Learning to Model the Tail

Yu-Xiong Wang **Deva Ramanan** **Martial Hebert**
Robotics Institute, Carnegie Mellon University
{yuxiongw, dramanan, hebert}@cs.cmu.edu

Abstract

We describe an approach to learning from long-tailed, imbalanced datasets that are prevalent in real-world settings. Here, the challenge is to learn accurate "few-shot" models for classes in the tail of the class distribution, for which little data is available. We cast this problem as transfer learning, where knowledge from the data-rich classes in the head of the distribution is transferred to the data-poor classes in the tail. Our key insights are as follows. First, we propose to transfer *meta*-knowledge about *learning-to-learn* from the head classes. This knowledge is encoded with a meta-network that operates on the space of model parameters, that is trained to predict many-shot model parameters from few-shot model parameters. Second, we transfer this meta-knowledge in a *progressive* manner, from classes in the head to the "body", and from the "body" to the tail. That is, we transfer knowledge in a gradual fashion, regularizing meta-networks for few-shot regression with those trained with more training data. This allows our final network to capture a notion of *model dynamics*, that predicts how model parameters are likely to change as more training data is gradually added. We demonstrate results on image classification datasets (SUN, Places, and ImageNet) tuned for the long-tailed setting, that significantly outperform common heuristics, such as data resampling or reweighting.

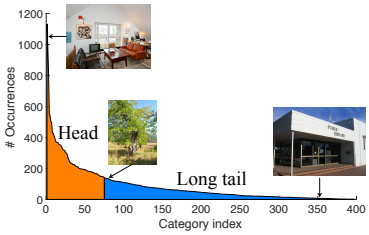
1 Motivation

Deep convolutional neural networks (CNNs) have revolutionized the landscape of visual recognition, through the ability to learn "big models" with hundreds of millions of parameters [1, 2, 3, 4]. Such models are typically learned with *artificially balanced* datasets [5, 6, 7], in which objects of different classes have approximately evenly distributed, very large number of human-annotated images. In real-world applications, however, visual phenomena follow a long-tailed distribution as shown in Fig. 1, in which the number of training examples per class varies significantly from hundreds or thousands for head classes to as few as one for tail classes [8, 9, 10].

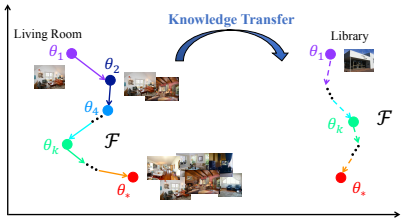
Long-tail: Minimizing the skewed distribution by collecting more tail examples is a notoriously difficult task when constructing datasets [11, 6, 12, 10]. Even those datasets that are balanced along one dimension still tend to be imbalanced in others [13]; *e.g.*, balanced scene datasets still contain long-tail sets of objects [14] or scene subclasses [8]. This *intrinsic* long-tail property poses a multitude of open challenges for recognition in the wild [15], since the models will be largely dominated by those few head classes while degraded for many other tail classes. Rebalancing training data [16, 17] is the most widespread state-of-the-art solution, but this is *heuristic and suboptimal* — it merely generates redundant data through over-sampling or loses critical information through under-sampling.

Head-to-tail knowledge transfer: An attractive alternative is to *transfer* knowledge from data-rich head classes to data-poor tail classes. While transfer learning [18, 19, 20] from a source to target task is a well studied problem [18, 21], by far the most common approach is fine-tuning a model pre-trained on the source task [22]. In the long-tailed setting, this fails to provide any noticeable improvement since pre-training on the head is quite similar to training on the unbalanced long-tailed dataset (which is dominated by the head) [10].

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



(a) Long-tail distribution on the SUN-397 dataset.



(b) Knowledge transfer from head to tail classes.

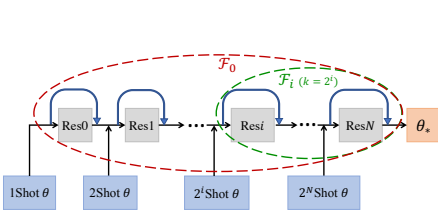
Figure 1: Head-to-tail knowledge transfer in model space for long-tail recognition. Fig. 1a shows the number of examples by scene class on SUN-397 [14], a representative dataset that follows an intrinsic long-tailed distribution. In Fig. 1b, from the data-rich head classes (*e.g.*, living rooms), we introduce a meta-learner $\mathcal{F}$ to learn the model dynamics — a series of transformations (denoted as solid lines) that represents how few k-shot models θ_k start from θ_1 and gradually evolve to the underlying many-shot models θ_* trained from large sets of samples. The model parameters θ are visualized as points in the "dual" model (parameter) space. We leverage the model dynamics as prior knowledge to facilitate recognizing tail classes (*e.g.*, libraries) by hallucinating their model evolution trajectories (denoted as dashed lines).

Transferring meta-knowledge: Inspired by the recent work on meta-learning [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37], we instead transfer meta-level knowledge about *learning to learn* from the head classes. Specifically, we make use of the approach of [23], which describes a method for learning from small datasets (the "few-shot" learning problem [20, 38, 39, 40, 41, 42, 43, 44, 23, 45, 24, 46, 47, 48, 41, 49, 50, 51, 52, 53, 54]) through estimating a generic model transformation. To do so, [23] learns a meta-level network that operates on the space of model parameters, which is specifically trained to *regress* many-shot model parameters (trained on large datasets) from few-shot model parameters (trained on small datasets). Our meta-level regressor, which we call *MetaModelNet*, is trained on classes from the head and then applied to those from the tail. As an illustrative example in Fig. 1, consider learning scene classifiers on a long-tailed dataset with many living-rooms but few outside libraries. We learn both many-shot and few-shot living-room models (by subsampling the training data as needed), and train a regressor that maps between the two. We can then apply the regressor on few-shot models of libraries learned from the tail.

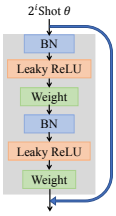
Progressive transfer: The above description suggests that we need to split up a long-tailed training set into a distinct set of source classes (the head) and target classes (the tail). This is most naturally done by thresholding the number of training examples per class. But what is the correct threshold? A high threshold might result in a meta-network that simply acts as an identity function, returning the input set of model parameters. This certainly would not be useful to apply on few-shot models. Similarly, a low threshold may not be useful when regressing from many-shot models. Instead, we propose a "continuous" strategy that builds multiple regressors across a (logarithmic) *range* of thresholds (*e.g.*, 1-shot, 2-shot, 4-shot regressors, *etc.*), corresponding to different head-tail splits. Importantly, these regressors can be efficiently implemented with a *single, chained* MetaModelNet that is naturally regularized with residual connections, such that the 2-shot regressor need only predict model parameters that are fed into the 4-shot regressor, and so on (until the many-shot regressor that defaults to the identity). By doing so, MetaModelNet encodes a trajectory over the space of model parameters that captures their evolution with increasing sample sizes, as shown in Fig. 1b. Interestingly, such a network is naturally trained in a *progressive* manner from the head towards the tail, effectively capturing the gradual dynamics of transferring meta-knowledge from data-rich to data-poor regimes.

Model dynamics: It is natural to ask what kind of dynamics are learned by MetaModelNet. How can one consistently predict how model parameters will change with more training data? We posit that the network learns to capture implicit *data augmentation*. For example, given a 1-shot model trained with a single image, the network may learn to implicitly add rotations of that single image. But rather than explicitly creating data, MetaModelNet predicts their impact on the learned model parameters.

Our contributions are three-fold. (1) We analyze the *dynamics* of how model parameters evolve when given access to more training examples. (2) We show that a single meta-network, based on deep residual learning, can learn to accurately predict such dynamics. (3) We train such a meta-network on long-tailed datasets through a recursive approach that gradually transfers meta-knowledge learned from the head to the tail, significantly improving long-tail recognition on a broad range of tasks.



(a) Learning a sample size dependent transformation.



(b) Structure of residual blocks.

Figure 2: MetaModelNet architecture for learning model dynamics. We instantiate MetaModelNet as a deep residual network with residual blocks $i = 0,\ 1, \ldots, N$ in Fig. 2a, which accepts few-shot model parameters θ (trained on small datasets across a logarithmic range of sample sizes k, $k = 2^i$) as (multiple) inputs and regresses them to many-shot model parameters θ_* (trained on large datasets) as output. The skip connections ensure the identity regularization. $\mathcal{F}_i$ denotes the meta-learner that transforms (regresses) k-shot θ to θ_*. Fig. 2b shows the structure of the residual blocks. Note that the meta-learners $\mathcal{F}_i$ for different k are derived from this single, chained meta-network, with nested circles (subnetworks) corresponding to $\mathcal{F}_i$.

2 Head-to-tail meta-knowledge transfer

Given a long-tail recognition task of interest and a base recognition model such as a deep CNN, our goal is to transfer knowledge from the data-rich head to the data-poor tail classes. As shown in Fig. 1, knowledge is represented as trajectories in model space that capture the evolution of parameters with more and more training examples. We train a meta-learner (MetaModelNet) to learn such model dynamics from the head classes, and then "hallucinate" the evolution of parameters for the tail classes. To simplify exposition, we first describe the approach for a fixed split of our training dataset into a head and tail. We then generalize the approach to multiple splits.

Fixed-size model transformations: Let us write H_t for the "head" training set of (x, y) data-label pairs constructed by assembling those classes for which there exists more than t training examples. We will use H_t to learn a meta-network thats maps few-shot model parameters to many-shot parameters, and then apply this network on few-shot models from the tail classes. To do so, we closely follow the model regression framework from [23], but introduce notation that will be useful later. Let us write a base learner as $g(x; \theta)$ as a feedforward function $g(\cdot)$ that processes an input sample x given parameters θ. We first learn a set of "optimal" model parameters θ_* by tuning g on H_t with a standard loss function. We also learn few-shot models by randomly sampling a smaller fixed number of examples per class from H_t. We then train a meta-network $\mathcal{F}(\cdot)$ to regress few-shot parameters to θ_*, where $\mathcal{F}(\cdot)$ is itself parameterized with weights w. We focus on parameters from the last fully-connected layer for a single class — *e.g.*, $\theta \in \mathbb{R}^{4096}$ for an AlexNet architecture. This allows us to learn regressors that are shared across classes (as in [23]), and so can be applied to any individual test class. The objective function for each class is:

$$\sum_{\theta \in k\text{Shot}(H_t)} \Big\{ ||\mathcal{F}(\theta; w) - \theta_*||^2 + \lambda \sum_{(x,y) \in H_t} \text{loss}\Big(g\big(x; \mathcal{F}(\theta; w)\big), y\Big) \Big\}. \tag{1}$$

The final loss is averaged over all the head classes and minimized with respect to w. Here, $k\text{shot}(H_t)$ is the set of few-shot models learned by subsampling H_t, and loss refers to the performance loss used to train the base network (*e.g.*, cross-entropy). [23] found that the performance loss was useful to learn regressors that maintained high accuracy on the base task.

Training: What should be the value of k, for the k-shot models being trained? One might be tempted to set $k = t$, but this implies that there will be some head classes near the cutoff that have only t training examples, implying θ and θ_* will be identical. To ensure that a meaningful mapping is learned, we set $k = t/2$.

Recursive residual transformations: We wish to apply the above module on all possible head-tail splits of a long-tailed training set. To do so, we extend the above approach in three crucial ways:

- (Sample-size dependency) Generate a sequence of different meta-learners $\mathcal{F}_i$ each tuned for a specific k, where $k = k(i)$ is an increasing function of i (that will be specified shortly). Through a straightforward extension, prior work on model regression [23] learns a single fixed meta-learner for all the k-shot regression tasks.

- (Identity regularization) Ensure that the meta-learner defaults to the identity function for large i: $\mathcal{F}_i \to \mathcal{I}$ as $i \to \infty$.
- (Compositionality) Compose meta-learners out of each other: $\forall i < j$, $\mathcal{F}_i(\theta) = \mathcal{F}_j\Big(\mathcal{F}_{ij}(\theta)\Big)$ where $\mathcal{F}_{ij}$ is the regressor that maps between $k(i)$-shot and $k(j)$-shot models.

Here we dropped the explicit dependence of $\mathcal{F}(\cdot)$ on w for notational simplicity. These observations emphasize the importance of (1) the identity regularization and (2) sample-size-dependent regressors for long-tailed model transfer. We operationalize these extensions with a recursive residual network:

$$\mathcal{F}_i(\theta) = \mathcal{F}_{i+1}\Big(\theta + f(\theta; w_i)\Big), \tag{2}$$

where f denotes a residual block parameterized by w_i and visualized in Fig. 2b. Inspired by [55, 23], f consists of batch normalization (BN) and leaky ReLU as pre-activation, followed by fully-connected weights. By construction, each residual block transforms an input $k(i)$-shot model to a $k(i+1)$-shot model. The final MetaModelNet can be efficiently implemented through a chained network of $N+1$ residual blocks, as shown in Fig. 2a. By feeding in a few-shot model at a particular block, we can derive any meta-learner $\mathcal{F}_i$ from the central underlying chain.

Training: Given the network structure defined above, we now describe an efficient method for training based on two insights. (1) The recursive definition of MetaModelNet suggests a recursive strategy for training. We begin with the *last* block and train it with the *largest* threshold (*e.g.*, those few classes in the head with many examples). The associated k-shot regressor should be easy to learn because it is similar to an identity mapping. Given the learned parameters for the last block, we then train the next-to-last block, and so on. (2) Inspired by the general observation that recognition performance improves *on a logarithmic scale* as the number of training samples increases [8, 9, 56], we discretize blocks accordingly, to be tuned for 1-shot, 2-shot, 4-shot, ... recognition. In terms of notation, we write the recursive training procedure as follows. We iterate over blocks i from N to 0, and for each i:

- Using Eqn. (1), train parameters of the residual block w_i on the head split H_t with k-shot model regression, where $k = 2^i$ and $t = 2k = 2^{i+1}$.

The above "back-to-front" training procedure works because whenever block i is trained, all subsequent blocks $(i+1, \ldots, N)$ have already been trained. In practice, rather than holding all subsequent blocks fixed, it is natural to fine-tune them while training block i. One approach might be fine-tuning them on the current $k = 2^i$-shot regression task being considered at iteration i. But because MetaModelNet will be applied across a wide range of k, we fine-tune blocks in a *multi-task* manner across the current viable range of $k = (2^i, 2^{i+1}, \ldots, 2^N)$ at each iteration i.

3 Experimental Evaluation

Evaluation and analysis on SUN-397: We first focus on fine-tuning the classifier module while freezing the representation module of a pre-trained ResNet152 CNN model [4] on long-tailed SUN-397 [14, 57] for its state-of-the-art performance. In addition to the "plain" baseline that fine-tunes on the target data following the standard practice, we compare against three state-of-the-art baselines that are widely used to address the imbalanced distributions. (1) Over-sampling [16, 17], which uses the balanced sampling via label shuffling as in [16, 17]. (2) Under-sampling [58], which reduces the number of samples per class to 47 at most (the median value). (3) Cost-sensitive [59], which introduces additional weights in the loss function for each class with inverse class frequency.

Table 1 summarizes the performance comparison averaged over all classes and Fig. 3 details the per class comparison. Table 1 shows that our MetaModelNet provides a promising way of encoding the shared structure across classes *in model space*. It outperforms existing approaches by a large margin. Fig. 3 shows that our approach significantly improves accuracy in the tail.

Ablation analysis: Table 2 shows that training for a fixed sample size and identity regularization provide a noticeable performance boost (2%). Adding multiple head-tail splits through recursion further improves accuracy by a small but noticeable amount (0.5% as shown in Table 2). Table 3 shows that progressively learning classifier dynamics while fine-tuning features performs the best when using ResNet50 [4].

Understanding model dynamics: Fig. 4 shows some empirical analysis of model dynamics.

Method	Plain [4]	Over-Sampling [16, 17]	Under-Sampling [58]	Cost-Sensitive [59]	MetaModelNet (Ours)
Acc (%)	48.03	52.61	51.72	52.37	**57.34**

Table 1: Performance comparison between our MetaModelNet and state-of-the-art approaches for long-tailed scene classification when fine-tuning the pre-trained ILSVRC ResNet152 on the SUN-397 dataset. We focus on learning the model dynamics of the classifier module while freezing the CNN representation module. By benefiting from the learned generic model dynamics from the head classes, ours significantly outperforms all the baselines for the long-tail recognition.

Method	Model Regression [23]	MetaModelNet+Fix Split (Ours)	MetaModelNet+ Recur Split (Ours)
Acc (%)	54.68	56.86	**57.34**

Table 2: Ablation analysis of variations of our MetaModelNet.

Scenario	Pre-Trained Features		Fine-Tuned Features (FT)		
Method	Plain [4]	MetaModelNet (Ours)	Plain [4]	Fix FT + MetaModelNet (Ours)	Recur FT + MetaModelNet (Ours)
Acc (%)	46.90	54.99	49.40	58.53	**58.74**

Table 3: Ablation analysis of joint feature fine-tuning and model dynamics learning.

Dataset	Places-205 [7]		ILSVRC-2012 [5]	
Method	Plain [1]	MetaModelNet (Ours)	Plain [1]	MetaModelNet (Ours)
Acc (%)	23.53	**30.71**	68.85	**73.46**

Table 4: Performance comparisons on large-scale scene-centric Places [7] and object-centric ImageNet [5] datasets, which are tuned for the long-tailed setting.

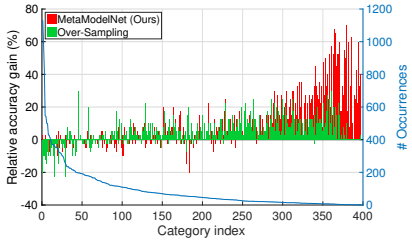


Figure 3: Detailed per class performance comparison between our MetaModelNet and the state-of-the-art over-sampling approach for long-tailed scene classification on the SUN-397 dataset. X-axis: class index. Y-axis (Left): per class classification accuracy improvement relative to the plain baseline. Y-axis (Right): number of training examples. Ours significantly improves for the few-shot tail classes.

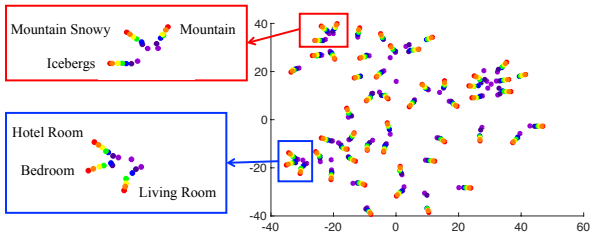


Figure 4: Visualizing model dynamics. We visualize models as points in the "dual" model (parameter) space and examine the evolution of parameters predicted by MetaModelNet with t-SNE [60]. 1-shot models (purple) to many-shot models (red) are plotted in a rainbow order. These visualizations show that MetaModelNet learns an approximately-smooth, nonlinear warping of this space that transforms (few-shot) input points to (many-shot) output points. Similar semantic classes tend to be close and transform in similar ways.

Generalization to other tasks and datasets: Table 4 shows the generality of our approach and shows that the MetaModelNets facilitate the recognition of other long-tailed datasets with significantly different visual concepts and distributions.

4 Conclusions

In this work we proposed a conceptually simple but powerful approach to address the problem of long-tail recognition through knowledge transfer from the head to the tail of the class distribution. Our key insight is to represent the model dynamics through meta-learning, *i.e.,* how a recognition model transforms and evolves during the learning process when gradually encountering more training examples. To do so, we introduce a meta-network that learns to progressively transfer meta-knowledge from the head to the tail classes. We present several state-of-the-art results on benchmark datasets (SUN, Places, and ImageNet) tuned for the long-tailed setting, that significantly outperform common heuristics, such as data resampling or reweighting.

Acknowledgments. We thank Liangyan Gui and Olga Russakovsky for valuable and insightful discussions. This work was supported in part by ONR MURI N000141612007 and U.S. Army Research Laboratory (ARL) under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. DR was supported in part by the National Science Foundation (NSF) under grant number IIS-1618903, Google, and Facebook. We also thank NVIDIA for donating GPUs and AWS Cloud Credits for Research program.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.
[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.
[7] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017.
[8] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.
[9] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan. Do we need more training data? *IJCV*, 119(1):76–92, 2016.
[10] G. Van Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.
[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.
[12] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalanditis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.
[13] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, 2016.
[14] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN database: Exploring a large collection of scene categories. *IJCV*, 119(1):3–22, 2016.
[15] S. Bengio. Sharing representations for long tail computer vision problems. In *ICMI*, 2015.
[16] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016.
[17] Q. Zhong, C. Li, Y. Zhang, H. Sun, S. Yang, D. Xie, and S. Pu. Towards good practices for recognition & detection. In *CVPR workshops*, 2016.
[18] S. J. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2010.
[19] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
[20] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016.
[21] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.
[22] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
[23] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.
[24] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.
[25] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.
[26] K. Li and J. Malik. Learning to optimize. In *ICLR*, 2017.
[27] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.
[28] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.
[29] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.
[30] A. Sinha, M. Sarkar, A. Mukherjee, and B. Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. In *ICLR*, 2017.
[31] J. Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[32] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.
[33] J. Schmidhuber. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, 1993.
[34] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *ICLR*, 2017.
[35] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
[36] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, 2017.
[37] T. Munkhdalai and H. Yu. Meta networks. In *ICML*, 2017.
[38] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006.
[39] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.
[40] Y.-X. Wang and M. Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015.
[41] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Workshops*, 2015.
[42] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
[43] J. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, 2015.
[44] Y.-X. Wang and M. Hebert. Learning by transferring from unsupervised universal sources. In *AAAI*, 2016.
[45] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016.
[46] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.
[47] Y.-X. Wang and M. Hebert. Learning from small sample sets by combining unsupervised meta-training with CNNs. In *NIPS*, 2016.
[48] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.
[49] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.
[50] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*, 2016.
[51] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.
[52] Y. Fu, T. Xiang, Y.-G. Jiang, X. Xue, L. Sigal, and S. Gong. Recent advances in zero-shot recognition. *IEEE Signal Processing Magazine*, 2017.
[53] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 2017.
[54] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *NIPS*, 2017.
[55] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.
[56] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.
[57] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *CVPR*, 2017.
[58] H. He and E. A. Garcia. Learning from imbalanced data. *TKDE*, 21(9):1263–1284, 2009.
[59] C. Huang, Y. Li, C. C. Loy, and X. Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.
[60] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9(Nov):2579–2605, 2008.