extractor such that it produces the most optimal embedding model which is further used during meta-testing. We also note from [1] that a mapping technique between the support and query sets in the extractor itself helps in revealing the inter-relation. Additionally, [9] tries to increase the re-usability of data by storing a few training samples and

reusing them during meta-testing. The proposed mechanism can be viewed as a combination of using memory from during training while at the same time finding the perfect match between the query and the support samples through the extractor model $\mathbf{f}_\theta(\mathbf{x})$ which acts a medium for both the mentioned operations.

3. Methodology

3.1. Generalized embedding for classification

We propose that a model pre-trained on a classification task can generate powerful embeddings for the downstream base learner. To that end, we merge tasks from meta-training set into a single task, which is given by

$$\begin{aligned} D^{new} &= \{(\mathbf{x}_i, y_i)\}_{k=1}^{K} \\ &= \cup\{D_1^{train}, \ldots, D_n^{train}\} \end{aligned} \tag{1}$$

where D_i^{train} is the task from the T. The embedding model is then

$$\phi = \arg\min_{\phi} \mathcal{L}^{ce}(D^{new}; \phi) \tag{2}$$

where $\mathcal{L}^{ce}$ denotes the cross-entropy loss between the predictions and the ground-truth labels. Now, for a task $(D_j^{train}, D_j^{test})$ sampled from meta-testing distribution, we train a base learner on D_j^{train}. The base learner is instantiated as multivariate logistic regression. Its parameters $\theta = \{\mathbf{W}, b\}$ include a weight term $\mathbf{W}$ and a bias term b, given by

$$\theta = \arg\min_{\{W,b\}} \sum_{t=1}^{T} \mathcal{L}_t^{ce}(\mathbf{W} f_\phi(\mathbf{x}_t) + \mathbf{b}, y_t) + \mathcal{R}(\mathbf{W}, b) \tag{3}$$

3.2. Architecture Details

3.3. Matching Feature Hierarchy

The goal of this module is to map an input image $\mathbf{x}$ to a feature map $\mathcal{H}$. These mappers are different from attention-based models in two ways. First, each mapper in our approach is composed of a single attention head, thus we do not rely on fully connected layers to concatenate multi-head outputs. Our feature mappers are therefore separate from each other and each extract their own set of features. Second, our feature mappers are shallow (unit depth), with the learning mechanisms relying on the convolutional layers of the backbone.

The detail of the m-th feature mapper $g(z_{b_m}|\theta_m^g)$, where b_m represents the block preceding the mapper, is illustrated in fig 1. The learned representation $z_{b_m} \in \mathcal{R}^{P \times D^p}$ is separated into $\mathcal{P}$ non-overlapping patches of D^p dimensions. In this work, we use patches of size 1×1, each patch is therefore a 1-D vector of D^p elements. Following [20], an attention map is first computed using two parameterized elements $q(z_{b_m}|\theta_m^g)$ and $K(z_{b_m}|\theta_m^g)$:

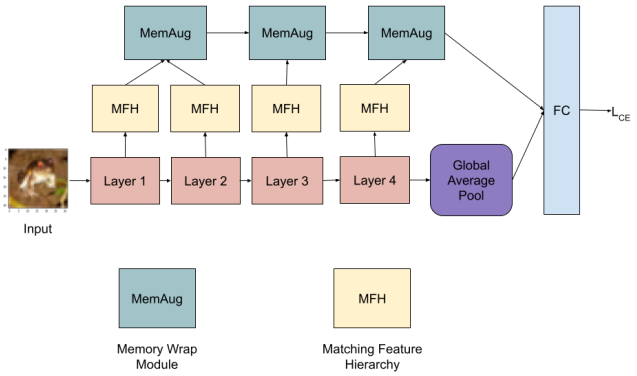

Figure 2. An overview of our architecture

$$\beta_m = Softmax(q(z_{b_m}|\theta_m^g)k(z_{b_m}|\theta_m^g)/\sqrt{d_k} \tag{4}$$

where $\beta_m \in R^{P \times P}$ is the attention score over the patches of z_{b_m}, and $\sqrt{d_k}$ is the scaling factor. Then, we compute the dot-product attention over the patches of β_m using $v(z_{b_m}|\theta_m^v)$ in the following form:

$$a_m = \beta_m v(z_{b_m}|\theta_m^v), \tag{5}$$

where $a_m \in \mathcal{R}^{P \times D^a}$ consists of P patches of D^a dimensions and D^a is the dimension of z_b. Since the backbone is resnet, to keep uniformity, we also add a residual to the computed attention $(a_m + z_{b_m})$. In this case, if the dimensions mismatch $(D^a \neq D^p)$, we use 1×1 convolution of unit stride and kernel size similar to downsampling. Finally, the feature vector $\mathbf{h}_m$ is computed by taking the mean of over the patches(over the P dimension).

3.4. Memory Augmented Propagation

Inspired by [14], we apply the idea of using the memory of the encoder when looking at the feature maps $\mathbf{h}_m$ from the previous module. Specifically, Memory Wrap stores previous examples(memories) that are then used at inference time. Thus, we can imagine it as an attention map at the current step trying to correlate with the attention map at the previous step in order to produce an output which only encapsulates the relevant areas as we progress down the backbone.

More formally, Memory Wrap computes the sparse content attention weights as the sparsemax of the similarity between the input encoding and memory set encodings, thus attaching a content weight w_j to each encoded sample m_j^i. We

model	backbone	miniImageNet		CIFAR-FS	
		1shot	5shot	1shot	5shot
MAML	32-32-32-32	48.70±1.84	63.11±0.92	58.9±1.9	71.5±1.0
Matching Networks	64-64-64-64	43.56±0.84	55.31±0.73	-	-
IMP	64-64-64-64	49.2±0.7	64.7±0.7	-	-
Prototypical Networks	64-64-64-64	49.42±0.78	68.20±0.66	55.5±0.7	72.0±0.6
TAML	64-64-64-64	51.77±1.86	66.05±0.85	-	-
SAML	64-64-64-64	52.22±n/a	66.49±n/a	-	-
GCR	64-64-64-64	53.21±0.80	72.34±0.64	-	-
KTN(Visual)	64-64-64-64	54.61±0.80	71.21±0.66	-	-
PARN	64-64-64-64	55.22±0.84	71.55±0.66	-	-
Dynamic Few-shot	64-64-128-128	56.20±0.86	73.00±0.64	-	-
Relation Networks	64-96-128-256	50.44±0.82	65.32±0.70	55.0±1.0	69.3±0.8
R2D2	96-192-384-512	51.2±0.6	68.8±0.1	65.3±0.2	79.4±0.1
SNAIL	ResNet-12	55.71±0.99	68.88±0.92	-	-
AdaResNet	ResNet-12	56.88±0.62	71.94±0.57	-	-
TADAM	ResNet-12	58.50±0.30	76.70±0.30	58.9±1.9	71.5±1.0
Shot-Free	ResNet-12	59.04±n/a	77.64±n/a	69.2±n/a	84.7±n/a
TEWAM	ResNet-12	60.07±n/a	75.90±n/a	70.4±n/a	81.3±n/a
MTL	ResNet-12	61.20±1.80	75.50±0.80	72.2±0.7	83.5±0.5
Variational FSL	ResNet-12	61.23±0.26	77.69±0.17	-	-
MetaOptNet	ResNet-12	62.64±0.61	78.63±0.46	72.6±0.7	84.3±0.5
Diversityw/Cooperation	ResNet-12	59.48±0.65	75.62±0.48	-	-
Fine-tuning	WRN-28-10	57.73±0.62	78.17±0.49	-	-
LEO-trainval	WRN-28-10	61.76±0.08	77.59±0.12	-	-
RFS-simple	Resnet-12	62.02±0.63	79.64±0.44	71.5±0.8	86.0±0.5
Ours	ResNet-12	**63.95±0.8**	**79.71±0.5**	**72.00±0.9**	84.21±0.66

Table 1. **Comparison to prior work on CIFAR-FS and miniImageNet.** Average few-shot classification accuracies %(with 95%) confidence intervals on miniImageNet and CIFAR-FS. a-b-c-d denotes a 4-layer convolutional network with a, b, c, and d filters in each layer.

compute content attention weights using the cosine similarity as in [4],replacing the softmax function with a sparsemax.

$$\mathbf{w} = sparsemax(cosine[e_x, \mathbf{M}_{S_i}]) \tag{6}$$

Since we are using the sparse max function, the memory vector only includes information from few samples of the memory. In this way, each sample contributes in a significant way, helping us to achieve output explainability. Similarly to [4], we compute the memory vector $\mathbf{v}_{S_i}$ as the weighted sum of memory set encodings, where the weights are the content attention weights:

$$\mathbf{v}_{S_i} = \mathbf{M}_{S_i}^T \mathbf{w}. \tag{7}$$

Finally, the last layer l_f takes the concatenation of the memory vector and the encoded input, and returns the final output

$$o_i = g(x_i) = l_f(e_{x_i}, \mathbf{v}_{S_i}) \tag{8}$$

The role of the memory vector is to enrich the input encoding with additional features extracted from similar samples, possibly missing on the current input. On average, considering the whole memory set and thanks to the cosine similarity, strong features of the target class will be more represented than features of other classes, helping the network in the decision process.

Post-Processing: We apply the Memory Wrap on consecutive outputs of equation 5. These four outputs are sequentially passed producing a feature vector at it's last block. This feature vector is concatenated with the output from resnet12 and is used to calculate the cross-entropy loss.

4. Experiment

We conducted experiments on four widely used few-shot image recognition benchmarks: miniImageNet [18] and CIFAR-FS [2]. The first one is derivative of ImageNet, while the last one is reorganized from the standard CIFAR-100 dataset.

4.1. Setup

Our Resnet12 is identical to that used in [10, 13]. The network consists of 4 residual blocks, where each has 3 convolutional layers with 3×3 kernel; a 2×2 max-pooling

layer is applied after each of the first 3 blocks; and a global average-pooling layer is on top of the fourth block to generate the feature embedding. Each of the feature map from the 4 blocks are fed to 4 individual Mappers. We now experiment with different ways of embedding six mappers throughout the backbone levels. We compare: 1)putting all mappers on the last layer (0-0-0-6); 2) a single mapper per block(1-1-1-1); 3)distributing mappers more equally(1-2-2-1); and 4) employing a upsame-downsample growth strategy(1-2-1-2). Our experiments reveal that the last strategy provided the highest accuracy. On the other hand, all the memory-capturing modules have the same input and output vector dimensions so that we have parity in storing the previous information for all the multi-scale feature maps.

Optimization: We use SGD optimizer with a momentum of 0.9 and a weight decay of $5e^{-4}$.Each batch consists of 64 samples. The learning rate is initialized as 0.05 and decayed with a factor of 0.1 by three times for CIFAR-FS except for miniImageNet where we only decay twice as the third decay has no effect. We train 100 epochs for miniImageNet and 90 epochs for both CIFAR-FS.

Data Augmentation: When training the embedding network on transformed meta-training set, we adopt random crop, color jittering, and random horizontal flip as in [10]. For meta-testing stage, we train an N-way logistic regression base classifier. We use the implementations in scikit learn for the base classifier.

5. Results

The miniImageNet dataset is a standard benchmark for few-shot learning algorithms for recent works. It consists of 100 classes randomly sampled from the ImageNet; each class contains 600 downsampled images of size 84x84. We follow the widely-used splitting protocol proposed in [12], which uses 64 classes for meta-training, 16 classes for meta-validation, and the remaining 20 classes for meta-testing. During meta-testing, the accuracy during each run is the mean accuracy of 1000 randomly sampled tasks. We report the median of 3 runs in Table 1. Our baseline with ResNet-12 is already comparable with the state-of-the-art MetaOptNet [10] on miniImageNet. We also note that this baseline is comparable to Prototypical Networks [15] and MetaOptNet [10] on CIFAR-FS dataset.

6. Key Observations

We did important ablation studies to find the relevance of every part. We find the following to be the major factors:

- We can argue that our data sample classes are mutually exclusive to each other resulting in an efficient multi-way classification task instead of the standard multi-task learning.
- The Matching Feature technique was removed from some layers while doing empirical studies. We found the average accuracy to decrease by 2%. This shows that our rationale of considering only features which the convolution layers look at as a major factor in deciding the class when given only few query samples is true.
- Accompanied with important features, we also have multi-scale instrinsic memory in the feature maps which help to store the helpful knowledge gained from previous layers of the backbone feature extractor.

7. Conclusion

This project proposes to extract and match feature vectors for few-shot image classification. This contrasts with the use of a monolithic single-vector representation, which is a popular strategy in that context. The simple baseline of training on the entire set and just remembering the top features at different scales is a fast yet susceptible technique. We end our experiments with the note that we can extend this to self-supervised approaches, since the set of features provide more choices for the comparison of different variations of single images and also memorizing in a top-down manner of feature importance.

References

[1] Arman Afrasiyabi, Hugo Larochelle, Jean-François Lalonde, and Christian Gagn'e. Matching feature sets for few-shot image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9014–9024, 2022. 1

[2] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *CoRR*, abs/1805.08136, 2018. 3

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Y. Wang, and Jia-Bin Huang. A closer look at few-shot classification. *ArXiv*, abs/1904.04232, 2019. 1

[4] Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. Bam! born-again multi-task networks for natural language understanding. *CoRR*, abs/1907.04829, 2019. 3

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. 1

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017. 1

[7] Yunhui Guo, Noel C. Codella, Leonid Karlinsky, James V. Codella, John R. Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning, 2020. 1

[8] Zhengdong Hu, Yifan Sun, and Yi Yang. Switch to generalize: Domain-switch learning for cross-domain few-shot classification. In *International Conference on Learning Representations*, 2022. 1

[9] Biagio La Rosa, Roberto Capobianco, and Daniele Nardi. A self-interpretable module for deep image classification on small data. *Applied Intelligence*, 53(8):9115–9147, Aug. 2022. 1

[10] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. *CoRR*, abs/1904.03758, 2019. 3, 4

[11] Hanwen Liang, Qiong Zhang, Peng Dai, and Juwei Lu. Boosting the generalization capability in cross-domain few-shot learning via noise-enhanced supervised autoencoder. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9404–9414, 2021. 1

[12] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. 4

[13] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. *CoRR*, abs/1905.04398, 2019. 3

[14] Biagio La Rosa, Roberto Capobianco, and Daniele Nardi. Memory wrap: a data-efficient and interpretable extension to image classification models. *CoRR*, abs/2106.01440, 2021. 2

[15] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017. 1, 4

[16] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning, 2018. 1

[17] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020. 1

[18] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096, 2019. 3

[19] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation, 2020. 1

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2

[21] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation, 2018. 1

[22] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. Generalizing to unseen domains: A survey on domain generalization, 2022. 1

[23] Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *Journal of Big Data*, 3, 2016. 1