Aligning Visual Prototypes with BERT Embeddings for Few-Shot Learning

Kun Yan
kyan2018@pku.edu.cn
School of Software and Microelectronics, Peking University, China

Zied Bouraoui
zied.bouraoui@cril.fr
CRIL - University of Artois & CNRS, France

Ping Wang*†
pwang@pku.edu.cn
National Engineering Research Center for Software Engineering, Peking University, China

Shoaib Jameel
shoaib.jameel@gmail.com
School of Computer Science and Electronic Engineering, University of Essex, UK

Steven Schockaert
schockaerts1@cardiff.ac.uk
School of Computer Science and Informatics, Cardiff University, UK

ABSTRACT
Few-shot learning (FSL) is the task of learning to recognize previously unseen categories of images from a small number of training examples. This is a challenging task, as the available examples may not be enough to unambiguously determine which visual features are most characteristic of the considered categories. To alleviate this issue, we propose a method that additionally takes into account the names of the image classes. While the use of class names has already been explored in previous work, our approach differs in two key aspects. First, while previous work has aimed to directly predict visual prototypes from word embeddings, we found that better results can be obtained by treating visual and text-based prototypes separately. Second, we propose a simple strategy for learning class name embeddings using the BERT language model, which we found to substantially outperform the GloVe vectors that were used in previous work. We furthermore propose a strategy for dealing with the high dimensionality of these vectors, inspired by models for aligning cross-lingual word embeddings. We provide experiments on miniImageNet, CUB and tieredImageNet, showing that our approach consistently improves the state-of-the-art in metric-based FSL.

* means the corresponding author
† Also with the School of Software and Microelectronics, Peking University and the Key Laboratory of High Confidence Software Technologies (PKU), Ministry of Education.

CCS CONCEPTS
• Computing methodologies → Object recognition; Object identification.

KEYWORDS
Few-shot learning, BERT, multi-modal, metric-based learning

ACM Reference Format:
Kun Yan, Zied Bouraoui, Ping Wang, Shoaib Jameel, and Steven Schockaert. 2021. Aligning Visual Prototypes with BERT Embeddings for Few-Shot Learning. In Proceedings of the 2021 International Conference on Multimedia Retrieval (ICMR ’21), August 21–24, 2021, Taipei, Taiwan. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3460426.3463641

1 INTRODUCTION
Recent years have witnessed significant progress in image classification and related computer vision tasks [15, 20, 39, 44, 52], but most existing methods still require an abundance of labeled training examples. This stands in stark contrast with humans’ ability to learn new categories from even a single example. This observation has fuelled research on designing systems that are capable of recognizing new image categories after only seeing a small number of examples, a task which is known as few-shot learning (FSL). In this paper, we focus in particular on metric-based FSL methods [19, 37, 40, 43, 48], which combine strong empirical performance with conceptual simplicity.

Metric-based methods aim to learn an embedding space which encourages generalization, i.e. where images from the same class are likely to have similar embeddings, even for unseen classes. An image can then be categorized based on its similarity to prototypes of the considered classes. Despite significant progress in recent years, however, few-shot learning remains highly challenging. To alleviate the inherent difficulty of this task, some authors have proposed models that additionally take into account the name of the image classes. While these class names may not be available in all application settings, in those settings where they are, we can intuitively expect that they should provide us with meaningful
prior knowledge. Two notable examples of models that rely on
class names are AM3 [53] and TRAML [22], both of which use the
GloVe [32] word embedding model for representing class names. In
particular, the AM3 model tries to predict visual prototypes from the
embeddings of the class names, while TRAML uses the similarity
encoded by the word vectors to adapt the margin of the classifier.

However, standard word vectors, such as those from GloVe, are
strongly influenced by topical similarity. This is illustrated in Table 1,
which shows the top-3 most similar classes from miniImageNet
for three example targets. For instance, the nearest neighbours of
catamaran include snorkel and jellyfish. These words are all clearly
topically related, but catamarans are not similar to snorkels or jel-
lyfish. This is problematic for few-shot learning, where we would
intuitively want that class names with similar embeddings denote
categories of the same kind. To address this issue, we propose a sim-
ple strategy for obtaining class name embeddings using the BERT
masked language model [6]. We qualitatively observe that the re-
sulting embeddings are indeed better suited for grouping classes
that are conceptually similar. For instance, as can be seen in Table 1,
with the proposed BERT embeddings, the top 2 nearest neighbours
are now also boats (being the only remaining boat classes in mini-
ImageNet), while the third neighbour is a vehicle. Furthermore,
as the example of house finch shows, the BERT embeddings also
tend to model semantic relatedness at a finer-grained level: while the
top neighbours for GloVe are all animals, none of them are birds.
In contrast, the top two neighbours for BERT are birds.

However, BERT embeddings also have the drawback of being
higher-dimensional: the BERT-large vectors on which we rely are
1024-dimensional, compared to 300 dimensions for the standard
GloVe embeddings. This makes it difficult to predict visual proto-
types from these vectors. Therefore, rather than predicting visual
prototypes from the class names, we model the visual and text-based
prototypes separately. Moreover, we also propose a dimensionali-
ty reduction strategy, inspired by work on aligning cross-lingual
word embeddings [1], which aims to find a subspace of the BERT
embeddings that is maximally aligned with the visual prototypes.
As illustrated in Table 1, the resulting embeddings remain at least
as useful as the original BERT embeddings, despite only being
50-dimensional. In fact, some of the nearest neighbours for the low-
dimensional vectors are arguably better than those of the BERT
eMBEDDINGS THEMSELVES, E.g. TOUCAN IS MORE SIMILAR TO HOUSE FINCH
THAN GOOSE IS, WHILE SCOREBOARD AND STREET SIGN ARE MORE MEANINGFUL
NEIGHBOURS OF HORIZONTAL BAR THAN UNICYCLE AND EAR.

The main contributions of this paper are as follows: (i) we pro-
solve a simple model of incorporating class names into metric-based
FSL models, in which visual prototypes and text-based prototypes
are decoupled; (ii) we propose and evaluate several strategies for
learning class name embeddings using BERT; (iii) we propose a
strategy for dealing with the high dimensionality of the BERT em-
beddings by identifying the subspace of these embeddings which is
most aligned with the visual prototypes.

2 RELATED WORK

Most few-shot learning methods can be divided into metric-based
[17, 37, 43, 55] and meta-learning based [7, 25, 34] methods, al-
though some other directions have also been explored, such as

| Table 1: Most similar miniImageNet classes to house finch, horizontal bar and catamaran, according to class name em-
beddings obtained using GloVe, BERT and the proposed projection of the BERT embeddings onto a 50-dimensional space (BERT_{proj}). |
|---|---|---|---|
| GloVe | snorkel | ladybug | pencil box |
|       | yawl  | komonodor | aircraft carrier |
|       | school bus | triceratops | beer bottle |
| BERT | yawl | goose | parallel bars |
|       | aircraft carrier | toucan | unicycle |
|       | school bus | ladybug | ear |
| BERT_{proj} | yawl | toucan | parallel bars |
|         | aircraft carrier | robin | scoreboard |
|         | school bus | ladybug | street sign |

hallucination based [11, 51, 58] and parameter-generation based
[9, 26] methods. Our focus in this paper is on metric-based methods,
which essentially aim to learn a generalizable visual embedding
space. Early metric-based approaches used deep Siamese networks
to compute the similarity between training and test images for
the one-shot object recognition task [19]. In these cases, a query
image is simply assigned to the class of the most similar training
image. Going beyond one-shot learning, [48] proposed Matching
Network, which uses a weighted nearest-neighbor classifier with
an attention mechanism over the features of labeled examples. An-
other important contribution of that work is the introduction of a
new training scheme called episode-based learning, which uses a
training procedure that is more closely aligned with the standard
test setting for few-shot learning (see Section 3). The ProtoNet
model from [40] generates a visual prototype for each class, by
simply averaging the embeddings of the available training images.
The class of a query image is then predicted by computing its Eu-
clidean distance to these prototypes. In the Relation Network [43],
rather than fixing the metric to be Euclidean, the model learns a
depth distance metric to compare each query-support image pair.
In addition, some works have used Graph Convolutional Networks
[18] to exploit the relationship among support and query examples
[17, 37]. The FEAT model, proposed by [55], uses a transformer
[47] to contextualize the image features relative to the support set
in a given task. Recently, the Earth Mover’s Distance (EMD) has
been adopted as a metric in DeepEMD [56] to compute a struc-
tural distance between dense image representations to determine
image relevance. The aforementioned methods all rely on global
image features. A few methods have also been proposed that aim
to identify finer-grained local features, such as DN4 [24], SAML [10],
STANet [54] and CTM [23].

The aforementioned methods only depend on visual features. A
few methods also take into account the class names. In AM3
[53], prototypes are constructed as a weighted average of a visual
prototype and a prediction from the class name. The relative weight
of both modalities is computed adaptively and can differ from class
to class. More recently, [22] used the class names as part of a margin
We now explain how BERT [6] is used to get vector representations $\boldsymbol{v}_p$ of class names. First note that BERT represents frequent words as a single token and encodes less common words as sequences of subword tokens, called word-pieces. Each of these tokens $t$ is associated with a static vector $\boldsymbol{t} \in \mathbb{R}^m$. The token vectors are used to construct the initial representation of a given sentence $s = t_1, \ldots, t_n$, which is subsequently fed to a deep transformer model. The output of this deep transformer model again consists of a sequence of token vectors, which intuitively represent the meaning of each token in the specific context of the given sentence. Let us write $m(s, i)$ for the output representation of $t_i$. When training BERT, some tokens of each input sentence are replaced by the special token [MASK]. If the token $t_i$ was masked, the output vector $m(s, i)$ acts as a prediction for the missing token.

Let $C$ be the set of classes. We first collect for each class $c \in C$ a bag of sentences $S(c) = s_1, \ldots, s_m$ mentioning the name of this class. In particular, for each class name, we sample $m = 1000$ such sentences from a given text corpus. We consider two strategies for learning class embeddings from these sentences. For the first strategy, we replace the entire class name by a single [MASK] token, and we use the corresponding output vector as the representation of $c$. We then take the average of the vectors we thus obtain across the $m$ sentences. In practice, the classes often correspond to WordNet synsets, meaning that we may have several synonymous names. In such cases, we first get a vector for each name from the synset (each learned from 1000 sentences), and then average the resulting vectors. The underlying assumption of this first approach is that when the $i$th token is masked, the prediction $m(s, i)$ essentially encodes what the given sentence reveals about the meaning of the class $c$. This strategy has the important advantage that it can naturally deal with class names that consist of multiple word-piece tokens. The second approach uses the full sentence as input, without masking any words. Following common practice [12, 33], the representation of $c$ is then obtained by averaging the output vectors of all the word-piece tokens corresponding to $c$. We write $n^c_{\text{mask}}$ and $n^c_{\text{nomask}}$ for the embeddings obtained with the first and second method respectively. In addition to using $n^c_{\text{mask}}$ or $n^c_{\text{nomask}}$ individually, we will also experiment with their concatenation $n^c_{\text{mask}} \oplus n^c_{\text{nomask}}$. We will furthermore consider variants in which other types of word vectors are included, such as the GloVe embedding $n^c_{\text{glove}}$.

### 4.3 Dimensionality Reduction

One disadvantage of BERT embeddings is that they are high dimensional, a problem which is exacerbated when using concatenations of several types of class name embeddings. Furthermore, we can expect that only some of the information captured by the class name embeddings may be relevant for image classification. To address both shortcomings, we propose a Correlation Exploration Module (CEM), whose aim is to find a low-dimensional subspace of the class name embeddings.

Specifically, we aim to find linear mappings $\mathbf{A} \in \mathbb{R}^{m_t \times d}$ and $\mathbf{B} \in \mathbb{R}^{m_s \times d}$, where $m_t$ is the dimension of the class name embeddings, $m_s$ is the dimension of the visual features and $d < \min(m_t, m_s)$. Let $n^c$ be the considered embedding of the name of class $c$, and let $\boldsymbol{v}_p^c$ be the visual prototype of the same class (for a given episode). Intuitively, we want $n^c A$ to maximally retain the predictive information about $\boldsymbol{v}_p^c$ that is captured by $n^c$. A natural strategy to find suitable matrices $\mathbf{A}$ and $\mathbf{B}$ is to use Canonical Correlation Analysis (CCA).
These matrices are then chosen such that the correlations between the coordinates of $n^A$ and the corresponding coordinates of $v^B$ are maximized. The advantage of using CCA is that it is based on well-founded statistical principles and straightforward to compute. However, it was noted by [1] that CCA is a sub-optimal choice for aligning cross-lingual word embeddings, which suggests that it may be a sub-optimal choice for cross-modal alignment as well. As pointed out in that paper, CCA can be seen as the combination of three linear transformations: (i) whitening of the initial vectors in the two embedding spaces, (ii) aligning the two spaces using orthogonal transformations and (iii) dimensionality reduction. It was found that better results can often be achieved by introducing an additional de-whitening step, which restores the original covariances. We will consider variants with and without this de-whitening step, which we will refer to as CCA+D and CCA respectively. The details of both variants are provided in the Appendix.

### 4.4 Classification Model

To classify a query image, we follow the set-up of ProtoNet, changing only the way in which the similarity between query images and prototypes is computed. In the case of ProtoNet, we have:

$$s_1(q, v^c_p) = -\|f_0(q) - v^c_p\|^2_2$$

The scores for each of the classes are then fed to a softmax layer to obtain class probabilities; the overall model is trained using the cross-entropy loss. In the case of FEAT, $f_0(q)$ and $v^c_p$ are first contextualized using a transformer, before computing the squared Euclidean distance for comparing vectors that come from different distributions [9], while keeping the squared Euclidean distance for comparing $v^c_p$ and $f_0(q)$. This leads to the following similarity score:

$$s_2(q, v^c_p) = -\|f_0(q) - v^c_p\|^2_2 + \lambda \cos(f_0(q), g_\psi(n))$$

where $\lambda$ is a hyper-parameter to control the contribution of the class name embeddings. To learn $g_\psi$, we use a shallow network consisting of a linear transformation onto a 512-dimensional layer with ReLU activation and batch normalization [16], followed by another linear transformation.

As mentioned above, learning a suitable mapping $g_\psi$ is challenging when $n^A$ is high-dimensional. Rather than learning the parameters of this mapping as part of the model, we therefore propose to use the mappings $A$ and $B$ that were found by the Correlation Exploration Module. The similarity score thus becomes:

$$s_3(q, v^c_p) = -\|f_0(q) - v^c_p\|^2_2 + \lambda \cos(f_0(q)B, n^A)$$

## 5 EXPERIMENTS

### 5.1 Experimental Setup

#### 5.1.1 Datasets

We conduct experiments on three benchmark datasets: miniImageNet [48], tieredImageNet [35] and CUB [49]. MiniImageNet is a subset of the ImageNet dataset [5]. It consists of 100 classes, each with 600 labeled images of size $84 \times 84$. We adopt the common setup introduced by [34], which defines a split of 64, 16 and 20 classes for training, validation and testing respectively. TieredImageNet is a larger-scale dataset with more classes, containing 351, 97 and 160 classes for training, validation and testing. The CUB dataset contains 200 classes and 11,788 images in total. We
used the splits from [4], where 100 classes are used for training, 50 for validation, and 50 for testing.

5.1.2 Training and Test Setting. We evaluate our method on 5-way 1-shot and 5-way 5-shot settings. We train 50,000 episodes in total for miniImageNet, 80,000 episodes for tieredImageNet and 40,000 episodes for CUB. During the test phase, 600 test episodes are generated. We report the average accuracy as well as the corresponding 95% confidence interval over these 600 episodes.

5.1.3 Class Name Embeddings. As baseline class name embedding strategies, we used 300-dimensional FastText \(^1\) [2], GloVe \(^2\) [32] and skip-gram embeddings\(^3\) [28]. For the BERT embeddings, we use the BERT-large-uncased model\(^4\), which yields 1024 dimensional vectors. To obtain the \(n^c_{\text{mask}}\) and \(n^c_{\text{nomask}}\) vectors, we used the May 2016 dump of the English Wikipedia. In addition to using the vectors \(n^c_{\text{mask}}\) (referred to as BERT\(_{\text{mask}}\)) and \(n^c_{\text{nomask}}\) (referred to as BERT\(_{\text{nomask}}\)), we also experiment with the following concatenations: \(n^c_{\text{mask}} \oplus n^c_{\text{nomask}}\) (referred to as CON\(_1\)) and \(n^c_{\text{mask}} \oplus n^c_{\text{nomask}} \oplus n^c_{\text{glove}}\) (referred to as CON\(_2\)).

5.1.4 Implementation Details. We have implemented\(^5\) our model using the PyTorch-based framework provided by [4]. As the backbone network for the visual feature embeddings, we used ResNet-10 [13] for the ablation study in Section 5.2 and ResNet-12 and Conv-64 [40] for our comparison with the state-of-the-art in Section 5.3. Conv-64 is the standard choice for CUB. It has four layers with each layer consisting of a \(3 \times 3\) convolution and filters, followed by batch normalization, a ReLU non-linearity, and \(2 \times 2\) max-pooling. All experiments are trained from scratch using the Adam optimizer with an initial learning rate of 0.0001. In experiments where the mapping network \(g_{\psi}\) is used, this network is trained separately, with a learning rate of 0.0001. The remaining parameters are selected based on the validation set. In particular, the coefficient \(\lambda\) is chosen from \([1, 2, ..., 10]\). For miniImageNet and CUB, the optimal value was \(\lambda = 5\); for tieredImageNet we obtained \(\lambda = 6\). We similarly select the type of class name embedding from \{BERT\(_{\text{mask}}\), CON\(_1\), CON\(_2\)\} and the number of dimensions from \([25, 50, 100, 200]\). In all cases, we used the CCA+D method for reducing the number of dimensions. For miniImageNet, 50-dimensional CON\(_2\) was selected; for CUB, 50-dimensional CON\(_1\) was selected; for tieredImageNet, 100-dimensional CON\(_2\) was selected.

5.2 Ablation Study

Our ablation study is based on the ProtoNet model. All experiments in this section are conducted on miniImageNet using ResNet-10 as the feature extractor.

5.2.1 Word Embedding Models. We first explore the impact of the considered word embedding model. We found that the BERT-based approach is sensitive to sentence segmentation errors. To mitigate the impact of such errors, we only considered sentences whose length is below a maximum of \(t_{\text{max}}\) word-piece tokens, where we considered values of \(t_{\text{max}}\) between 16 and 100. The results are shown in Table 2, where we used the variant of our model with the learned mapping network \(g_{\psi}\) for 5-way 5-shot learning. The results show that BERT\(_{\text{mask}}\) consistently outperforms BERT\(_{\text{nomask}}\), while \(t_{\text{max}} = 64\) achieves the best balance between avoiding sentences with segmentation issues and removing too many sentences. BERT\(_{\text{mask}}\) Perform consistently better than GloVe, which achieves the best performance among the baseline models. The static BERT input vectors (shown as BERT\(_{\text{static}}\)) achieve the worst performance overall. In the remainder of the experiments, we fix \(t_{\text{max}} = 64\).

5.2.2 Correlation Exploration Module. We now analyze the usefulness of the Correlation Exploration Module, comparing in particular the CCA and CCA+D alignment strategies. Note that when the mapping network \(g_{\psi}\) is used we are forced to keep the dimensionality the same as that of the visual features (which is 512 in the case of ResNet), whereas with the CCA-based alignment methods, we can use lower-dimensional textual prototypes. Table 3 explores the effect of the dimensionality \(d\) of the textual prototypes. The best results were found for \(d = 50\). The results for \(d = 50\) are similar to the results we obtained with the mapping network \(g_{\psi}\) in Table 2, with CCA+D performing slightly better and CCA performing slightly worse than BERT\(_{\text{mask}}\).

However, a key advantage of the CCA methods is that we can further increase the dimensionality of the class name embeddings, without increasing the number parameters of the classification.

---

1. https://fasttext.cc/docs/en/crawl-vectors.html
2. https://nlp.stanford.edu/projects/glove/
3. https://code.google.com/archive/p/word2vec/
4. Available from https://github.com/huggingface/transformers
5. https://github.com/yankun-pku/Aligning-Visual-Prototypes-with-BERT-Embeddings-for-Few-shot-Learning

Table 2: Comparison of the performance of different word embedding models on miniImageNet, for the 5-way 5-shot setting using the learned mapping network \(g_{\psi}\).

| Word Emb. | \(t_{\text{max}}\) | Accuracy   |
|-----------|----------------|------------|
| FastText  | 74.97 + 0.65   | 75.99 + 0.64 |
| GloVe     | 75.30 + 0.61   | 76.40 + 0.63 |
| Skip-gram | 74.91 + 0.66   | 76.17 + 0.67 |
| BERT\(_{\text{static}}\) | 74.50 + 0.63   | 75.98 + 0.68  |
| BERT\(_{\text{mask}}\) | 16             | 74.75 + 0.67  |
| BERT\(_{\text{mask}}\) | 32             | 75.47 + 0.68  |
| BERT\(_{\text{mask}}\) | 64             | \textbf{76.30} + 0.76 |
| BERT\(_{\text{mask}}\) | 100            | 76.17 + 0.67  |
| BERT\(_{\text{nomask}}\) | 16             | 74.73 + 0.66  |
| BERT\(_{\text{nomask}}\) | 32             | 74.79 + 0.67  |
| BERT\(_{\text{nomask}}\) | 64             | 75.62 + 0.65  |
| BERT\(_{\text{nomask}}\) | 100            | 74.76 + 0.69  |

Table 3: Results for textual prototypes of different dimensionality on miniImageNet, for the 5-way 5-shot setting.

| Dim | CCA | CCA+D |
|-----|-----|-------|
| 25  | 76.21 + 0.62 | 75.99 + 0.64 |
| 50  | 76.17 + 0.67 | \textbf{76.40} + 0.63 |
| 100 | 75.91 + 0.66 | 76.32 + 0.65 |
| 200 | 75.98 + 0.68 | 76.17 + 0.64 |
Table 4: Comparison of different alignment strategies on miniImageNet, for the 5-way 5-shot setting, with $d = 50$.

| Alignment Method | Word Emb. | Accuracy     |
|------------------|-----------|--------------|
| $g_\theta$       | BERT$_{\text{mask}}$ | 76.30 ± 0.76 |
| $g_\phi$         | CON$_1$   | 75.72 ± 0.60 |
| $g_\phi$         | CON$_2$   | 75.16 ± 0.79 |
| CCA              | BERT$_{\text{mask}}$ | 76.21 ± 0.62 |
| CCA              | CON$_1$   | 76.31 ± 0.67 |
| CCA              | CON$_2$   | 76.50 ± 0.62 |
| CCA+D            | BERT$_{\text{mask}}$ | 76.40 ± 0.63 |
| CCA+D            | CON$_1$   | 76.61 ± 0.65 |
| CCA+D            | CON$_2$   | 76.82 ± 0.64 |

Figure 2: 5-way 5-shot accuracy with different $\lambda$ values on the miniImageNet validation dataset.

Table 5: Comparison with AM3 on miniImageNet (using ResNet-12 in all cases), showing mean accuracies (%) with a 95% confidence interval.

| Word Emb. | Base Met. | AM3     | Ours     |
|-----------|-----------|---------|----------|
| 5-way 1-shot setting:                      |          |         |          |
| GloVe     | ProtoNet  | 62.43 ± 0.80 | 63.49 ± 0.67 |
| BERT$_{\text{mask}}$ | ProtoNet  | 62.11 ± 0.39 | 63.84 ± 0.32 |
| CON$_1$   | ProtoNet  | 62.14 ± 0.41 | 64.13 ± 0.45 |
| CON$_2$   | ProtoNet  | 62.03 ± 0.46 | 64.53 ± 0.37 |
| 5-way 5-shot setting:                      |          |         |          |
| GloVe     | ProtoNet  | 74.87 ± 0.65 | 78.72 ± 0.64 |
| BERT$_{\text{mask}}$ | ProtoNet  | 74.72 ± 0.64 | 79.10 ± 0.63 |
| CON$_1$   | ProtoNet  | 74.24 ± 0.68 | 79.26 ± 0.65 |
| CON$_2$   | ProtoNet  | 74.09 ± 0.70 | 79.37 ± 0.64 |

Table 6: Comparison with TRAML on miniImageNet (using ResNet-12 in all cases), showing mean accuracies (%) with a 95% confidence interval.

| Word Emb. | Base Met. | TRAML | Ours     |
|-----------|-----------|-------|----------|
| 5-way 1-shot setting:                      |          |       |          |
| GloVe     | ProtoNet  | 60.31 ± 0.48 | 63.49 ± 0.67 |
| GloVe     | AM3(ProtoNet) | 67.10 ± 0.52 | 67.75 ± 0.39 |
| CON$_2$   | AM3(ProtoNet) | -     | 68.42 ± 0.51 |
| 5-way 5-shot setting:                      |          |       |          |
| GloVe     | ProtoNet  | 77.94 ± 0.57 | 78.72 ± 0.64 |
| GloVe     | AM3(ProtoNet) | 79.54 ± 0.60 | 80.62 ± 0.76 |
| CON$_2$   | AM3(ProtoNet) | -     | 81.29 ± 0.59 |

5.3 Experimental results

AM3 [53] and TRAML [22] are the most direct competitors of our method, as these models also use class name embeddings. For this reason, we first present a detailed comparison with these methods in Section 5.3.1. Subsequently, in Section 5.3.2 we present a more general comparison with the state-of-the-art in few-shot learning.

5.3.1 Comparison with AM3 and TRAML. The comparison with AM3 can be found in Table 5, where we also show the impact of different types of class name embeddings. As can be seen, our proposed method outperforms AM3 in all cases, both in the 1-shot and 5-shot setting. This confirms the usefulness of decoupling the visual and textual prototypes, as this is the key difference between our model and AM3 when low-dimensional vectors, such as those from the GloVe model, are used. Furthermore, we can see that AM3 is not able to take advantage of the higher-dimensional embeddings, with the results for BERT, CON$_1$ and CON$_2$ all being worse than those for GloVe. This can be explained from the observation that these higher-dimensional class name embeddings result in a substantially
higher number of parameters in the case of AM3, leading to overfitting. In contrast, thanks to the correlation exploration module, our method can exploit the additional semantic information that is encoded in the higher-dimensional embeddings without introducing any additional parameters in the classification model. In both the 1-shot and 5-shot settings, our model achieves the best results with CON2 embeddings, which is in accordance with our findings from Section 5.2.

Regarding the TRAML model, as we did not have access to the source code, we only compare our method against the published results from the original paper [22]. As the base method, they considered both ProtoNet and AM3. As can be seen in Table 6, our method outperforms TRAML in both of these settings, for 1-shot as well as 5-shot learning. This is even the case if GloVe vectors are used for our model, although the best results are obtained when using the CON2 embeddings for our model, while still using the GloVe vectors for the AM3 base model.

5.3.2 Comparison with the State-of-the-Art. Tables 7, 8 and 9 compare our model with existing methods on the miniImageNet, CUB and tieredImageNet datasets respectively, where miniImageNet and tieredImageNet are standard benchmarks for few-shot learning. CUB, which consists of 200 bird classes, allows us to evaluate the performance of our model on finer-grained classes. The performance of all methods is generally impacted by the choice of the backbone network. To allow for a fair comparison with different

| Method                        | Backbone | Type | 5-way 1-shot | 5-way 5-shot |
|-------------------------------|----------|------|--------------|--------------|
| MAML [7]                      | Conv-64  | Meta | 48.70 ± 1.75 | 63.15 ± 0.91 |
| Reptile [30]                  | Conv-64  | Meta | 47.07 ± 0.26 | 62.74 ± 0.37 |
| LEO [36]                      | WRN-28   | Meta | 61.76 ± 0.08 | 77.59 ± 0.12 |
| MTL [42]                      | ResNet-12| Meta | 61.20 ± 1.80 | 75.50 ± 0.80 |
| MetaOptNet-SVM [21]           | ResNet-12| Meta | 62.64 ± 0.61 | 78.63 ± 0.46 |

Table 7: The mean accuracies (%) with a 95% confidence interval on the miniImageNet dataset.

| Method                        | Backbone | Type | 5-way 1-shot | 5-way 5-shot |
|-------------------------------|----------|------|--------------|--------------|
| Matching Net [48]             | Conv-64  | Metric | 43.56 ± 0.84 | 55.31 ± 0.73 |
| ProtoNet [40]                 | Conv-64  | Metric | 49.42 ± 0.78 | 68.20 ± 0.66 |
| RelationNet [43]              | Conv-64  | Metric | 50.44 ± 0.82 | 65.32 ± 0.70 |
| ProtoNet [40]                 | ResNet-12| Metric | 56.52 ± 0.45 | 74.28 ± 0.20 |
| TADAM [31]                    | ResNet-12| Metric | 58.50 ± 0.30 | 76.70 ± 0.38 |
| Baseline++ [4]                | ResNet-18| Metric | 51.87 ± 0.77 | 75.68 ± 0.63 |
| SimpleShot [50]               | ResNet-18| Metric | 62.85 ± 0.20 | 80.02 ± 0.14 |
| CMT [23]                      | ResNet-18| Metric | 64.12 ± 0.82 | 80.51 ± 0.13 |
| AM3(ProtoNet, GloVe)          | ResNet-12| Metric | 62.43 ± 0.80 | 74.87 ± 0.65 |
| AM3(ProtoNet++) [53]          | ResNet-12| Metric | 65.21 ± 0.49 | 75.20 ± 0.36 |
| TRAML(ProtoNet) [22]          | ResNet-12| Metric | 60.31 ± 0.48 | 77.94 ± 0.57 |
| CAN [14]                      | ResNet-12| Metric | 63.85 ± 0.48 | 79.44 ± 0.34 |
| DSN-MR [38]                   | ResNet-12| Metric | 64.60 ± 0.48 | 79.51 ± 0.50 |
| FEAT [55]                     | ResNet-12| Metric | 66.78         | 82.05        |
| DeepEMD [56]                  | ResNet-12| Metric | 65.91 ± 0.82 | 82.41 ± 0.56 |
| Ours(ProtoNet)                | ResNet-12| Metric | 64.53 ± 0.37 | 79.37 ± 0.64 |
| Ours(AM3,ProtoNet)            | ResNet-12| Metric | 68.42 ± 0.51 | 81.29 ± 0.59 |
| Ours(FEAT)                    | ResNet-12| Metric | 67.84 ± 0.45 | 83.17 ± 0.72 |
| Ours(DeepEMD)                 | ResNet-12| Metric | 67.03 ± 0.79 | 83.68 ± 0.65 |

Table 8: The mean accuracies (%) with a 95% confidence interval on the CUB dataset.

| Method                        | Backbone | Type | 5-way 1-shot | 5-way 5-shot |
|-------------------------------|----------|------|--------------|--------------|
| MAML                          | Conv-64  |       | 55.92 ± 0.95 | 72.09 ± 0.76 |
| Matching Net                  | Conv-64  |       | 61.16 ± 0.89 | 72.86 ± 0.70 |
| ProtoNet                      | Conv-64  |       | 51.31 ± 0.91 | 70.77 ± 0.69 |
| RelationNet                   | Conv-64  |       | 62.45 ± 0.98 | 76.11 ± 0.69 |
| Baseline++                    | Conv-64  |       | 60.53 ± 0.83 | 79.34 ± 0.61 |
| SAML [10]                     | Conv-64  |       | 69.35 ± 0.22 | 81.37 ± 0.15 |
| DN4 [24]                      | Conv-64  |       | 53.15 ± 0.84 | 81.90 ± 0.60 |
| AM3(ProtoNet)                 | Conv-64  |       | 57.26 ± 0.66 | 71.34 ± 0.93 |
| AM3(ProtoNet) [53]            | ResNet-12|       | 73.6         | 79.9         |
| Ours(ProtoNet)                | Conv-64  |       | 69.79 ± 0.73 | 83.06 ± 0.66 |
| Ours(AM3,ProtoNet)            | Conv-64  |       | 72.14 ± 0.68 | 83.14 ± 0.69 |
| Ours(ProtoNet)                | ResNet-12|       | 76.58 ± 0.82 | 87.11 ± 0.71 |
| Ours(AM3,ProtoNet)            | ResNet-12|       | 77.03 ± 0.85 | 87.20 ± 0.70 |

published results from the literature, in the case of miniImageNet, we show results of our model with ResNet-12 as the backbone, where possible (i.e. unless no published results are available for ResNet-12). The results of the baselines in Table 7 (miniImageNet) are obtained from [22], [55], [38], [14] and [56]. The results for the
baselines in Table 8 (CUB) are obtained from [10], [24] and [53]. These results are based on the Conv-64 and ResNet-12 backbone, which we therefore adopt as well for this dataset. The results for tieredImageNet in Table 9 primarily rely on ResNet-12 as backbone, where the baseline results have been obtained from [55], [53], [14] and [46]. Apart from changes to the backbone network, we also vary the base method that is used as the visual classification component of our model. We have used ProtoNet, AM3 (with ProtoNet and GloVe vectors), FEAT and DeepEMD for this purpose. The results in Table 7 show that when ProtoNet is used as the base model, our method substantially outperforms the standard ProtoNet model, with the accuracy increasing from 56.52 to 64.53 in the 1-shot setting and from 74.28 to 79.37 in the 5-shot setting. Similarly, when using AM3, FEAT and DeepEMD as the base model, the results improve on the standard AM3, FEAT and DeepEMD models, respectively. The versions of our model with AM3 and DeepEMD also achieve the best overall results for the 1-shot and 5-shot settings respectively. The results for CUB in Table 8 again show that our model is able to substantially outperform the standard ProtoNet model. We also find that our model outperforms AM3, with the best results obtained when combining our model with AM3. In addition to the Conv-64 backbone, we have also included results with ResNet-12 for our model and AM3, which confirm these conclusions. Finally, for the tieredImageNet results in Table 9, we again see that our method consistently leads to improvements of the base model. In particular, this is shown for four different choices of the base model: ProtoNet, AM3, FEAT and DeepDEM. The version of our model that is based on DeepEMD leads to the best results overall.

6 CONCLUSIONS

We have proposed a method to improve the performance of metric-based FSL approaches by taking class names into account. Experiments on three datasets show that our method consistently improves the results of existing metric-based models. Moreover, our method is conceptually simple and can easily be added to a wide range of (existing and future) FSL models. An important advantage compared to previous work on exploiting class name embeddings, such as the AM3 method, is that we do not have to increase the number of parameters of the classification model. This has allowed us to exploit higher-dimensional class name embeddings. In particular, we have used class name embeddings that were learned using the BERT masked language model, as well as concatenations that combine different types of embeddings. From a technical point of view, our approach relies on two key insights. First, we found that decoupling the visual and textual prototypes is essential to achieving good results. Second, to avoid the introduction of new parameters, we rely on variants of canonical correlation analysis to align class name embeddings with the corresponding visual prototypes.

ACKNOWLEDGMENTS

This research was supported in part by the National Key R&D Program of China (2017YFB1200700); Capital Health Development Scientific Research Project (Grant 2020-1-4093); Clinical Medicine Plus X - Young Scholars Project, Peking University; the Fundamental Research Funds for the Central Universities; Global Challenges Research Fund (GCRF) grant (Essex reference number: GCRF G004); HPC resources from GENCI-IDRIS (Grant 2021-[AD011012273] and ANR CHAIRE IA BE4musIA.

Table 9: The mean accuracies (%) with a 95% confidence interval on the tieredImageNet dataset.

| Method             | Backbone | 5-way 1-shot | 5-way 5-shot |
|--------------------|----------|--------------|--------------|
| ProtoNet           | ResNet-12| 53.31 ± 0.89 | 72.69 ± 0.74 |
| RelationNet        | ResNet-12| 54.48 ± 0.93 | 71.32 ± 0.78 |
| MetaOptNet         | ResNet-12| 65.99 ± 0.72 | 81.56 ± 0.63 |
| CMT                | ResNet-18| 68.41 ± 0.39 | 84.28 ± 1.73 |
| SimpleShot         | ResNet-18| 69.09 ± 0.22 | 84.58 ± 0.16 |
| AM3(ProtoNet)      | ResNet-12| 58.53 ± 0.46 | 72.92 ± 0.68 |
| AM3(ProtoNet++)    | ResNet-12| 67.23 ± 0.34 | 78.95 ± 0.22 |
| CAN                | ResNet-12| 69.89 ± 0.51 | 84.23 ± 0.37 |
| FEAT               | ResNet-12| 70.80 ± 0.23 | 84.79 ± 0.16 |
| DeepEMD            | ResNet-12| 71.16 ± 0.87 | 86.03 ± 0.58 |
| Rethinking [46]    | ResNet-12| 71.52 ± 0.69 | 86.03 ± 0.49 |
| Ours(ProtoNet)     | ResNet-12| 66.82 ± 0.65 | 78.97 ± 0.53 |
| Ours(AM3,ProtoNet) | ResNet-12| 67.22 ± 0.43 | 79.08 ± 0.58 |
| Ours(FEAT)         | ResNet-12| 72.31 ± 0.68 | 85.76 ± 0.36 |
| Ours(DeepEMD)      | ResNet-12| 73.76 ± 0.72 | 87.51 ± 0.75 |

REFERENCES

[1] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In Proc. AAAI. 5012–5019.
[2] Piotr Bojanowski, Édouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. Transactions of the Association of Computational Linguistics. 5.1 (2017). 135–146.
[3] Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. 2018. Zero-Shot Visual Recognition Using Semantics-Preserving Adversarial Embedding Networks. In Proc. CVPR. 1043–1052.
[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A Closer Look at Few-shot Classification. In Proc. ICLR.
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In Proc. CVPR. 248–255.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proc. NAACL-HLT.
[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proc. ICLR. 1126–1135.
[8] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffery Dean, Marc’Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In Proc. NIPS. 2121–2129.
[9] Spyros Gidaris and Nikos Komodakis. 2018. Dynamic Few-Shot Visual Learning Without Forgetting. In Proc. CVPR. 4367–4375.
[10] Fuzheng Hao, Fengxian He, Jun Cheng, Lei Wang, Jianzhong Cao, and Dacheng Tao. 2019. Collect and Select: Semantic Alignment Metric Learning for Few-Shot Learning. In Proceedings of the IEEE International Conference on Computer Vision. 8460–8469.
[11] Bharath Hariharan and Ross Girshick. 2017. Low-shot visual recognition by shrinking and hallucinating features. In Proc. ICCV. 3018–3027.
[12] Han He and Jinhuo Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with BERT. In Proc. FLAIRS.
[13] Kuiming He, Xiangyu Zhang, Shaqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proc. CVPR. 770–778.
[14] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. 2019. Cross Attention Network for Few-shot Classification. In Proc. NeurIPS. 4085–4096.
[15] Guo Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In Proc. CVPR. 4700–4708.
[16] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proc. ICLR. 448–456.
A APPENDIX

We now explain in more detail how the matrices A and B are constructed. Let $X_0$ and $Y_0$ be the matrices whose $i^{th}$ row is, respectively, the class name embedding and the visual prototype of the $i^{th}$ class. The visual prototypes in $Y_0$ are estimated by averaging the visual features $f_0(x)$ of all images $x$ from the training set that belong to the $i^{th}$ class. These visual prototypes thus differ from those that are used for training the main model, as they are estimated from the full training set, rather than from a sampled episode.

As pointed out by [1], we can think of alignment methods such asCCA as performing a sequence of linear transformation steps. In particular, to find the matrices $A$ and $B$, we can use the following steps. The first transformation, called whitening, ensures that the individual components of the vectors have unit variance and are uncorrelated:

$$X_1 = X_0 A_1$$
$$Y_1 = Y_0 B_1$$

where

$$A_1 = (X_0^T X_0)^{-1}$$
$$B_1 = (Y_0^T Y_0)^{-1}$$

The second transformation maps the two embedding spaces onto a shared space using two orthogonal transformations $A_2$ and $B_2$. In particular, let us write the singular value decomposition of $X_1^T Y_1$ as $A_2 S_2 B_2^T$.

The final step is dimensionality reduction. Let $A_4$ be the $m_4 \times d$ matrix whose $i^{th}$ row has a 1 in the $i^{th}$ column and 0s everywhere else, and similar for $B_4$. 

$$B_4 = B_2^T B_1^{-1} B_3$$
In summary, the transformation $A$ of the class name embedding space is given by $A = A_1A_2A_3A_4 = A_2A_4$ if de-whitening is used and by $A = A_1A_2A_4$ if standard CCA is used. Similarly, the transformation $B$ of visual prototype space is given by $B = B_1B_2B_3B_4 = B_2B_4$ if de-whitening is used and by $B = B_1B_2B_4$ otherwise.