# Incremental Few-Shot Learning with Attention Attractor Networks

**Mengye Ren, Renjie Liao, Ethan Fetaya & Richard S. Zemel**
University of Toronto and Vector Institute
{mren, rjliao, ethanf, zemel}@cs.toronto.edu

## Abstract

This paper addresses the problem, *incremental few-shot learning*, where a regular classification network has already been trained to recognize a set of base classes; and several extra novel classes are being considered, each with only a few labeled examples. The model is then evaluated on the overall performance of both base and novel classes. To this end, we propose a meta-learning model, the Attention Attractor Network, which regularizes the learning of novel classes. In each episode, we train a set of new weights to recognize novel classes until they converge. We demonstrate that the learned attractor network can recognize novel classes while remembering old classes, outperforming baselines that do not rely on an iterative optimization process.

## 1 Introduction

The success of deep learning stems from the availability of large scale annotated datasets, such as ImageNet [1]. The need for such a large dataset is however a limitation, since its collection requires intensive human labor. This is also strikingly different from human learning, where new concepts can be learned from very few examples. One line of work that attempts to bridge this gap is few-shot learning [2, 3, 4], where a model learns to output a classifier given only a few labeled examples of the unseen classes. While this is a very promising line of work, few-shot models only focus on learning novel classes, ignoring the fact that many common classes are already available.

An approach that aims to enjoy the best of both worlds, the ability to learn from large datasets for common classes with the flexibility of few-shot learning for others, is *incremental few-shot learning* [5]. This combines incremental learning where we want to add new classes without catastrophic forgetting [6], with few-shot learning when the new classes, unlike the base classes, only have a small amount of examples. One use case to illustrate the problem is a visual aid system. Most objects of interest are common to all users, e.g., cars, pedestrian signals; however, users would also like to augment the system with additional personalized items or important landmarks in their area.

In this work we present a novel method for incremental few-shot learning where during meta-learning we optimize a regularizer that reduces catastrophic forgetting from the incremental few-shot learning. Our proposed "attention attractor network" regularizer is inspired by attractor networks [7] and can be thought of as a learned memory of the base classes. We also show how this regularizer can be optimized, using recurrent back-propagation [8, 9, 10] to back-propagate through the few-shot optimization stage. Finally, we show that our proposed method can produce state-of-the-art results in on the *mini*-ImageNet [3] and *tiered*-ImageNet [11] datasets.

## 2 Attention Attractor Network for Incremental Few-Shot Learning

In this section, we first define the setup of incremental few-shot learning, and then we introduce our new model, the Attention Attractor Network, which attends to the set of base classes according to the few-shot training data, and learns the attractor energy function as an additional regularizing term for the few-shot episode. Figure 1 illustrates the high-level model diagram of our method.

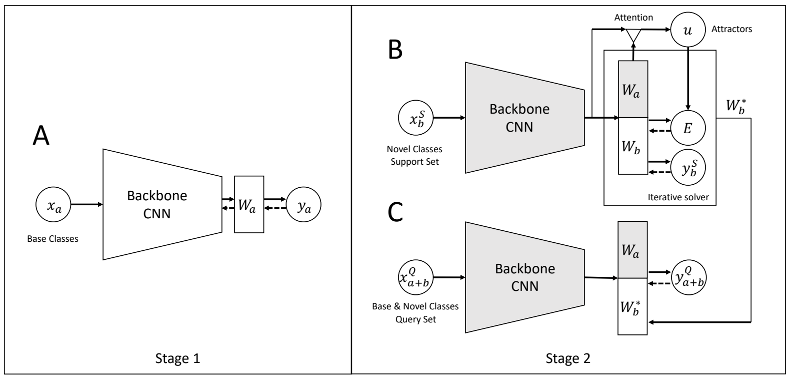

Figure 1: Our proposed attention attractor network for incremental few-shot learning. **A**: In stage 1, we learn $W_a$ and the feature extractor CNN backbone through supervised pretraining. **B**: In stage 2 we learn $W_b$ on a few-shot episode through an iterative solver, to minimize cross entropy plus an additional energy term predicted by attending to the base class representation $W_a$. **C**: The attention attractor network is learned end-to-end to minimize the expected query loss.

## 2.1 Incremental few-shot learning

We first define the task of incremental few-shot learning in this section. In this paper, we consider the two stage learning model proposed by [5].

**Stage 1:** We learn a base model for the regular supervised classification task on dataset $\mathcal{D}_a = \{(x_{a,i}, y_{a,i})\}_{i=1}^{N_a}$ where $x_{a,i}$ is the $i$-th example from dataset $a$ and its labeled class $y_{a,i} \in \{1, 2, ..., K\}$.

**Stage 2:** A few-shot dataset $\mathcal{D}_b$ is presented, from which we can sample few-shot learning episodes $\mathcal{E}$. For each $N$-shot $K'$-way episode, there are $K'$ novel classes disjoint from the base classes. Each novel class has $N$ and $M$ images from the support set $S_b$ and the query set $Q_b$ respectively. Therefore, we have $\mathcal{E} = (S_b, Q_b), S_b = (x^S_{b,i}, y^S_{b,i})_{i=1}^{N\times K'}, Q_b = (x^Q_{b,i}, y^Q_{b,i})_{i=1}^{M\times K'}$ where $y_{b,i} \in \{K+1, ..., K+K'\}$. $S_b$ and $Q_b$ can be regarded as the training set and the validation set in a regular supervised learning setting. To evaluate the performance on a joint prediction of both base and novel classes, a mini-batch $Q_a = \{(x_{a,i}, y_{a,i})\}_{i=1}^{M\times K}$ from $\mathcal{D}_a$ is also added to $Q_b$ to form $Q_{a+b} = Q_a \cup Q_b$. During this stage, the learning algorithm has only access to samples from $S_b$, i.e., few-shot examples from the new classes.

### 2.1.1 Joint prediction on old and new classes

When solving a few-shot episode, we consider learning a linear classifier on the novel classes in the support set $S_b$. We fix the hidden representation $h$ and the linear classification weights $W_a$ for the base classes, that where trained in stage 1, and only train the classification weights $W_b$ for the novel classes. During joint prediction we concatenate the weights to produce a $(K+K')$-way prediction.

During the learning stage of each few-shot episode, we treat it as a classification problem and aim to minimize the following regularized cross-entropy objective on the support set $S$,

$$W_b^* = \arg\min L^S(W_b) = \frac{1}{NK'} \sum_{i=1}^{NK'} \sum_{c=1}^{K+K'} y^S_{b,i,c} \log \hat{y}^S_{b,i,c}(W_b) + R(W_b; \theta_E) \qquad (1)$$

## 2.2 Attention attractor networks

Directly learning the few-shot episode, e.g., by setting $R(W_b; \theta_E)$ to be zero or simple weight decay, can cause catastrophic forgetting on the old classes. This is because $W_b$ is trying to minimize the probability of all the old classes, as it is trained only on new classes. In this section, we introduce Attention Attractor Networks to address this problem.

The attractor network adds an energy term $E$ to the few-shot learning objective, $R(W_b; \theta_E) = \frac{1}{2}\lambda\|W_b\|_2^2 + E(W_b; \theta_E)$. During meta-learning, $\theta_E$ are updated to minimize an expected loss of the query set $Q_{a+b}$ which contains both old and new classes, averaging over all few-shot learning episodes.

$$\theta_E^* = \arg\min_{\theta_E} \mathbb{E}_{\mathcal{E}}[L^Q(\theta_E, S_b)] = \arg\min_{\theta_E} \mathbb{E}_{\mathcal{E}} \left[ \sum_{j=1}^{M(K+K')} \sum_{c=1}^{K+K'} y_{j,c} \log \hat{y}_{j,c}(\theta_E, S_b) \right]. \tag{2}$$

Conceptually, the energy function regularizes the learning of $W_b$ so that it is compatible with $W_a$ during the joint prediction. In our proposed model each base class in $W_a$ has a learned attractor $U_k$ stored in the knowledge base matrix $U = [U_1, ..., U_K]$. When a novel class $k'$ is seen, its classifier is regularized towards its attractor $u_{k'}$ which is a weighted sum of $U_k$. The weighting can be seen as an attention mechanism where each new class attends to the old classes according to the level of interference. Specifically, the attention mechanism is implemented by the cosine similarity function. For each image in the support set, we compare it with the set of base weights $W_a$, average over each support class and apply a softmax function,

$$a_{k',k} = \frac{\exp\left(\frac{1}{N}\sum_j \tau A(h_j, W_{a,k}) \mathbb{1}[y_{b,j} = k']\right)}{\sum_k \exp\left(\frac{1}{N}\sum_j \tau A(h_j, W_{a,k}) \mathbb{1}[y_{b,j} = k']\right)}, \tag{3}$$

where $h_j$ are the representations of the inputs in the support set $S_b$ and $\tau$ is a learnable temperature scalar. $a_{k',k}$ encodes a normalized pairwise attention matrix between the novel classes and the base classes. The attention vector is then used to compute a linear weighted sum of entries in the knowledge base matrix $U$, $u_{k'} = \sum_k a_{k',k} U_k + U_0$. The final energy function is defined as a sum of squared $L_2$ distance from the attractors,

$$E(W_b; \theta_E) = \frac{1}{2} \sum_{k'} (W_{b,k'} - u_{k'})^\top \mathrm{diag}(\exp(\gamma))(W_{b,k'} - u_{k'}), \tag{4}$$

where $\gamma$ is a learnable vector, which defines a diagonal distance metric over the hidden dimensions. Our proposed attention attractor network takes inspiration from attractor networks [12, 7], where for each base class we learn an "attractor" that stores the relevant memory regrading that class.

In summary, $\theta_E$ is a tuple of $(U, U_0, \gamma, \tau)$. The number of parameters is on the same order as a fully connected layer $W_a$. It is important to note that $E(W_b; \theta_E)$ is convex in $W_b$, so the optimum $W_b^*(\theta_E, S_b)$ is guaranteed to be unique and achievable.

### 2.3 Learning an energy function using recurrent backpropagation

As there is no closed-form solution to the regularized linear classification problem shown above, in each training step, we need to solve for $L^S$ to obtain $W_b^*$ through an iterative optimizer. For simple energy functions, we can unroll the iterative optimization process in the computation graph and use the backpropagation through time (BPTT) algorithm [13]. However, the number of iterations for a gradient-based optimizer to converge can be on the order of thousands, and BPTT can be computationally prohibitive. Another way is to use the truncated BPTT [14] (T-BPTT) algorithm that optimizes for the initial $T$ steps of gradient-based optimization, and is commonly used in meta-learning problems. However, when $T$ is small the training objective is biased compared to the objective that considers the optimal $W_b$.

Alternatively, the recurrent backpropagation (RBP) algorithm [9, 10, 8] allows us to backpropagate through fixed-point iterations efficiently without unrolling the computation graph and storing intermediate activations. Our model uses a standalone solver to solve the inner optimization and uses RBP for efficiently passing gradients to $\theta_E$.

## 3 Experiments

**Experimental setup** We used few-shot image classification benchmarks, *mini*-ImageNet [3] and *tiered*-ImageNet [11], and made modifications of the datasets to accommodate the incremental few-shot learning settings. For meta-learning we need to split between training base classes, and training classes from which we sample few-shot episodes. In *tiered*-ImageNet we split the 351 training classes to 200 base classes and 151 novel classes used to train the meta-learner. In *mini*-ImageNet

Table 1: *mini-ImageNet* 64+5-way few-shot classification results

| Model | Backbone | Base | 1-shot Novel | 1-shot Both | 1-shot $\overline{\Delta}$ | 5-shot Novel | 5-shot Both | 5-shot $\overline{\Delta}$ |
|---|---|---|---|---|---|---|---|---|
| MatchingNets [3] | C64 | - | 43.60 | - | - | 55.30 | - | - |
| Meta-LSTM [15] | C32 | - | $43.40 \pm 0.77$ | - | - | $60.20 \pm 0.71$ | - | - |
| MAML [16] | C64 | - | $48.70 \pm 1.84$ | - | - | $63.10 \pm 0.92$ | - | - |
| RelationNet [17] | C64 | - | $50.44 \pm 0.82$ | - | - | $65.32 \pm 0.70$ | - | - |
| R2-D2 [18] | C256 | - | $51.20 \pm 0.60$ | - | - | $68.20 \pm 0.60$ | - | - |
| SNAIL [19] | ResNet | - | $55.71 \pm 0.99$ | - | - | $68.88 \pm 0.92$ | - | - |
| ProtoNet [4] | C64 | - | $49.42 \pm 0.78$ | - | - | $68.20 \pm 0.66$ | - | - |
| ProtoNet (our implementation) | ResNet | 75.79 | $50.09 \pm 0.41$ | 42.73 | -20.21 | $70.76 \pm 0.19$ | 57.05 | -31.72 |
| LwoF [5] | ResNet | 80.24 | $55.45 \pm 0.89$ | 51.23 | - | $70.92 \pm 0.35$ | 56.04 | - |
| LwoF (our implementation) | ResNet | 74.58 | $56.97 \pm 0.24$ | 52.37 | -13.65 | $70.50 \pm 0.36$ | 59.90 | -14.18 |
| Ours (1st stage) | ResNet | 77.17 | $54.78 \pm 0.43$ | 52.74 | -13.95 | $70.57 \pm 0.36$ | 60.34 | -13.60 |
| Ours (full model) | ResNet | 76.84 | $55.72 \pm 0.41$ | **54.89** | **-11.39** | $70.50 \pm 0.36$ | **62.37** | **-11.48** |

Table 2: *tiered-ImageNet* 200+5-way few-shot classification results

| Model | Backbone | Base | 1-shot Novel | 1-shot Both | 1-shot $\overline{\Delta}$ | 5-shot Novel | 5-shot Both | 5-shot $\overline{\Delta}$ |
|---|---|---|---|---|---|---|---|---|
| ProtoNet [4] | ResNet | 57.76 | $58.48 \pm 0.43$ | 34.12 | -24.00 | $75.44 \pm 0.38$ | 45.70 | -20.88 |
| LwoF [5] | ResNet | 62.66 | $56.29 \pm 0.49$ | 57.96 | -1.52 | $72.98 \pm 0.41$ | 66.54 | -1.33 |
| Ours (1st stage) | ResNet | 62.45 | $56.89 \pm 0.46$ | 50.82 | -8.85 | $74.82 \pm 0.38$ | 63.66 | -5.00 |
| Ours (full model) | ResNet | 62.02 | $58.89 \pm 0.46$ | **60.08** | **-0.25** | $74.79 \pm 0.38$ | **68.16** | **-0.35** |

this is problematic as there are only 64 training classes, therefore at each meta-learning episode we randomly erase 5 classes, considering the 59 left as our base classes and sampling a few-shot episode from the removed 5 classes.

**Evaluation metrics** We consider the following evaluation metrics: 1) overall accuracy on individual query sets and the joint query set ("Base", "Novel", and "Both"); and 2) decrease in performance during joint prediction within the base and novel classes, considered separately ("$\Delta_a$" and "$\Delta_b$"). Finally we take the average $\overline{\Delta} = \frac{1}{2}(\Delta_a + \Delta_b)$ as a key measure of the overall decrease in accuracy.

**Results** We present the few-shot benchmark in Table 1 and2. On *mini*-ImageNet, we compare our models to other incremental few-shot learning methods as well as pure few-shot learning methods. Note that our model uses the same backbone architecture as [19] and [5], and is directly comparable with their results. The "1st stage" baseline is a CNN trained on the base classes, where we train the new classes using logistic regression with weight decay.

We implemented and compared to two methods. First, we adapted ProtoNet [4] to incremental few-shot settings. For both base and novel classes, we store the average representation, and we retrieve the nearest neighbor by comparing the representation of a test image. We also compare to Dynamic Few-Shot Learning without Forgetting (LwoF) [5]. Here the base weights $W_a$ are learned regularly through supervised pre-training, and $W_b$ are computed using prototypical averaging. We implemented the most advanced variants that involves a class-wise attention mechanism. The main difference to this work is that we use an iterative optimization to compute $W_b$.

Shown in Table 1 and2, our proposed method consistently outperform the two approaches described above, particularly in "$\overline{\Delta}$". Our method is significantly improved during the second stage training, compared to a baseline that has only been pre-trained on base classes in the first stage.

## 4 Conclusion and Future Work

Incremental few-shot learning, the ability to jointly predict based on a set of pre-defined concepts as well as additional novel concepts, is an important step towards making machine learning models more flexible and usable in everyday life. In this work, we propose an attention attractor model, which outputs an additional energy function by attending to the set of base classes. The iterative model that learns to remember the base classes without needing to review examples from the original training set, outperforming baselines that only do one-step inference. Future directions of this work include sequential iterative learning of few-shot novel concepts, and hierarchical memory organization.

## References

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[2] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[3] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

[4] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

[5] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[7] Richard S Zemel and Michael C Mozer. Localist attractor networks. *Neural Computation*, 13(5):1045–1064, 2001.

[8] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard S. Zemel. Reviving and improving recurrent back-propagation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[9] Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the 1st International Conference on Neural Networks*, volume 2, pages 609–618. IEEE, 1987.

[10] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

[11] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proceedings of 6th International Conference on Learning Representations (ICLR)*, 2018.

[12] Michael C Mozer. Attractor networks. *The Oxford companion to consciousness*, pages 86–89, 2009.

[13] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[14] Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

[15] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[17] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[19] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

## A Datasets

We experiment on two datasets, *mini*-ImageNet and *tiered*-ImageNet. Both are sub-sets of imagenet [1], with images size reduced to size of $84 \times 84$ pixels

- ***mini*-ImageNet** Proposed by [3], *mini*-ImageNet which contains 100 object classes and 60,000 images. We used the splits proposed by [15], where training, validation, and testing have 64, 16, and 20 classes respectively.
- ***tiered*-ImageNet** Proposed by [11], *tiered*-ImageNet is a larger subset of ILSVRC-12. It features a categorical split among training, validation, and testing subsets. The categorical split means that classes that belong to the same high-level category, e.g. working dog, are not split between training, validation and test. This is harder task, but one that more truthfully evaluates generalization to new classes. It is also an order of magnitude larger than *mini*-ImageNet.

Details about number of samples are presented in Table 3.

Table 3: *mini*-ImageNet and *tiered*-ImageNet split statistics

| | | *mini*-ImageNet | | | *tiered*-ImageNet | | |
|---|---|---|---|---|---|---|---|
| Classes | Purpose | Split | N. Cls | N. Img | Split | N. Cls | N. Img |
| Base | Train | Train-Train | 64 | 38,400 | Train-A-Train | 200 | 203,751 |
| | Val | Train-Val | 64 | 18,748 | Train-A-Val | 200 | 25,460 |
| | Test | Train-Test | 64 | 19,200 | Train-A-Test | 200 | 25,488 |
| Novel | Train | Train-Train | 64 | 38,400 | Train-B | 151 | 193,996 |
| | Val | Val | 16 | 9,600 | Val | 97 | 124,261 |
| | Test | Test | 20 | 12,000 | Test | 160 | 206,209 |

### A.1 Validation and testing splits for base classes

In standard few-shot learning, meta-training, validation, and test set have disjoint sets of object classes. However, in our incremental few-shot learning setting, to evaluate the model performance on the base class predictions, additional splits of validation and test splits of the meta-training set are required. Splits and dataset statistics are listed in Table 3. For *mini*-ImageNet, [5] released additional images for evaluating training set, namely "Train-Val" and "Train-Test". For *tiered*-ImageNet, we split out $\approx 20\%$ of the images for validation and testing of the base classes.

### A.2 Novel classes

In *mini*-ImageNet experiments, the same training set is used for both $\mathcal{D}_a$ and $\mathcal{D}_b$. In order to pretend that the classes in the few-shot episode are novel, following [5], we masked the base classes in $W_a$, which contains 64 base classes. In other words, we essentially train for a 59+5 classification task. We found that under this setting, the progress of meta-learning in the second stage is not very significant, since all classes have already been seen before.

In *tiered*-ImageNet experiments, to emulate the process of learning novel classes during the second stage, we split the training classes into base classes ("Train-A") with 200 classes and novel classes ("Train-B") with 151 classes, just for meta-learning purpose. During the first stage the classifier is trained using Train-A-Train data. In each meta-learning episode we sample few-shot examples from the novel classes (Train-B) and a query base set from Train-A-Val.

**200 Base Classes ("Train-A"):**

```
n02128757, n02950826, n01694178, n01582220, n03075370, n01531178,
n03947888, n03884397, n02883205, n03788195, n04141975, n02992529,
n03954731, n03661043, n04606251, n03344393, n01847000, n03032252,
n02128385, n04443257, n03394916, n01592084, n02398521, n01748264,
n04355338, n02481823, n03146219, n02963159, n02123597, n01675722,
n03637318, n04136333, n02002556, n02408429, n02415577, n02787622,
n04008634, n02091831, n02488702, n04515003, n04370456, n02093256,
```

n01693334, n02088466, n03495258, n02865351, n01688243, n02093428,
n02410509, n02487347, n03249569, n03866082, n04479046, n02093754,
n01687978, n04350905, n02488291, n02804610, n02094433, n03481172,
n01689811, n04423845, n03476684, n04536866, n01751748, n02028035,
n03770439, n04417672, n02988304, n03673027, n02492660, n03840681,
n02011460, n03272010, n02089078, n03109150, n03424325, n02002724,
n03857828, n02007558, n02096051, n01601694, n04273569, n02018207,
n01756291, n04208210, n03447447, n02091467, n02089867, n02089973,
n03777754, n04392985, n02125311, n02676566, n02092002, n02051845,
n04153751, n02097209, n04376876, n02097298, n04371430, n03461385,
n04540053, n04552348, n02097047, n02494079, n03457902, n02403003,
n03781244, n02895154, n02422699, n04254680, n02672831, n02483362,
n02690373, n02092339, n02879718, n02776631, n04141076, n03710721,
n03658185, n01728920, n02009229, n03929855, n03721384, n03773504,
n03649909, n04523525, n02088632, n04347754, n02058221, n02091635,
n02094258, n01695060, n02486410, n03017168, n02910353, n03594734,
n02095570, n03706229, n02791270, n02127052, n02009912, n03467068,
n02094114, n03782006, n01558993, n03841143, n02825657, n03110669,
n03877845, n02128925, n02091032, n03595614, n01735189, n04081281,
n04328186, n03494278, n02841315, n03854065, n03498962, n04141327,
n02951585, n02397096, n02123045, n02095889, n01532829, n02981792,
n02097130, n04317175, n04311174, n03372029, n04229816, n02802426,
n03980874, n02486261, n02006656, n02025239, n03967562, n03089624,
n02129165, n01753488, n02124075, n02500267, n03544143, n02687172,
n02391049, n02412080, n04118776, n03838899, n01580077, n04589890,
n03188531, n03874599, n02843684, n02489166, n01855672, n04483307,
n02096177, n02088364.

**151 Novel Classes ("Train-B"):**

n03720891, n02090379, n03134739, n03584254, n02859443, n03617480,
n01677366, n02490219, n02749479, n04044716, n03942813, n02692877,
n01534433, n02708093, n03804744, n04162706, n04590129, n04356056,
n01729322, n02091134, n03788365, n01739381, n02727426, n02396427,
n03527444, n01682714, n03630383, n04591157, n02871525, n02096585,
n02093991, n02013706, n04200800, n04090263, n02493793, n03529860,
n02088238, n02992211, n03657121, n02492035, n03662601, n04127249,
n03197337, n02056570, n04005630, n01537544, n02422106, n02130308,
n03187595, n03028079, n02098413, n02098105, n02480855, n02437616,
n02123159, n03803284, n02090622, n02012849, n01744401, n06785654,
n04192698, n02027492, n02129604, n02090721, n02395406, n02794156,
n01860187, n01740131, n02097658, n03220513, n04462240, n01737021,
n04346328, n04487394, n03627232, n04023962, n03598930, n03000247,
n04009552, n02123394, n01729977, n02037110, n01734418, n02417914,
n02979186, n01530575, n03534580, n03447721, n04118538, n02951358,
n01749939, n02033041, n04548280, n01755581, n03208938, n04154565,
n02927161, n02484975, n03445777, n02840245, n02837789, n02437312,
n04266014, n03347037, n04612504, n02497673, n03085013, n02098286,
n03692522, n04147183, n01728572, n02483708, n04435653, n02480495,
n01742172, n03452741, n03956157, n02667093, n04409515, n02096437,
n01685808, n02799071, n02095314, n04325704, n02793495, n03891332,
n02782093, n02018795, n03041632, n02097474, n03404251, n01560419,
n02093647, n03196217, n03325584, n02493509, n04507155, n03970156,
n02088094, n01692333, n01855032, n02017213, n02423022, n03095699,
n04086273, n02096294, n03902125, n02892767, n02091244, n02093859,
n02389026.