Gotta Learn 'Em All

Xavier Sánchez, Judith Aguilera, Carlota Criado

December 28, 2025

1 Introduction

Deep Learning models have achieved remarkable success in image classification tasks, often surpassing human performance. However, this success relies heavily on large-scale labeled datasets. While the human visual system demonstrates an exceptional ability to generalize from a few examples, instantly recognizing a new species after seeing it once, conventional neural networks require thousands of iterations to master similar distinctions [4]. This discrepancy highlights a critical limitation in current AI: data inefficiency.

In this work, we address the challenge of Few-Shot Learning (FSL), specifically in the context of fine-grained classification using Pokémon species as a proxy for complex biological taxonomies. The core problem is to bridge the gap between human-like rapid learning and machine learning optimization, enabling a model to generalize effectively from limited support sets (1-shot or 5-shot) rather than extensive databases [10]. To address this challenge, we investigate three distinct theoretical paradigms in FSL, representing opposing philosophies in learning strategies:

1. **Transfer Learning (Baseline):** A standard pre-training and fine-tuning approach, serving as a stability reference [12].
2. **Optimization-based Meta-Learning (Reptile):** A method focused on finding an optimal initialization of parameters that can be rapidly adapted to new tasks [5].
3. **Model-based Meta-Learning (HyperNetwork):** An approach that generates weights dynamically for a given task, effectively learning to generate a classifier rather than just classifying [3].

Beyond raw accuracy, this study aims to stress-test the "intelligence" and robustness of these architectures. We structure our investigation around three key Research Questions (RQs):

RQ1 (Data Efficiency): How does model performance degrade in extreme low-data regimes (1-shot vs. 5-shot)?

RQ2 (Robustness): Can meta-learning algorithms maintain performance under high inter-class similarity ("Hard Mode")?

RQ3 (Semantic Understanding): Do the models learn hierarchical semantic features (evolutionary stages) or merely rely on low-level visual cues (color/texture)?

2 Related Work

The field of FSL has evolved into two main streams: metric-based and optimization-based approaches. Metric-based methods, such as Prototypical Networks [8] or Matching Networks [10], learn a shared metric space where classification is performed by computing distances to class prototypes. While effective, they often struggle when the domain shift between base and novel classes is significant.

To overcome these limitations, optimization-based meta-learning aims to "learn to learn." The most prominent example is MAML (Model-Agnostic Meta-Learning) [2], which optimizes for an initialization that can adapt to new tasks with few gradient steps. Reptile [5] simplifies this by removing the need for second-order derivatives, making it computationally more efficient for our constraints. Alternatively, model-based approaches like HyperNetworks [3] use a secondary network to predict the parameters of the main classifier, offering a flexible "generative" solution to weight adaptation.

Despite the rise of meta-learning, recent studies suggest that simple Transfer Learning [12] with a robust backbone (e.g., ResNet) can be a surprisingly strong baseline, sometimes outperforming complex meta-learning algorithms when domain differences are minimal [1].

3 Methodology

3.1 Datasets and Pre-processing

We aggregated the primary dataset via PokeAPI [6], covering Generations I–V (649 species). This restriction ensures visual consistency by focusing exclusively on 2D pixel art, avoiding the domain shift introduced by 3D models in later generations. The collection consists of:

Sprites: Sourced from 9 distinct game releases, utilizing both Front and Back views to maximize intra-class diversity.

Metadata: A comprehensive CSV mapping species to biological attributes (Type, Generation) and evolutionary lineage IDs

Figure 1: All available Bulbasaur sprites across mainline Pokémon games from generation 1 to 5, "?" meaning non-existent

We manage the dataset using a custom "PokemonMetaDataset" class which performs three critical functions:

1. **Metadata Integration:** Maps images to semantic attributes for structured sampling.
2. **Background Standardization:** Converts backgrounds to white using alpha composition to ensure uniformity.
3. **Optimized Indexing:** Pre-calculates indices by label, enabling $O(1)$ retrieval for episodic support/query splits.

3.1.1 Data Augmentation

To mitigate data leakage from near-identical sprites and artificially increase sample diversity, we implemented a strict augmentation pipeline:

Standardization: We resize all inputs to 84×84 pixels.

Geometric Variations: We apply random horizontal flips (50% probability) and affine transformations (rotation $\pm 15°$, shift, and scale) while maintaining background integrity.

Color Robustness: We apply random jitter to brightness, contrast, and saturation to prevent the model from overfitting to specific color palettes.

3.1.2 Dataset Analysis and Data Preparation

Following data aggregation, we performed a structural validation to ensure the dataset's suitability for Meta-Learning.

Validation & Preprocessing

We verified a 100% correspondence between metadata and image files. To standardize inputs for the CNN backbone, we implemented a resizing pipeline to normalize all sprites (originally 48x48 to 96x96) to 84x84 pixels. Additionally, due to sample scarcity in Generation V, we restricted the final training set to Generations I–IV, ensuring sufficient sample density to form stable support and query sets.

Distribution & Imbalance

Analysis of the elemental distribution revealed a Long-Tail profile: while types like Water are abundant, others like Ghost or Dragon are scarce (2). We deliberately preserved this imbalance rather than correcting it, utilizing these rare classes to construct "Hard Mode" episodes that test the model's generalization capabilities under strict scarcity.

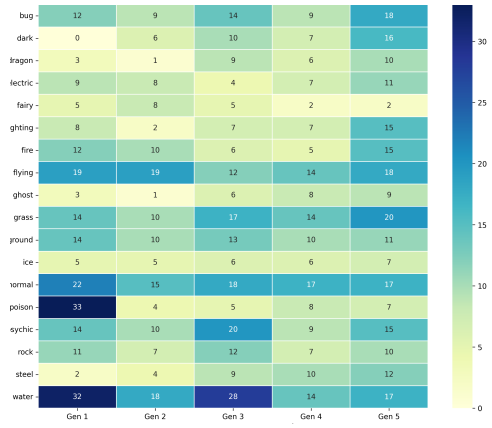

Figure 2: Heatmap showing the distribution of Pokémon types per generation

Evolutionary Structure (Task 2)

To enable evolutionary relationship prediction, we mapped the biological structure of the dataset. We identified approximately 232 unique families, distinguishing "First Links" (base forms) from "Last Links" (final evolutions). This hierarchy was encoded into new metadata columns (family_id, pre_evo, family_leader), allowing us to sample episodes based on biological roles rather than mere visual similarity.

name	Charmander	Charmeleon	Squirtle
dex_num	4	5	6
type_1	Fire	Fire	Water
type_2	None	None	None
gen	1	1	1
images	15	15	15
pre_evo	None	Charmander	None
evolution	Charmeleon	Charizard	Wartortle
folder_name	004-Charmander	005-Charmeleon	007-Squirtle
family_leader	Charmander	Charmander	Squirtle
family_id	1	1	2

Table 1: Sample of the final processed dataset. Note the family_id column, which assigns the same ID to all members of an evolutionary line.

3.1.3 Meta-Train, Meta-Test and Meta-Validation partitions

Unlike standard supervised learning, Meta-Learning requires a class-disjoint split [10]: if a species appears in the training set, it must not appear in the test set. This ensures evaluation focuses on learning new concepts rather than memorization. To rigorously evaluate the model's generalization capabilities under different constraints, we implemented three distinct partitioning strategies:

Random Partitioning (Baseline): Species are sampled uniformly at random. This assumes an i.i.d. distribution and serves as the standard benchmark for raw adaptation capabilities within a consistent visual domain.

Chronological Partitioning (Generation Split): Data is split by game release (e.g., Train on Gen I–III, Test on Gen IV). This evaluates robustness to Domain Shift, testing if feature extractors can extrapolate to the evolving artistic styles and character designs observed across 15 years of game development [7] [9].

Semantic Partitioning (Type Split): Data is split by elemental attributes (e.g., Train on Fire/Water; Test on Dragon/Ghost). Since types often correlate with color (e.g., Fire $\approx$ Red), holding out entire types removes visual shortcuts, forcing the model to learn shape-based and textural features rather than color histograms.

For all configurations, we maintained strict species separation. A subset of training classes (20%) was extracted for Meta-Validation. To ensure fair benchmarking, we enforced a fixed random seed, guaranteeing that the stochastic assignment of species to splits remains identical across all experiments.

3.2 Models

3.2.1 Backbone

To ensure a rigorous comparison between the three proposed methodologies (Baseline, Reptile, and HyperNetwork), we establish a common feature extraction architecture. All

models share the exact same Conv-4 Backbone, preventing architectural variations from confounding the results regarding the efficacy of the meta-learning algorithms.

The selected architecture is a variant of the standard Conv-4, a benchmark widely used in Few-Shot Learning literature [8] . This network is responsible for mapping the high-dimensional input space (raw pixels) into a lower-dimensional, semantically meaningful embedding space.

Architectural Details

The network accepts an input tensor $x \in \mathbb{R}^{3\times84\times84}$, representing an RGB image resized to a standard resolution. The processing pipeline consists of four identical convolutional blocks applied sequentially.Each block B_i performs the following operations:

Convolution: A 3×3 convolution with 128 filters (num_features) and padding of 1 to preserve spatial dimensions before pooling.

Normalization: We made a critical design choice to implement Group Normalization [11] instead of the standard Batch Normalization.

> In Few-Shot Learning, particularly in 1-shot scenarios or during episodic testing, batch sizes can be extremely small or statistically unstable. Group Normalization calculates statistics based on channel groups within a single sample, making the feature extraction robust to batch size variations and ensuring stability during meta-training.

Activation: A Rectified Linear Unit (ReLU) nonlinearity.

Pooling: A 2×2 Max-Pooling operation to reduce spatial dimensionality.

Dimensionality Reduction

The backbone transforms the input through a progressive reduction of spatial resolution while increasing the depth of the feature maps. The transformation flow is defined as:

$$\text{Input } (84 \times 84) \xrightarrow{B_1} (42 \times 42) \xrightarrow{B_2} (21 \times 21) \xrightarrow{B_3} (10 \times 10) \xrightarrow{B_4} (5 \times 5) \quad (1)$$

The final output is a tensor of shape $128 \times 5 \times 5$. This tensor is flattened into a single embedding vector $z \in \mathbb{R}^d$, where $d = 3200$. This vector z serves as the concise "fingerprint" of the Pokémon, which is subsequently fed into the distinct classification heads or adaptation mechanisms of the Baseline, Reptile, or HyperNetwork.

3.2.2 Baseline Model: Transfer Learning with Finetuning

Although classically categorized as Transfer Learning, our Baseline effectively implements an explicit optimization-based adaptation strategy at test time. While Meta-Learning algorithms learn how to adapt during training, our Baseline separates the process: it acquires universal knowledge during pre-training and mimics the "inner-loop" of a meta-learner during inference. This allows us to test whether a strong, fixed visual representation coupled with aggressive, hand-crafted optimization is sufficient to solve the Few-Shot problem.

Phase 1: Global Pre-training

The objective of this phase is to learn a robust feature representation using the set of base classes (D_{train}). Unlike the episodic testing phase, this is formulated as a standard supervised classification problem. To construct the task, we implement a Label Remapping strategy: since global Pokémon IDs are non-contiguous, we map the indices of the training species to a local range $y \in \{0, \dots, N_{base} - 1\}$. Crucially, to monitor convergence and prevent overfitting to the base classes, we perform a stratified 80/20 internal split on the training data. The backbone f_θ is optimized end-to-end using the Adam optimizer to minimize standard Cross-Entropy loss, ensuring the extraction of discriminative visual traits that remain effective when transferred to unseen species.

Phase 2: Episodic Test-Time Adaptation

During the testing phase, the model shifts from a static classifier to a dynamic learner. We treat each episode not merely as an inference task, but as a discrete learning problem. The inference process effectively becomes a "Meta-Learning" loop defined as follows:

Frozen Backbone (The Constant): The pre-trained backbone f_θ is frozen. It acts as the immutable "eye" of the model, mapping images to embeddings ($z \in \mathbb{R}^{3200}$) based on its prior knowledge.

Transient Head Initialization (The Variable): For every episode, we instantiate a strictly temporary, linear classification head $h_{\phi'}$ initialized from scratch for the episode's N classes.

The Adaptation Loop: The model explicitly learns the new task on the fly. We execute an optimization loop using the Support Set where, crucially, we inject Data Augmentation (rotations, color jitter) at every step. We update the head parameters ϕ' via Adam for a fixed budget of 12 iterations.

By doing this, the Baseline does not just "match" images; it actively constructs a decision boundary tailored specifically to the unique distribution of the current episode's support set.

3.2.3 Reptile

Reptile is an optimization-based meta-learning algorithm designed to find an optimal initialization for a neural network. Unlike traditional learning (which learns to classify specific images), Reptile learns a set of base parameters that are "easy to fine-tune." The goal is to find a starting point θ in the parameter space such that performing a few steps of Gradient Descent on any new task (e.g., a new batch of Pokémon) should result in high accuracy. Reptile operates using a double-loop structure, which we implemented directly in your training and evaluation scripts:

Outer Loop (Meta-Optimization): The model samples a specific task (e.g., distinguishing Charmander vs. Squirtle).

Inner Loop (Adaptation):

- It creates a copy of the current model.
- It trains this copy on the Support Set for k steps, 12 in our case).
- This produces a set of adapted weights, $\tilde{\theta}$.

Weight Update:

- Instead of keeping the adapted weights (which are now overfitted to just that one task), Reptile updates the original model by moving it slightly towards the adapted weights.
- Formula: $\theta \leftarrow \theta + \epsilon(\tilde{\theta} - \theta)$
- This gently pulls the base model towards a "centroid" parameter space that is close to the solution for all tasks.

3.2.4 Hypernetworks

Unlike Reptile, which relies on iterative gradient updates, the HyperNetwork [3] adopts a generative approach. We decouple the architecture into two components: a static backbone f_θ that extracts feature embeddings, and a dynamic generator g_ϕ (a Multi-Layer Perceptron). Crucially, while typical HyperNetworks might generate weights for an entire architecture, we restrict our generator to predict only the parameters of the final classification layer. This transforms the problem: instead of training a classifier directly, we train a generator to construct the optimal decision boundary for the current episode's distribution. This represents a paradigm shift: instead of training a classifier to recognize specific Pokémon, we train the generator to predict the optimal parameters for any potential classifier given the task context.

The adaptation process is instantaneous. First, the model aggregates the features of the entire support set to compute a global context vector c by averaging the embeddings of all shots. The HyperNetwork takes this context as input and predicts the complete classification matrix $\mathbf{W}$ required for the episode in a single step:

$$\mathbf{W} = g_\phi(c)$$

where $\mathbf{W} \in \mathbb{R}^{N \times d}$ contains the weights for all N classes simultaneously. This generated matrix acts as the final dense layer for the current episode, allowing the model to classify query images via simple matrix multiplication.

The distinction between phases is critical. During Meta-Training, we optimize the HyperNetwork's parameters ϕ by backpropagating the loss through the generated weights $\mathbf{W}$. However, during Meta-Testing, the model operates in a strictly feed-forward manner. Unlike gradient-based methods that require fine-tuning steps to adapt, the HyperNetwork generates the specific matrix $\mathbf{W}$ for the new task in a single pass without updating any parameters, effectively "hallucinating" a valid decision boundary for a new species immediately.

3.3 Task Design (Samplers)

To execute the episodic training protocol, we implemented a custom data ingestion pipeline encapsulated in the PokemonMetaDataset class. On top of this foundation, we developed two distinct sampler algorithms, Pokédex and Oak, to govern the episodic generation logic.

All experiments adhere to the standard N-way K-shot protocol [10]. Each training iteration (episode) $\mathcal{T}$ is constructed by sampling N classes, from which K examples are drawn for the Support Set $\mathcal{S}$ and Q examples for the Query Set $\mathcal{Q}$.

3.3.1 Task 1: Identification (Pokédex Sampler)

This sampler addresses the fundamental classification problem. Upon initialization, it filters the dataset $\mathcal{D}$ to retain only classes satisfying $|C| \geq K + Q$. For each episode, the sampler selects N distinct classes without replacement. The Support and Query sets are populated disjointly to prevent data leakage.

To evaluate robustness, we modulate the semantic difficulty of the task by curating the input label space:

Random Mode (High Variance): Classes are sampled uniformly from the entire distribution. This creates episodes with high visual disparity (e.g., mixing Types and Generations), establishing the baseline for Data Efficiency.

Hard Mode (High Similarity): To test Robustness, we implement structured splits. The valid label pool is restricted to a specific subset sharing a meta-attribute (e.g., only "Water-type" Pokémon). This compels the model to discriminate based on fine-grained morphological features rather than broad visual distinctors.

Note: This sampler is utilized in experiments 1 & 2 (RQ1 & RQ2)

3.3.2 Task 2: Evolution (Oak Sampler)

This sampler shifts the learning objective from visual matching to relational reasoning. Instead of sampling individual species, it groups the dataset by family_id, treating the Evolutionary Family as the atomic sampling unit.

The critical innovation is the strict separation of distinct morphological forms. For a given family: The Support Set $\mathcal{S}$ contains images exclusively from one specific form (e.g., Stage A). The Query Set $\mathcal{Q}$ contains images exclusively from a different form (e.g., Stage B). (**Note**: If $\mathcal{S}$ is Charizard, $\mathcal{Q}$ could be Charmander OR Charmeleon.)

In a standard 5-way episode, this setup presents a complex discrimination challenge. The model is provided with the base forms of 5 distinct families. When presented with an evolved form (Query), it must correctly map it to its biological ancestor among the 5 available options (1 correct lineage, 4 distractor families).

This constraint ensures that $\mathcal{S}$ and $\mathcal{Q}$ contain physically distinct subjects. Consequently, the model cannot rely on pixel-level texture matching; it must instead learn an abstract representation of the biological continuity across metamorphosis.

Note: This sampler is utilized in experiments 3 & 3.1 (RQ3)

Implementation Note: While the core logic defined above is global, minor adapter layers were implemented for each architecture (Baseline, Reptile, HyperNetwork) to align tensor dimensions and batching strategies with their specific computational graphs

4 Experiments and Results

4.1 Experiment 1: The "K-Shot" & "N-Way" Sensitivity Analysis

The first experiment serves as the foundational benchmark for this study. The primary objective is to establish a comprehensive performance profile for the three proposed architectures (Baseline, Reptile, and HyperNetwork) under standard Few-Shot Learning conditions. This phase ad-

dresses the necessity of understanding how model performance scales with data availability, specifically responding to the need to "experiment with different sizes of support sets."

To achieve this, we employed the Random Species Identification task (Task A). **Detailed visualizations of the training dynamics, including the evolution of loss and accuracy for these models, can be found in Appendix A.1 (see Figures 9, 10 and 11).**

In this setup, we partition the dataset using a standard high-variance random split, where training and testing classes are disjoint but drawn from the same general distribution (randomly selected from the Pokédex). We conducted a rigorous sensitivity analysis by varying two critical dimensions of the Few-Shot problem:

Data Scarcity (K-Shot): We ranged the support set size from $K = 1$ to $K = 5$ images per class. This allows us to observe the "learning curve" of each model, from the extreme One-Shot scenario to a slightly more data-rich environment.

Task Complexity (N-Way): We varied the number of classes per episode from $N = 2$ (binary classification) to $N = 5$. This tests the models' ability to maintain robust decision boundaries as the number of distractors increases.

4.1.1 Outcome

K-Shot / N-Way	1	2	3	4	5
2	92.3%	94.3%	96.4%	96.3%	97.5%
3	88.2%	91.2%	93.7%	94.1%	96.3%
4	85.9%	89.3%	90.2%	92.3%	92.9%
5	83.2%	86%	89.8%	92.1%	92.4%

Table 2: Baseline test accuracy results with finetuning.

K-Shot / N-Way	1	2	3	4	5
2	62%	60%	63%	74.5%	76%
3	40.7%	50%	55.6%	50.3%	60.7%
4	35.5%	41.3%	49.6%	47.5%	53.8%
5	32.8%	36.6%	42.4%	43.6%	52.6%

Table 3: Reptile test accuracy results for Experiment 1

K-Shot / N-Way	1	2	3	4	5
2	83%	85%	89%	87.5%	86%
3	86.33%	84.67%	83.67%	90%	84.67%
4	82.5%	81%	85.5%	85.5%	83%
5	80.6%	85.6%	83.2%	84.4%	85%

Table 4: Hypernetwork test accuracy Results for Experiment 1

The experimental results highlight distinct performance profiles across the evaluated architectures. As shown in Table 2, the Baseline demonstrates remarkable efficiency, starting at 83.2% accuracy in the complex 5-Way 1-Shot scenario and steadily improving to 92.4% with more shots, indicating that its pre-trained features align rapidly with minimal data. In contrast, Reptile (Table 3) struggles with high-variance tasks, collapsing from 62% in binary classification to 32.8% in 5-Way 1-Shot settings, suggesting its initialization is too generic to handle increased complexity without extensive fine-tuning. Meanwhile, the HyperNetwork (Table 4) offers a middle ground characterized by exceptional stability; it achieves a strong 80.6% in 5-Way 1-Shot tasks with negligible sensitivity to distractor classes (<2.5% drop), though it yields diminishing returns as support set size increases compared to the gradient-based Baseline.

4.2 Experiment 2: The "Hard Mode" Challenge (Sampling Strategy)

To evaluate the models' robustness beyond standard conditions, we transitioned from the high-variance Random Split used in Experiment 1 to a "Hard Mode" sampling strategy. We designed this experiment to determine if the models rely on superficial visual shortcuts or learn meaningful semantic features by introducing two rigorous partitioning schemes:

The Type Split (Semantic Constraint): We designed this setting to force the model to discriminate based on morphological features rather than simple color correlations. We enforced a disjoint split where the training set consisted of ten elemental types (Fairy, Dark, Dragon, Rock, Bug, Psychic, Flying, Water, Fire, Grass), and validation used Steel, Ground, and Ghost. Crucially, the models were evaluated on five entirely unseen types: Ice, Poison, Fighting, Electric, and Normal. This tests the meta-learner's ability to generalize to biological concepts and structures never encountered during optimization.

The Generation Split (Stylistic Constraint): To test resilience against art style shifts and temporal domain changes, we separated the data by game release era. The training phase was restricted to Pokémon from Generation I and Generation III, with Generation II serving as validation. Testing was conducted exclusively on Generation IV. This setup evaluates whether the models can adapt to the specific graphical fidelity and artistic style.

4.2.1 Outcome Generation Split

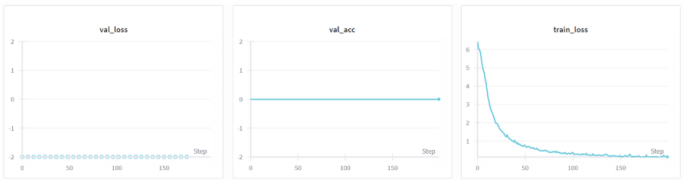

Figure 3: Evolution of Baseline loss and accuracy during training in generation split.

Figure 4: Evolution of Reptile loss and accuracy during training in generation split.

Figure 5: Evolution of Hypernetwork loss and accuracy during training in generation split.

K-Shot / N-Way	1	2	3	4	5
2	91%	94%	96%	97.2%	97.8%
3	87%	89.9%	93%	94.4%	96.5%
4	84.5%	88%	89.8%	92.3%	93.1%
5	82.7%	85.7%	89.6%	91.5%	92.5%

Table 5: Baseline test accuracy results in generation split.

K-Shot / N-Way	1	2	3	4	5
2	51.5%	52%	64.5%	70.5%	84%
3	29.3%	40%	40%	40%	44%
4	29%	32%	34.3%	34.3%	35.8%
5	24.2%	28.8%	32%	33.4%	38.4%

Table 6: Reptile test accuracy results in generation split.

K-Shot / N-Way	1	2	3	4	5
2	93%	91.5%	91.5%	92.5%	93.5%
3	90.33%	91%	88.33%	88.33%	91.67%
4	83%	89.25%	91.25%	91%	88.25%
5	82.6%	86%	86.6%	88.2%	92.2%

Table 7: Hypernetwork test accuracy results in generation split.

Under the Generation Split condition, the Baseline (Table 5) and HyperNetwork (Table 7) exhibit remarkably similar and robust performance profiles. Both models handle the complexity of 5-Way tasks effectively, starting at approximately 82.6%–82.7% accuracy (1-Shot) and recovering to over 92% (5-Shot), demonstrating a consistent gain of 10 percentage points as support data increases. In stark contrast, Reptile (Table 6) reveals a distinct inability to scale with task complexity; while it reaches 84% in simple 2-Way scenarios, its performance collapses to the 24%–38% range in 5-Way settings regardless of shot count, indicating a severe limitation in adapting to this specific domain shift compared to the other architectures.

4.2.2 Outcome Type Split

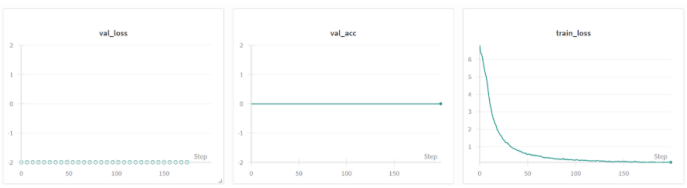

Figure 6: Evolution of Baseline loss and accuracy during training in type split.

Figure 7: Evolution of Reptile loss and accuracy during training in type split.

Figure 8: Evolution of Hypernetwork loss and accuracy during training in type split.

K-Shot / N-Way	1	2	3	4	5
2	91.8%	93.67%	95.8%	96.4%	97.5%
3	87.5%	90.2%	93.2%	93.4%	96.7%
4	85.6%	88.4%	90.8%	92%	93.4%
5	82.4%	86.7%	88.8%	91.5%	92.2%

Table 8: Baseline test accuracy results in type split.

K-Shot / N-Way	1	2	3	4	5
2	52.5%	54%	61.5%	66%	69.5%
3	37.5%	39%	35.5%	44.7%	43.3%
4	33.3%	35.3%	35.3%	37.6%	40.2%
5	30.3%	32.8%	35.8%	35.8%	35.8%

Table 9: Reptile test accuracy results in type split.

K-Shot / N-Way	1	2	3	4	5
2	87%	89%	88%	91.5%	84%
3	80%	86.33%	90%	86.33%	84.33%
4	81.5%	79.75%	86.75%	88%	88.25%
5	78.6%	83.2%	83.6%	91.6%	84.4%

Table 10: Hypernetwork test accuracy results in type split.

In the Type Split experiment, the Baseline (Table 8) proves highly robust to semantic constraints, virtually mirroring its Generation Split performance; it maintains a linear scalability where increasing support from 1-Shot to 5-Shot in 5-Way tasks consistently adds ∼10 percentage points of accuracy (82.4% to 92.2%). Conversely, Reptile (Table 9) remains in the lower performance tier similar to previous experiments, fluctuating between 30.3% and 38.5% in 5-Way settings with negligible gains from additional shots, confirming its inability to resolve high inter-class similarity. The HyperNetwork (Table **??**), however, shows a distinct sensitivity to this specific split compared to the Generation experiment: while it starts strong at 78.60% (1-Shot), its scalability is dampened, achieving only a 5.8 percentage point gain by 5-Shot

(capping at 84.40%), which is notably lower than the ∼10 point improvement observed in the Generation Split.

4.3 Experiment 3: The Semantic Leap (Evolution Task)

The third experiment was designed to assess whether the models could extract robust, high-level features from the sprites, rather than relying solely on superficial attributes like color or texture. Our primary objective was to determine if the models had achieved a semantic understanding of 'Pokémon biology', specifically, the ability to recognize morphological consistency across different evolutionary stages.

To test this, we utilized the models pre-trained on the standard classification task (Experiment 1: Random Split) and applied them to a more challenging problem: Evolutionary Line Classification. In this task, the model must identify which evolutionary family a specific Pokémon belongs to by comparing it against a support set of its pre- or post-evolutionary forms. This requires the model to bridge the visual gap between distinct stages (e.g., linking a small Bulbasaur to a large Venusaur), proving it can recognize the underlying biological traits that define a species. The tables below present the accuracy metrics across varying N-Way (number of families) and K-Shot (number of support examples) configurations. In these results, we specifically look for evidence of feature transferability: high accuracy in low-shot scenarios (e.g., 1-Shot) would indicate that the model has successfully learned to generalize the concept of a 'Pokémon family' from minimal data, effectively identifying the morphological traits shared between different evolutionary stages.

4.3.1 Outcome

N-Way \ K-Shot	1	2	3	4	5
2	58.83%	58.2%	60.9%	60.5%	62.2%
3	44%	44.89%	45.5%	45.9%	47%
4	34.4%	35.08%	35.7%	38.75%	36.6%
5	28.23%	29.4%	30.17%	29.97%	31.7%

Table 11: Baseline + Finetuning results for Experiment 3 with model from task 1

N-Way \ K-Shot	1	2	3	4	5
2	41.50%	46.00%	45.75%	46.25%	43.50%
3	28.67%	32.00%	31.83%	29.67%	32.67%
4	23.88%	25.50%	26.00%	25.37%	27.50%
5	19.40%	20.90%	20.90%	21.20%	22.70%

Table 12: Reptile results for Experiment 3 with model from task 1

N-Way \ K-Shot	1	2	3	4	5
2	70%	64%	73%	69.5%	77.5%
3	54%	66%	57.67%	57.33%	61.33%
4	48.50%	51.25%	52.75%	46.75%	55.75%
5	44%	45.2%	46.6%	50.2%	51.2%

Table 13: Hypernetwork results for Experiment 3 with model from task 1

In the Evolutionary Line task, the Hypernetwork (Table 13) demonstrates the highest overall efficacy, achieving robust 2-Way accuracies (70%–77.5%) and maintaining a substantial margin in complex 5-Way settings (44%–51%) with a general positive correlation to shot count. In comparison, the Baseline (Table 11) reveals a sharper performance degradation as complexity increases, dropping from a stable 58.83%–62.2% window in 2-Way tasks to the 28%–31% range in 5-Way configurations, though it retains a consistent modest improvement (3–4%) with additional shots. Reptile (Table 12) exhibits the most limited adaptation, peaking at only 46.25% in 2-Way scenarios and falling to 19%–22% in 5-Way settings, characterized by variable sensitivity to support set size rather than the steady gains observed in the other models.

4.4 Experiment 3.1: The Semantic Leap (Evolution Task) with Oak Sampler

The results from the previous phase revealed the limitations of standard transfer learning: models trained to distinguish random species struggle to spontaneously grasp the subtle morphological nuances of evolutionary families. To address this, we designed a second experimental phase focused on Domain-Specific Meta-Training.

In this setup, rather than reusing the weights from Experiment 1, we re-trained all three architectures (Baseline, Reptile, and HyperNetwork) from scratch using the specialized data sampling strategy: "Oak Sampler".

Detailed visualizations of the training dynamics, including the evolution of loss and accuracy for these models, can be found in Appendix A.1 (see Figures 12, 13 and 14).

4.4.1 Outcome

N-Way \ K-Shot	1	2	3	4	5
2	79%	77.92%	80.25%	81.17%	81.92%
3	67.78%	70.44%	70.50%	71.94%	73%
4	60.04%	61.88%	63.50%	65.50%	66.79%
5	58.00%	61.40%	59.10%	59.80%	60.37%

Table 14: Baseline + Finetuning test accuracy results for Experiment 3.1 (trained model)

N-Way \ K-Shot	1	2	3	4	5
2	46.00%	53.00%	56.00%	56.00%	53.00%
3	31.67%	27.33%	34.00%	38.00%	36.00%
4	26.75%	27.50%	27.50%	28.75%	28.25%
5	22.20%	23.00%	24.60%	28.20%	29.40%

Table 15: Reptile test accuracy results for Experiment 3.1 (trained model)

N-Way \ K-Shot	1	2	3	4	5
2	69%	60.5%	66%	60%	68.5%
3	59.33%	61%	61.33%	52.33%	57.33%
4	47.25%	46.5%	48.25%	50%	51.75%
5	45.80%	43.40%	44.20%	40.80%	46%

Table 16: Hypernetwork test accuracy results for Experiment 3.1 (trained model)

Training models directly on the evolutionary task yields significant gains for gradient-based methods. The Baseline (Table 14) achieves the highest accuracy across all experiments, doubling its transfer-learning performance in 5-Way scenarios (hovering around 58%–60%) while maintaining highly linear scalability up to nearly 82% in 2-Way tasks.

Reptile (Table 15) also exhibits improved stability over its pre-trained counterpart, demonstrating a monotonic performance climb in 5-Way settings (rising from 22.20% to 29.40%) and solidifying its grasp on binary classification with peaks of 56%.

In contrast, the Hypernetwork (Table 16) displays a divergent trend, performing comparably to—or slightly worse than—its transfer-learning configuration. Its accuracy ranges from 60%–69% in 2-Way and 40%–46% in 5-Way tasks, but unlike the Baseline, it suffers from metric volatility (e.g., performance drops in 2-Way 2-Shot) and lacks a clear upward trajectory in high-complexity settings, suggesting that task-specific training offers diminishing returns for this architecture compared to the other models.

5 Discussion

5.1 Paradigm Trade-offs (Answering RQ1)

We used this first experiment as the battleground to compare the fundamental philosophies of Few-Shot Learning: Transfer Learning (Baseline), Gradient-Based Optimization (Reptile), and Model-Based Generation (HyperNetwork). The results from Tables [X, Y, Z] reveal a distinct trade-off between adaptation speed and asymptotic performance, effectively answering Research Question 1.

Contrary to our initial expectation that specialized meta-learning algorithms would dominate, the Baseline emerged as the most robust performer, particularly in the 5-Shot regime (reaching 92.4% in 5-Way). The success of the Baseline confirms that for standard tasks where the training and testing domains share the same underlying statistics (Random Split), a robust feature extractor (f_θ) is more critical than the adaptation mechanism itself. The features learned during pre-training transferred seamlessly. The "Test-Time Training" strategy (12 steps of Adam with Augmentation) proved to be a formidable adaptation mechanism. Unlike Reptile, which is constrained by its initialization, the Baseline's linear head is initialized from scratch, allowing it the plasticity to form decision boundaries freely without "unlearning" previous biases.

The HyperNetwork provided the most distinct behavioral profile. It dominated the 1-Shot setting (80.6% in 5-Way), validating the hypothesis that generative models excel when data is scarce. Because the HyperNetwork generates weights in a single pass, it does not require gradients to "find" the solution; it constructs it. This makes it immune to the noise of a single support image, treating it as a prototype rather than a data point to optimize over. However, the results show a "Generative Ceiling." As K increases from 1 to 5, the HyperNetwork's improvement is marginal compared to the Baseline. This is inherent to its architecture: the model aggregates the Support Set into a single mean context vector c. Averaging 5 embeddings reduces noise but does not provide the same granular supervision signal that gradient descent exploits in the Baseline. The HyperNetwork is a "fast learner" but lacks the capacity to refine its solution iteratively as more data becomes available.

Reptile struggled significantly, particularly as the number of classes (N-Way) increased, dropping to 32% in difficult settings.

Here we can observe the "Centroid" Problem, where Reptile seeks an initialization θ that is "close" to all tasks. In the high-variance space of Pokémon sprites, where visual diversity is extreme (e.g., distinguishing a round Jigglypuff from a geometric Porygon), finding a single optimal starting point may be mathematically impossible. The "centroid" of the manifold might be in a region of high loss for all specific tasks. The poor performance suggests that the few gradient steps allowed during meta-testing were insufficient to traverse the loss landscape from the generic initialization to the specific task solution. Unlike the Baseline, which only optimizes a simple linear head, Reptile must optimize the entire deep network, which requires more data and stability than the few-shot regime provided. The results suggest a clear hierarchy of utility:

- **For extreme data scarcity (1-Shot):** The HyperNetwork is superior due to its stability and generative nature.
- **For precision with slightly more data (5-Shot):** The Baseline (Transfer Learning) wins, proving that simple fine-tuning is hard to beat when features are robust.
- **Optimization-based methods (Reptile)** proved too brittle for this specific dataset, highlighting the risk of relying on a "universal initialization" in high-variance domains.

5.2 Robustness to Ambiguity (Answering RQ2)

The core research question of this experiment was whether meta-learning algorithms could maintain performance when facing high inter-class similarity. Our results indicate a positive answer: while there is a performance decrease compared to the random baseline, this drop is relatively small across all models. This suggests that the models are generally resilient to semantic grouping, preventing the expected catastrophic failure in "Hard Mode." We observed distinct behaviors in how each architecture handled the transition from Random to Type-based splitting. The baseline model demonstrated the highest robustness. It achieved the best absolute performance in the Type split (reaching 92% in 5-way 5-shot) with a negligible performance drop compared to the random split. This indicates that the feature extractor, pre-trained on the larger dataset, learns sufficiently generic features that (when combined with test-time finetuning) are not significantly affected by the semantic similarity of the classes. The "Hard Mode" did not compromise its effectiveness.

The Hypernetwork maintained a strong middle ground, achieving solid accuracy in the Type split (84% in 5-way 5-shot). Although it showed a slightly larger performance gap than the Baseline when moving to the semantic split, the drop remained contained. The dynamic weight generation proved capable of adapting to the specific characteristics of the "Types," even if it couldn't fully match the raw capacity of the finetuned baseline.

The Reptile algorithm exhibited the highest sensitivity to the distribution shift. It experienced the largest performance drop among the three methods (approximately 11% decrease in the Type split relative to Random) and yielded the lowest absolute accuracy (38% in 5-way 5-shot). While it did not completely collapse, the 11% drop highlights that

Reptile's initialization-based approach is less stable when the tasks within a batch share high semantic similarity, making it the least robust candidate for this specific "Hard Mode" scenario. Contrary to our initial hypothesis that the Generation split would constitute a "Hard Mode," our results revealed a counter-intuitive phenomenon: all models performed better on the Generation split than on the Random split. We attribute this to two compounding factors:

1. Visual Homogeneity (Style Bias): Since Pokémon generations are defined by distinct artistic eras (e.g., pixel art in Gen 1 vs. 3D models in Gen 5), the images within a task share a high degree of visual stylistic coherence. This likely allowed the models to exploit low-level texture statistics as "shortcuts," simplifying the feature extraction process.
2. Structured Variance vs. Stochastic Noise: A critical, often overlooked factor is the stability of the task distribution. In the random split, the formation of tasks is entirely stochastic. This can lead to "noisy" episodes with high intra-task variance, where the model must simultaneously process conflicting domains (e.g., a noisy pixel image next to a clean 3D render) or deal with unrepresentative samples that appear by chance.By dividing by Generation, we implicitly enforce distributional consistency. This ensures that the variance within an episode is controlled. The model is not exposed to chaotic "outlier" batches that might occur in a purely random selection process. This structural regularity likely helps Reptile find a more stable optimization path (θ), as the gradients produced during meta-training are less noisy compared to the highly entropic Random split.

Analyzing the training accuracy trajectories (Figures 3–8, provides further insight into the distinct learning mechanisms of each architecture.

The training metrics for Reptile were notably lower and more volatile compared to the other models. This is consistent with the algorithm's objective: rather than minimizing loss for a specific batch of tasks (which would look like high training accuracy), Reptile seeks a saddle point in the parameter space that is "close" to all tasks. The lower training accuracy reflects the model's struggle to satisfy conflicting gradients from diverse classes simultaneously. However, as seen in the Generation split results, this "compromise" initialization allows for rapid adaptation when the test domain is cohesive.

Conversely, the Hypernetwork displayed high training accuracy (70-90%). This reflects its generative nature: the network learns to map task embeddings to specific weights. While this allows it to "solve" the training set effectively (high accuracy), it introduces a risk of meta-overfitting, where the model memorizes the training tasks rather than learning generalizable learning rules. This explains why, despite high training scores, it struggled to match the Baseline's generalization in the unseen "Type" split.

The Baseline presented a unique phenomenon: a validation accuracy of 0% throughout the training phase. This is an evaluation artifact stemming from the structural difference between the pre-training objective (global classification over all classes) and the episodic few-shot metric used for logging. While the validation metric failed to capture the model's progress due to label misalignment or lack of test-time finetuning during the epoch loop, the final test results (reaching 97%) confirm that the model was effectively minimizing the training loss. This discrepancy highlights that for the Baseline, the learning of a robust, static feature extractor occurred "silently" regarding the few-shot metric, yet ultimately produced the most discriminative representations for the target task.

5.3 Learning Semantics vs. Visuals (Answering RQ3)

This experiment represents a "Hard Transfer" scenario. The models were trained on Experiment 1 (Random Split), where their objective was to distinguish every individual Pokémon species (e.g., separate a Charmander from a Charmeleon). In Experiment 3 (Oak Task), the objective flipped: they now had to group visually distinct Pokémon into the same family. The results (Tables 1, 2, and 3) reveal a fascinating hierarchy of adaptability: Hypernetworks ¿ Baseline ¿ Reptile. The core reason why this task is difficult, and why the results vary so much, is the conflict between Discriminative Training and Generalization. During training Phase (Exp 1) the models were penalized if they confused a Pichu with a Pikachu. They learned to focus on the specific shape and size differences that separate evolutionary stages. However, in testing Phase (Exp 3) the models are now rewarded for grouping Pichu and Pikachu. To succeed, they must effectively "ignore" the shape differences they worked so hard to learn and focus instead on shared traits (like color palette, eye style, or texture). This creates a feature conflict. The models have learned to separate these images in the latent space, but now they must draw a decision boundary that circles them together. The Hypernetwork (Table 3) is the clear winner, achieving 70% accuracy in the 2-Way 1-Shot setting, significantly outperforming the others.

The Hypernetwork does not rely on a fixed set of weights like the Baseline. Instead, it generates weights based on the input Support Set. When shown a Bulbasaur and a Venusaur as the "same class" in the support set, the Hypernetwork can instantly generate a specific classifier that emphasizes their shared features (e.g., the teal color) and suppresses the conflicting features (size/flower). It also likely learned a robust "Task Embedding" during Experiment 1. Even though the task changed, the mechanism of comparing images remained efficient. It allowed the model to "reprogram" itself for the new biological grouping rule much faster than gradient descent could update a static model. The Reptile model (Table 2) performed the worst, with accuracies hovering around 41-46% for the 2-Way task, barely above random chance in some complex configurations.

Initialization Bias: Reptile learns an optimal initialization (θ) that is "easy to fine-tune." However, this initialization was optimized for Species Classification (separating evolutions).

The "Unlearning" Problem: Because the initialization is so good at separating Charmander from Charizard, the fine-tuning process (which only consists of 12 gradient steps) has to fight against this bias. It has to "unlearn" the separation before it can learn the grouping. The gradient steps are simply not enough to traverse the loss landscape

from a "Separator" solution to a "Grouper" solution.

Rigidity: unlike the Hypernetwork, which dynamically shifts its state, Reptile is constrained by the geometry of its initial weights. If the optimal weights for "Pokémon Biology" are far away from the optimal weights for "Species Identification," Reptile cannot reach them in time.

The Baseline sat in the middle, achieving 58-62% on the 2-Way task. The Baseline uses a standard feature extractor with a new linear head trained at test time. While the feature extractor is static (and likely separates evolutions), the Linear Head is initialized from scratch and has high plasticity. It turns out that even if the backbone separates Charmander and Charizard, their features are likely still closer to each other than to Squirtle. A fresh linear classifier can draw a rough boundary around the "Fire Type" region of the feature space more easily than Reptile can adjust its entire deep network. Here we can also see the impact that data availability has on the different architectures. In the Hypernetwork, we see a massive jump in accuracy as K-Shot increases (e.g., 70% $\rightarrow$ 77.5%). This confirms the model is successfully using the extra data to refine its understanding of the "Family" concept. Differently, in reptile the performance is erratic and flat (e.g., 2-Way accuracy fluctuates between 41% and 46%). This indicates that the model is confused. Giving it more images of the evolutions doesn't help because the model's internal initialization is fundamentally fighting the task. The results suggest that standard meta-learning (Reptile) is brittle when the semantic definition of the task changes (from "Identity" to "Family"). In contrast, Hypernetworks demonstrate a "meta-cognitive" ability: they can look at the support set and realize, "Oh, in this episode, we are grouping by color/family, not separating by shape," and adapt the network accordingly.

6 Conclusions & Future Work

6.1 Summary of Contributions

This study investigated the efficacy of distinct meta-learning paradigms, Transfer Learning, Gradient-Based Optimization, and Model-Based Generation, within the high-variance domain of Pokémon evolution. By systematically varying the availability of data (1-Shot vs. 5-Shot) and the semantic objective (Species Classification vs. Family Grouping), we arrived at three critical conclusions regarding the trade-offs in few-shot learning. First, regarding paradigm trade-offs (RQ1), our results established a clear hierarchy of utility based on data scarcity. The HyperNetwork demonstrated superior stability in extreme scarcity (1-Shot), leveraging its generative nature to construct solutions without the noise sensitivity of gradient descent. However, the Baseline (Transfer Learning) proved that a robust feature extractor combined with test-time fine-tuning remains the dominant strategy when even marginal data (5-Shot) is available. This mirrors findings by Chen et al. [1], who demonstrated that a well-tuned baseline can consistently outperform complex optimization-based meta-learners. Conversely, Reptile (Gradient-Based) struggled significantly, highlighting the fragility of relying on a "universal initialization" in domains with extreme visual diversity (e.g., distinguishing geometric Porygon from organic Jigglypuff). Second, regarding semantic adaptability (RQ3), we identified a "Biology Gap" where models trained to distinguish species struggled to group them by family. In this "Hard Transfer" scenario, the HyperNetwork emerged as the most "meta-cognitive" architecture. Its dynamic weight generation allowed it to resolve the feature conflict, shifting from a "separator" to a "grouper", far more effectively than gradient-based methods, which required extensive "unlearning" of previous biases. Finally, our investigation into task-specific training revealed a counter-intuitive "Lazy Learner" paradox. While the Baseline excelled when trained directly on families due to task alignment, the HyperNetwork performed worse than its transfer-learning counterpart. This suggests that training on a harder, more granular task (Species ID) forces the model to learn richer, high-resolution features that generalize better than the coarser features learned from easier, family-level objectives. These findings confirm that there is no single "silver bullet" architecture; finding the best model will always be a calculated trade-off between the stability required for robust feature extraction and the plasticity needed for rapid, low-data adaptation.

6.2 Future Work

The constraints and discoveries of this project point toward several promising avenues for future research in few-shot classification.

Multimodal Few-Shot Learning (Pokedex Integration): Pokémon data is inherently multimodal, consisting of visual sprites and rich textual descriptions (Pokedex entries). In future work, we could integrate Natural Language Processing (NLP) to condition the image classifier. For example, using the keywords "flame tail" or "turtle shell" to guide its visual attention, potentially bridging the semantic gap between visually distinct evolutionary stages (e.g., Remoraid to Octillery) that pure computer vision struggles to resolve.

Hierarchy-Aware Meta-Learning: Our experiments treated classes as flat categories, but biological data is hierarchical (Species $\rightarrow$ Family $\rightarrow$ Type $\rightarrow$ Egg Group). We propose employing a 'Hyperbolic Prototypical Network' to explicitly model these phylogenetic relationships. By embedding images into a hyperbolic space, a model might better capture the "parent-child" relationship inherent in evolutionary lines.

Curriculum-Based Meta-Training: Addressing the "Lazy Learner" paradox observed in Experiment 4, we could pre-train the model a curriculum learning strategy. The model could be pre-trained on the granular "Hard Task" (Species ID) to develop robust feature extractors, and then gradually transition to the "Abstract Task" (Family Grouping). This could theoretically combine the rich feature representations of the HyperNetwork with the task-aligned stability of the Baseline.

Generative Data Augmentation: Given the success of the HyperNetwork in 1-Shot scenarios, a hybrid approach could be explored where a Generative Adversarial Network (GAN) or Diffusion Model is used to "hallucinate" additional support samples for the Baseline. If a model sees one Charmander, it could generate synthetic variations (poses, lighting), artificially converting a 1-Shot problem into a 5-Shot problem to unlock the Baseline's superior asymptotic performance.

References

[1] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[3] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[4] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[5] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[6] PokéAPI. Pokéapi.

[7] Sam Santala. How pokemon's art style & design has 'evolved', 2 2022.

[8] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017.

[9] Tamamushi University. The philosophy of pokémon design, 12 2021.

[10] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[11] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[12] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

A Appendix

A.1 Training Dynamics and Loss Curves for Experiment 1

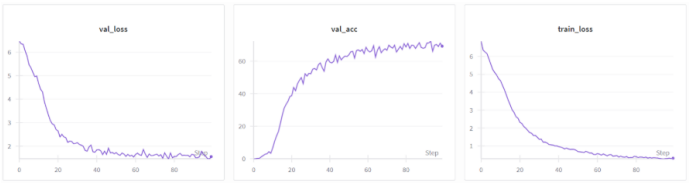

Figure 9: Evolution of Baseline loss and accuracy during training in random split.

Figure 10: Evolution of Baseline loss and accuracy during training in random split.

Figure 11: Evolution of Reptile loss and accuracy during training in random split.

A.2 Training Dynamics and Loss Curves for Experiment 3.1

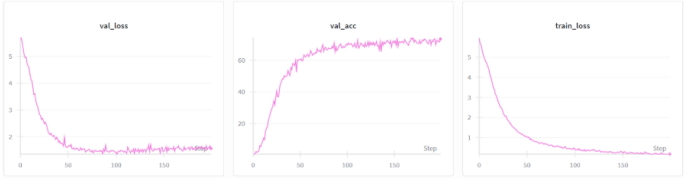

Figure 12: Evolution of Baseline loss and accuracy during training in evolution task.

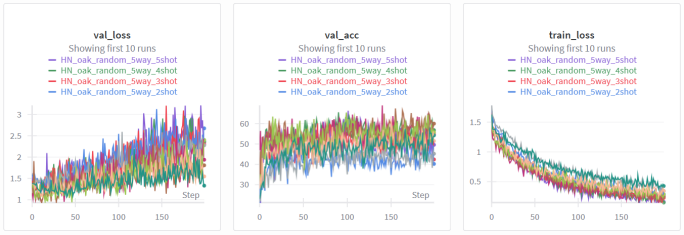

Figure 13: Evolution of Hypernetwork loss and accuracy during training in evolution task.

Figure 14: Evolution of Reptile loss and accuracy during training in evolution task.