Shoot less and Sketch more: An Efficient Sketch Classification Via Joining Graph Neural Networks and Few-shot Learning

Asma Bensalah, Pau Riba, Alicia Fornés, Josep Lladós
Computer Vision Center, Computer Science Department
Universitat Autònoma de Barcelona
Barcelona, Spain
{*abensalah,priba,afornes,josep*}*@cvc.uab.es*

***Abstract*—With the emergence of the touchpad devices and drawing tablets, a new era of sketching started afresh. However, the recognition of sketches is still a tough task due to the variability of the drawing styles. Moreover, in some application scenarios there is few labelled data available for training, which imposes a limitation for deep learning architectures. In addition, in many cases there is a need to generate models able to adapt to new classes. In order to cope with these limitations, we propose a method based on few-shot learning and graph neural networks for classifying sketches aiming for an efficient neural model. We test our approach with several databases of sketches, showing promising results.**

***Keywords*-Sketch classification; Convolutional Neural Network; Graph Neural Network; Few-shot learning;**

I. Introduction

One of the first illustrations of human critical thinking was sketching. Sketching allows humans to graphically communicate ideas with a high degree of conceptualization and personalization. With the pervasive use of the touchscreen devices and drawing tablets, a new era of sketching started afresh. Sketches involve a communication language taking into account static (shape) and dynamic (motion, pressure) features. Sketch understanding has been therefore one of the most important areas of Graphics Recognition. The automatic understanding of sketched drawings involves the recognition of the composing symbols and their relationships. Depending on the context, this recognition consists of parsing the sketches in terms of the basic composition symbols and the rules that define a diagrammatic language. There is a large range of applications involving sketch recognition: technical design (sketching diagrams), ideation tools (mind mapping), cross-modal search (sketch-based image retrieval), different mobile apps involving doodling experiences (graphical passwords), etc.

The automatic recognition of sketched symbols is a tough task. First of all, due to their conceptual nature that make users to draw ideas at different levels of abstraction. There may be a high intra-class variability, ranging from amateur sketches to photo-realistic ones. This is also due to a semantic association between the class concept and the sketch that a user draws. For example, the concept *cat* can be drawn with different poses, levels of sophistication, fidelity to the realistic shape... This semantic gap is especially noticeable in cross-domain applications, like sketch-based image retrieval, where the similarity between sketches and images can not be solved exclusively in terms of visual shape features. A second difficulty is the scarcity of available labeled data. In supervised machine learning approaches that require training data, sketches have to be drawn one by one by human being. Some doodling platforms like *Google Quick, Draw!* [7] have allowed to compile a database of amateur sketches carefully sourced from internet users. The database, consisting of 3,000 sketches per category (corresponding to real objects) is available for training and testing sketch classifiers. However, in some domains the amount of labeled data is barely available, e.g. in technical domains like engineering diagrams, architectural drawings, flowcharts, etc. The difficulty in obtaining annotated data is also evident in those applications that require adaptation to the users style. In this case, the annotated data must be provided by the same user, or at least start from generic databases and perform a fine-grained adaptation.

When few labeled samples are given, an adequate strategy is the Few-shot Learning (FSL). The FSL paradigm in particular the one-shot is inspired from the fact that when humans have to deal with unprecedented experiences, the link between cause and effect is quickly learnt in order to survive. This way of learning is not similar to the incremental learning, where knowledge is gained gradually through trial and error [18]. Alternatively stated, FSL is a learning approach in which the classifier must adapt to acclimate to unseen classes in training, with the constraint that there are only very few annotated examples of these classes [18]. This technique has been successfully applied to object classification [15], [14].

In the FSL training scenario, which is a particular realization of transductive learning, instead of providing to the system a large set of samples and their associated images, it is fed by a collection of images and their associated label similarity. Thus, training and test data is provided beforehand, but instead of inferring a predictive model from the training data, the system leverages the unlabeled data

and the relations between labeled and unlabeled data to predict the labels of the test (unlabeled) data. The main drawback of the few-shot learning is that, since a model is not inferred, every time a new set of instances must be classified, the whole training must be re-done again. In the context of sketches, suppose the system receives a set of sketched symbols. Some of these samples are labeled with their corresponding class, but some of them are not. The aimed output is to associate the corresponding labels to the unlabeled sketches. The prediction is based on the observation of the similarities between labeled and unlabeled data.

In this work, we propose a few shot learning approach for sketch recognition. Based on the architecture proposed by Garcia and Bruna [16] we propose an architecture based on graph representations. Graph nodes are associated with the sketches, and edges represent similarity relations. Using a supervised message-passing formalism [6] the model is trained in an end-to-end manner using graph neural networks. The model propagates the node information, casting a posterior inference over the graph determined by the input sketches and labels. We contribute with several improvements regarding the baseline method. First, we incorporate edge classification in our model. Second, we remove the hot-encoding representation. We also explore the creation of master nodes (i.e. prototypes) to speed up the system in large datasets. Master nodes can be seen as joint node embeddings that contract the information of a subgraph in a kind of hyperedge. In addition, from the application point of view, we adapt the approach to the particular problem of sketched symbol recognition.

The rest of the paper is organized as follows. Section II reviews related works in the deep learning literature. Section III describes the baseline model, whereas Section IV exhibits the intended architecture improvements. Section V describes the exploited datasets, the metrics and discusses the results. Finally, Section VI draws the conclusions and suggests future research lines.

II. Related Work

In this section, we provide information about the most germane works in the deep learning literature to our work.

In the leading work for Few Shot Learning [13], a Bayesian one-shot algorithm was put forward in the behalf of recognition purposes. Conversely to intuition, it was proven that interesting features of a novel object category could be learned from one (or few) training example.

Let's consider a large dataset of labeled instances, with a number of classes denoted C_{train}, the key objective of an FSL approach, is to build classifiers on the testing set with a disjoint of new classes C_{test}, albeit the fact that only a small labeled support set will be available [8]. When the model is given a support set including K examples from N classes, then it is solicited to classify some query examples into the N classes, this is a K shots-N ways learning scenario [20].

Many advances have been made in Few-shot learning in order to improve the already existent models. One of them is Matching Networks [19], where attention mechanism are applied over the embedding space of the support set according to the test sample. With this definition, the output for a new class is characterized by a linear combination of the labels in the support set. Every single episode is destined to imitate a few-shot task.

In addition, Prototypical Network [18] are founded on the concept that points cluster around one prototype representation. It involves computing a prototype c_k for each present class in the support set S_k. The prototype is taken to be the mean of the learned embedded elements $\phi(x_i)$ of the class:

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, x_j) \in S_k} f_\phi\Big(x_i\Big), \tag{1}$$

where f_ϕ is the embedding function.

Later, the classification is accomplished by finding the nearest class prototype for an embedded query point. The approach has proven to be simpler and more efficient then prior meta-learning approaches.

In [16], a framework that addresses the few-shot paradigm by means of Graph Convolutional Networks (GNN) is presented. The proposed architecture extrapolates a number of prior few-shot architectures namely the Siamese Networks [11], the Matching Networks [18] and the Prototypical Networks [18]. Over and above, by dint of the graph formulation it is conceivable to have different training setups within the same framework. As a result, one learner can adapt to different learning schemes such as semi-supervised learning and active learning paradigms.

Initially, Graph Neural Networks (GNN) were proposed in [1]. The proprieties of the convolutions were used in the Fourier domain. The authors demonstrated that learning convolutional layers with a certain number of parameters is unrelated to the size of the inputs, when it regards low-dimensional graphs.

Later, in [10], for the sake of overcoming the high cost of computing an eigen-decomposition, Chebyshev polynomial was applied to approximate the linear graph convolution acting on a certain node and generate the localized filters in an effective manner. Hence, all the methods described above are solely able to be trained on a single graph and cannot be instantly used to a set of graphs with dissimilar structures, by virtue of their dependence Laplacian eigenbasis which is associated to a fixed graph structure [17].

III. Baseline

In this section we describe the few shot learning architecture that will be used as the baseline of our model.

A. Baseline Architecture

Garcia and Bruna introduced in [16] a classification method based on an end-to-end graph neural network architecture inferred from beforehand models. The aim of this few shot learning model was to propagate label information from labeled instances to the unlabeled query, all of them belonging to a set of images denoted $\mathcal{T}$.

The input $\mathcal{T}$ is associated to a graph $\mathcal{G}_{\mathcal{T}} = (\mathcal{V}, \mathcal{E})$. In this graph, the nodes of the graph are initialized by an embedding of the original images $x_i \in \mathcal{T}$. Afterwards, the learned image embedding is concatenated with a label encoding:

$$x_i^{(0)} = (\varphi(x_i), h(l_i)), \tag{2}$$

where φ is a Convolutional neural network and $h(l_i) \in \mathbb{R}_+^K$ is the one-hot encoding of the label if the label is known, and a uniform probability distribution otherwise.

Finally, they make use of a GNN convolutions to obtain a final node classification. A GNN convolution layer Gc is defined as:

$$x_l^{k+1} = Gc(x^k) = \rho\left(\sum_{B \in \mathcal{A}} B x^{(k)} \theta_{B,l}^{(k)}\right), l = d_1 .....d_{k+1}, \tag{3}$$

where $\Theta = \left\{\theta_k^1, ..., \theta_k^{|A|}\right\}_k, \theta_k^B \in \mathbb{R}^{d_k \times d_{k+1}}$, are trainable parameters and $\rho(.)$ is a point-wise non-linearity. Moreover, $\mathcal{A}$ is a set of local graph linear operators which in its simple form can be composed of the *adjacency operator* or powers of the adjacency matrix.

The priorly mentioned operators are learnt by means of extracting edge features. Hence, the edge learning process is performed as fellows:

$$\tilde{A}_{i,j}^{(k)} = \varphi_{\tilde{\theta}}\left(x_i^{(k)}, x_j^{(k)}\right) \tag{4}$$

where φ is a symmetric learnable function. In their work, it is defined as:

$$\varphi(x_i^{(k)}, x_j^{(k)}) = MLP_{\tilde{\theta}}(abs(x_i^{(k)} - x_j^{(k)})) \tag{5}$$

where MLP stands for a Multi Layer Perceptron. Finally, the family of operations is defined $\mathcal{A} = \{\tilde{A}^{(k)}, 1\}$. The training adjacency is normalized to a stochastic kernel by applying a softmax along each row.

B. Limitations

Although this architecture has many advantages, it has the following limitations:

Node Classification: The one-hot encoding will arbitrary change according to a given support set, hence, it will become hard to set it as a prediction goal. In other words, the prediction depends on the labelling of the support set and it can easily differ from one support set to another while the query is still the same.

One-hot Encoding: This encoding does not bring out the similarities between the classes. This point jointly with the node classification will fix the number of classes that we may use in our support set.

Scalability: The size of the GNN would easily explode in certain applications. In the current setting, an increase on the number of classes and ways will lead to generate large graphs that will be difficult to use. Hence, we could seek the optimization of the GNN in order to address Few-shot problems at a larger scale when specially the number of classes may increase a lot.

Next, we describe our proposed architecture, with the aim to palliate with the above pointed challenges.

IV. Method

As stated before, in this section we describe our proposal, which aims to alleviate the limitations of the baseline model, and also, to adapt it to the task of sketch classification. An overview of the architecture is shown in Figure 1.

Basically, our improvements are the following: the incorporation of the edge classification, the removal of the one-hot encoding, and finally, the exploration of the use of master nodes. These improvements are described next.

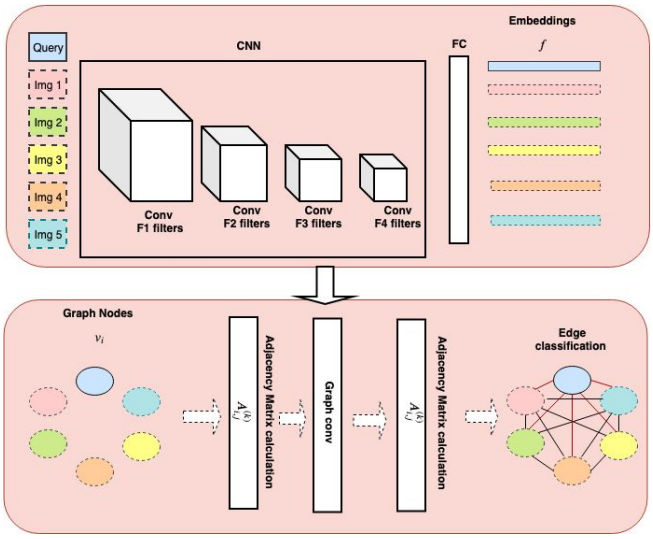

Figure 1. Model architecture.

A. Edge classification

Many modern works apropos of graph convolutional neural networks explore the use of the information associated in the edges of the graph. In [17] attention weights and node features are learned simultaneously. For every bond attribute, the binary adjacency matrix is switched to a real-valued attention matrix. In an other work [2] the framework consists of an encoder and a decoder. The encoder takes an incomplete graph as an input, and it generates an embedding for all the nodes. Thereupon, the decoder predicts the missing edges using the node embedding.

To address the limitation concerning node classification described before, we aim to avail of the information available in the edges, because the stochastic normalized adjacency matrix has nice properties that can make the use of the edges beneficial.

Thus, in our approach, the model is requested to predict the respective weights for a query image. The labels are $1 \times K \times N$ vector (with 1 whenever the query and the node belong to the same class). As a consequence, the number of model's learnable parameters is less than the one in [2], owing to the deletion of last convolutional network, which is not reckoning with edge classification. Contrary to the baseline method, we are not applying a softmax along each row of the adjacency matrix.

B. Hot-encoding removal

In the baseline model, a one-hot encoding is concatenated to the ensuing embedding of the CNN, at the time of constructing the initial node features of the graph. In this section, we want to asses the need of using the one-hot encoding in our model. For this reason, we omit concatenating the one-hot encoding to the embeddings before feeding them to the GCN.

The employed one-hot-encoding does not provide any varying similarity between the classes. Furthermore, the manner the classes are arbitrarily elected prior to each training experiment implies that the exact same class can have distinct one-hot-encoding labeling at every single training experiment. To cope with this limitation, we opt to remove the hot-encoding when forming the initial graph's node, assuming that we are always using the Edge classification outlined in IV-A. Ergo, equation 2 becomes:

$$x_i^{(0)} = \Big(\phi(x_i)\Big) \tag{6}$$

Thereby, the size of the embedding of the image is less compared to the one in the baseline architecture.

C. Master Node

Exploring large datasets with this model is feasible. However, enlarging the number of ways or shots in an FSL regime will lead to an exponential increase in terms of the nodes' number and graph's size that may not fit in the memory. Many techniques were introduced in the field of graph reduction to find a substitute for a large graph that preserves the basic properties of the original graph and is less complex with regards to the number of edges/nodes, including clustering and segmentation exempli gratia [12] [3].

To cope with the scalability limitation of the model, we designate a representative for each class of the support set. Premised on the concept of Prototypical Networks [18] and the desire to optimize the size of the graph when the number of ways and shots is big. So, rather than forming the Graph from all the embeddings of all the images in the support set, the graph is created from the prototypes of the classes solely. We consider the prototype of a class to be the mean as in equation 1.

As a result of using a reduced graph instead of the whole graph as in the baseline architecture, the size of the GNN's input decreases drastically by $\frac{1}{shots}$, in few shot scenarios.

D. Training

Our proposed approach is trained in a supervised manner, so we know whether or not a pair of images belong to the same class. All the models were trained using the *Adam* [9] optimizer with weight decay *i.e.* L_2 regularization and an initial learning rate of 0.01.

The use of edge classification in our model implies attempting to solve a multi-binary classification problem. This means that the question is "Does the query belong to the same class as the image in the support set?" instead of "To which class does the query belong?" like in [16]. We consider the label of each example as being a binary vector $l_1.....l_{N\times K}$, where $l_j = 1$ if the query and the image of the support class in $node_j$ in the graph belong to the same class.

We consider a binary cross-entropy loss between the learned adjacency matrix and the label vector.

V. Experimental Results

In order to assess our architecture, we use an evaluation approach akin to the one in [16].We conduct different q-shot, K-way experiments. Concretely, we evaluate our architecture on a sketch classification scenario, using the TU-Berlin and the Mini-QuickDraw sketch datasets. Next, we describe the datasets, the metrics, and analyze the results.

A. Datasets

TUBerlin: The TU-Berlin dataset [5] is formed by 20,000 unique hand-drawn sketches equally partitioned into 250 object categories. At first introduced within a quest to assess the human recognition [5], it includes the most common objects we could stumble upon in everyday life. The objects are designed to be recognizable devoid of the context and are sufficiently specific to have very few visual representations. In the experiments, we use 161 seen classes for training, 41 for validation and 48 unseen classes for testing. Some examples are shown in Figure 2.

Mini-QuickDraw: We introduce Mini-QuickDraw, a subset of the *Quick, Draw!* dataset[1] [4]. We have selected a subset because we have removed very simple classes such as zigzag and line, and also because the original dataset is too large and not necessary to test the performance of a FSL method.

The *Quick, Draw!* dataset contains 50 million drawings (doodles), denoted by a metadata. Our Mini-QuickDraw is

[1]https://github.com/googlecreativelab/quickdraw-dataset

Table I
TEST ACCURACIES (SHOWN IN PERCENTAGES) FOR 5WAYS SCENARIO: 1-SHOT AND 5-SHOT ON TU-BERLIN AND MINI-QUICKDRAW DATASETS. WE ALSO PROVIDE THE TRAINING TIME OF EACH MODEL (MINUTES AND SECONDS).

Model	TU-Berlin				Mini-QuickDraw			
	1-shot		5-shot		1-shot		5-shot	
	accuracy	time	accuracy	time	accuracy	time	accuracy	time
Baseline [16]	73.10%	673m58.122s	90.42%	709m12.185s	**59.47%**	446m47.035s	**86.30%**	860m3.924s
Edge classification	**77.44%**	362m18.529s	**90.72%**	799m45.579s	58.72%	501m8.989s	76.32%	799m49.934s
Hot-encoding removal	73.79%	1110m30.559s	87.34%	818m42.496s	59.16%	495m44.017s	73.67%	823m7.613s
Master node	-	-	48.08%	780m17.290s	-	-	41.14%	796m43.326s

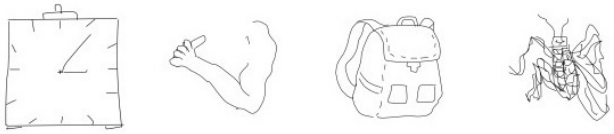

Figure 2. Examples from the TU-Berlin dataset.

composed of 32,700 drawings uniformly distributed in 109 object categories. We consider 70 seen classes for training, 18 classes for validation and 21 classes for testing. Some examples are shown in Figure 3.

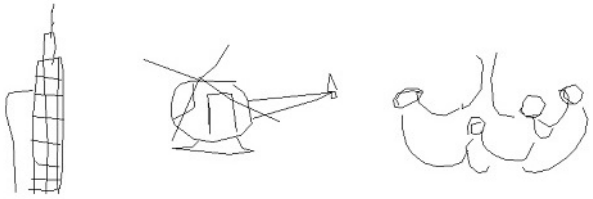

Figure 3. Examples from the Mini-QuickDraw dataset.

B. Metrics

In order to evaluate the performance of our approach, we have compared the different models with the baseline architecture [16]. The comparison is carried out in fair setting. To measure the effectiveness of our model, we report the mean accuracy on the test set over several experiments.

We also provide the training time, the number of trainable parameters and the GNN input size (in Kilobits) for each one of the proposed models.

C. Results

For all our experiments we have used an embedding network of four convolutional layers followed by a fully connected one. We train and test our model using similar values for the number of ways N and the same number of shots S. In other words, the number of ways and shots used in training is the same as the ones used in testing (e.g. 5Way-5shot for train and test).

Table I shows the accuracy and training times on both datasets, whereas Table II shows the number of parameters and the input size for the different versions of the model.

Table II
NUMBER OF PARAMETERS (5WAY 1 SHOT) AND INPUT SIZE (5WAY 5SHOT) FOR EACH MODEL.

Model	N.parameters (5W-1s)	Input size (5W-5s)
Baseline [16]	335994	345.8K
Edge classification	333699	345.8K
Hot-encoding removal	333699	332.8K
Master node	333699	79.8K

Concerning the Edge classification version, we perceive a moderate increase in terms of accuracy in the TU-Berlin experiments. Moreover, in the 5-Way 1-shot there is a significant decrease in the training time (instead of 673 minutes, it trains in 362 minutes). In Table II we can observe a slight reduction by 0.6% (5Way-1shot) in the number of trainable parameters due to the deletion of the last convolutional layer.

Regarding the hot-encoding removal version, we discern a small change in the accuracy, positive in the case of TU-Berlin by 0,72% and in a form of a slight drop in the Mini-Quickdraw case (from 59.47% to 59.16%) in the 5-Way 1-shot. Along, with a rational slip in the input size by 3,32% (5Way-5shots). Additionally, Table I projects an increase in the training time compared to the baseline model.

Finally, anent the master node version, we observe that the performance significantly drops. This behaviour is expected because, instead of using all the instances in a class of the support class, the few-shot task is conducted with only one prototype for representing the class. On the flip side, the input size experiences a free-fall of 76% (5Way-5shots). At the same time, the training time goes down in the case of Mini-QuickDraw.

In summary, we have observed that the edge classification version is reliable since it either outperforms the state of the art as in TU-Berlin or experiences a tiny slip as in Mini-Quickdraw, in terms of accuracy. Also, it occupies less memory space, due to the decline in trainable parameters' number. Admittedly, hot-encoding does not play a major role when relying on edge classification, because, compared to the state of the art, we obtained a very similar accuracy but

with a slight increase in the training time.

Finally, the master node approach seems to be adequate for large ways and shots values in an FSL scenario, since it significantly reduces the size of the input, but at the cost of a significant performance decrease.

VI. Conclusion

In this paper we have proposed a sketch classification method based on graph neural networks. The problem has been framed as a few-shot learning task where a small support set of sketches is fed to the model. We conducted a series of experiments on two datasets: TU-Berlin and the Mini-Quickdraw. We have shown that the different improvements over the baseline can provide an increase in the performance or a significant decrease in training times and model size.

As a future work, we would like to explore the incorporation of spatial attention mechanisms to the architecture in order to enhance the query features, as well as changing the network embedding architecture and enlarging the graph neural network.

Acknowledgment

This work has been partially supported by the Spanish project RTI2018-095645-B-C21, the grant FPU15/06264 from the Spanish Ministerio de Educación, Cultura y Deporte, the Ramon y Cajal Fellowship RYC-2014-16831 and the CERCA Program/ Generalitat de Catalunya. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, Apr 2014.

[2] Guillem Cucurull, Perouz Taslakian, and David Vazquez. Context-aware visual compatibility prediction. In *CVPR*, Jun 2019.

[3] Matthias Dehmer, editor. *Structural Analysis of Complex Networks*. Birkhäuser / Springer, 2011.

[4] Sounak Dey, Pau Riba, Anjan Dutta, Josep Llads, and Yi-Zhe Song. Doodle to search: Practical zero-shot sketch-based image retrieval. *CVPR*, 2019.

[5] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4):44:1–44:10, 2012.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272, Aug 2017.

[7] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick, draw! - a.i. experiment. *https://quickdraw.withgoogle.com*, 2016.

[8] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, Jun 2019.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, May 2015.

[10] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, Apr 2017.

[11] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[12] Nicolas Lermé, François Malgouyres, and Lucas Létocart. Reducing graphs in graph cut segmentation. In *ICIP*, pages 3045–3048, Sep 2010.

[13] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.

[14] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. *CoRR*, abs/1904.05160, 2019.

[15] Minseop Park, Jungtaek Kim, Saehoon Kim, Yanbin Liu, and Seungjin Choi. Mxml: Mixture of meta-learners for few-shot classification. *CoRR*, abs/1904.05658, 2019.

[16] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *ICLR*, Apr 2018.

[17] Chao Shang, Qinqing Liu, Ko-Shin Chen, Jiangwen Sun, Jin Lu, Jinfeng Yi, and Jinbo Bi. Edge attention-based multi-relational graph convolutional networks. Feb 2018.

[18] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090, Dec 2017.

[19] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, Dec 2016.

[20] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, pages 321–328, 2003.