Language to Network: Conditional Parameter Adaptation with Natural Language Descriptions

Tian Jin[*]
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598
tian.jin2@ibm.com

Zhun Liu[*◇]
Microsoft
Bellevue, WA 98004
zhunliu@microsoft.com

Shengjia Yan
Tandon School of Engineering
New York University
Brooklyn, NY 11201
sjyan@nyu.edu

Alexandre Eichenberger
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598
alexe@us.ibm.com

Louis-Philippe Morency
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
morency@cs.cmu.edu

Abstract

Transfer learning using ImageNet pre-trained models has been the de facto approach in a wide range of computer vision tasks. However, fine-tuning still requires task-specific training data. In this paper, we propose **N^3** (**N**eural **N**etworks from **N**atural Language) - a new paradigm of synthesizing task-specific neural networks from language descriptions and a generic pre-trained model. **N^3** leverages language descriptions to generate parameter adaptations as well as a new task-specific classification layer for a pre-trained neural network, effectively "fine-tuning" the network for a new task using only language descriptions as input. To the best of our knowledge, **N^3** is the first method to synthesize entire neural networks from natural language. Experimental results show that **N^3** can out-perform previous natural-language based zero-shot learning methods across 4 different zero-shot image classification benchmarks. We also demonstrate a simple method to help identify keywords in language descriptions leveraged by **N^3** when synthesizing model parameters.[1]

1 Introduction

A person with generic world knowledge can learn to perform a new task based on verbal instructions. On the other hand, despite recent successes in deep learning, it remains challenging to re-purpose pre-trained visual classification models to recognize a new set of objects without labeling a new training dataset. A natural question emerges from this observation: can a computer also learn to recognize new objects, simply by reading the descriptions of them in natural language? Concretely, can we create visual classifiers using language descriptions of the objects of interest?

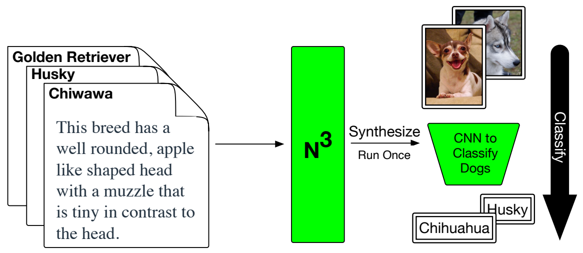
Golden Retriever
Husky
Chiwawa
This breed has a well rounded, apple like shaped head with a muzzle that is tiny in contrast to the head.
N3
Synthesize
Run Once
CNN to Classify Dogs
Husky
Chihuahua
Classify

Figure 1: An illustration of N^3. A list of object class descriptions is fed to the N^3 model to produce a CNN classifier that can classify images belonging to the object classes described.

In this paper, we introduce a new paradigm for synthesizing task-specific neural networks for image classification simply from language descriptions of the relevant objects. We propose **N^3** - **N**eural **N**etworks from **N**atural Language, a **meta-model** that takes a list of object descriptions as input to produce a classification model for these objects, as illustrated in Figure 1. The capability of producing task-specific neural network from lan-

[*]First two authors contributed equally.

[◇]This work was performed when Zhun Liu was affiliated with Carnegie Mellon University.

[1]Code is released at https://github.com/tjingrant/n3cr .

guage descriptions makes N^3 ideal to a wide range of zero-shot tasks (Wah et al., 2011; Zhu et al., 2017; Elhoseiny et al., 2017; Zhu et al., 2017; Nilsback and Zisserman, 2006) where we do not have visual training data but can still easily obtain language descriptions of the objects of interest.

Prior zero-shot learning methods aim to achieve a similar goal of applying pre-trained networks on unseen classes. In zero-shot image classification, a typical method of generalizing to unseen classes is to construct class embeddings to augment the classification layer of the pre-trained network or take retrieval-based approaches while utilizing generic visual features from pre-trained networks (Akata et al., 2015; Kodirov et al., 2017; Elhoseiny et al., 2013; Lei Ba et al., 2015; Reed et al., 2016).

Extending on the idea of generating classification layers for the pre-trained network, N^3 modifies the parameters of all layers in the pre-trained network. While the underlying pre-trained network in previous approaches only extracts generic visual features, N^3 makes it possible to extract task-specific ones. This approach effectively increases the capacity at which semantic information in the descriptions can affect the pre-trained network.

In our experiments, we evaluated our proposed N^3 method with 4 popular zero-shot image classification benchmark datasets. We performed ablation studies to understand the importance of synthesizing task-specific feature extractors, the necessity of a pre-trained visual classification model and the effects of language representation choices on the efficacy of **N^3**. In addition, we provide a simple approach to help interpret what aspects in the language descriptions are N^3-generated models examining when making predictions.

To summarize, our contributions are 3-fold:

1. We propose a novel meta-model N^3 to synthesize task-specific neural network models using natural language descriptions.

2. We demonstrate N^3's superior efficacy in solving natural language guided zero-shot learning problems. Our analysis shows that N^3's ability to tailor neural network models to extract task-specific features plays an important role in achieving such superior accuracy.

3. We show that N^3 can aid the interpretation of predictions of synthesized models as they can be traced back to both supportive and refutative evidence within language descriptions.

2 Related work

In this section, we are situating our work in the context of zero-shot learning and dynamic parameter generation for neural networks.

Zero-shot Learning Zero-shot learning studies how we can generalize our models to perform well for tasks without any labeled training data at all. Achieving classification accuracy above chance-level in such scenarios requires modeling the relationships between the seen classes during training and the unseen classes during testing. A typical and effective method is to manually engineer class attribute vectors to obtain representations of seen and unseen classes in a shared attribute space (Duan et al., 2012; Kankuekul et al., 2012; Parikh and Grauman, 2011; Zhang and Saligrama, 2016; Akata et al., 2016). Yet such a method requires laborious engineering of class attributes, which is not feasible for large-scale and/or fine-grained classification tasks (Russakovsky et al., 2015; Khosla et al., 2011; Welinder et al., 2010). Hence, there is also work in zero-shot learning that attempts to leverage textual data as object class representations (Lei Ba et al., 2015; Elhoseiny et al., 2013). The majority of these models are committed to embedding-based retrieval approaches, where classification is re-formulated as retrieving the class embedding with maximal similarity (Akata et al., 2015; Kodirov et al., 2017). While they can handle well the case where there is an indefinite number of classes during test time, such approaches suffer from extra computation cost at inference time since they need to traverse all the seen data points. Moreover, these models often rely on a pre-trained feature extractor for input, which is usually fixed and cannot be further adapted for the unseen classes (Akata et al., 2015; Kodirov et al., 2017; Elhoseiny et al., 2013). While there is a handful of work that tries to modify the parameters in-place during test time, they either rely on shallow language representation with limited expressiveness (Lei Ba et al., 2015) or require a significant amount of textual descriptions per class to train their model (Reed et al., 2016), both of which are not ideal. In our work, we aim to learn a model that can synthesize parameters for entire neural networks to adapt to the new tasks using short descriptions of the object classes. This leads to better metadata efficiency of our proposed method.

Dynamic Parameter Generation As mentioned before, N^3 dynamically generates classification models for designated classes. Dynamic parameter generation has been explored in the context of generating recurrent cells at different time-steps of RNNs (Ha et al., 2016), constructing intermediate linear models for interpretability inside neural networks (Al-Shedivat et al., 2017), and contextual parameter generation for different language pairs in multilingual machine translation (Platanios et al., 2018). As mentioned in Section 2, some zero-shot learning methods can also be viewed as generating classifier parameters (Lei Ba et al., 2015; Elhoseiny et al., 2013). However, many of the previous work directly or indirectly mentions the challenge of memory and computation complexity - after all, the output of the parameter generation model are large matrices that are excessively high dimensional. To tackle this issue, previous work either only generate very simple linear layers (Lei Ba et al., 2015; Elhoseiny et al., 2013; Al-Shedivat et al., 2017), or impose low-rank constraints on the weights to mitigate the memory issues (Ha et al., 2016). In our work, we utilize the architecture of sequence-to-sequence models and treat the weight matrices to be generated as a sequence of vectors. This allows parameter generation for entire neural networks with little memory bottleneck.

3 N^3 Methodology

In this section, we describe our approach for synthesizing task-specific neural networks to recognize new objects with their natural language descriptions and a generic pre-trained model. We denote a list of natural language descriptions for K objects, each containing L tokens as $D = \{d_{k,l}\}_{k=1,\ l=1}^{k=K,\ l=L}$. We denote a pre-trained model with parameters Θ as $\mathcal{F}(\cdot;\Theta)$, so that for a set of images X, $\mathcal{F}(X;\Theta)$ produces the classification prediction Y.

We can now precisely formulate our problem as follows: given a pre-trained classification model, $\mathcal{F}(\cdot;\Theta)$, and the natural language description of K object classes, D, adapt the original parameters Θ to the specialized parameters Θ' so that the fine-tuned $\mathcal{F}(\cdot;\Theta')$ model accurately classifies the K objects described in D.

3.1 Synthesizing Task-specific Models via Parameter Adaptation

Our method draws inspiration from transfer learning, which is often employed when the training dataset is small. Transfer learning entails training a neural network from a generic pre-trained model to one that solves a new, often task-specific problem. Thus, we can similarly formulate N^3 to synthesize task-specific model parameters by adapting existing ones in the generic pre-trained model with the guidance of natural language task descriptions. While transfer learning updates the pre-trained parameters using signals derived from task-specific training data, N^3 relies only on language descriptions to achieve the same objective. Concretely, the adapted parameters Θ' are computed as follows:

$$\Theta' = \Theta + \mu \cdot \Phi(D;\Gamma) \tag{1}$$

where $\Phi(\cdot;\Gamma)$ is a function with parameter Γ, mapping natural language descriptions to **parameter adaptations** for all parameters in the pre-trained model. Since the transfer learning process often proceeds with a tiny learning rate to restrict the effect of fine-tuning, we introduced a trainable scaling factor μ to similarly regulate the effect of parameter adaptation. The initial value of μ is a hyper-parameter. In our experiments, we used an initial μ value of $1e^{-3}$ to mimic the effect of using a small learning rate for transfer learning. The specific value of $1e^{-3}$ is derived from the default learning rate used in the PyTorch transfer learning tutorial (Chilamkurthy). In a later section, we evaluate the necessity of this scaling factor as well as the effect of a range of initial values of μ.

3.2 Making Adaptation Computationally Feasible via Hierarchical Attention and Layer Sharing

The mapping Φ from natural language descriptions to parameter adaptations is particularly high-dimensional. Constructing Φ with the transformer block (Vaswani et al., 2017) is not straight-forward for our scenario because it requires $O(N^2)$ size of memory where N is the length of the input/output sequence and our $N = K \times L$ can be prohibitively large. Thus, to reduce the memory consumption of Φ, its attention span must be restricted to a small but semantically relevant subset of all input elements. To this end, we designed Φ to be a two-level hierarchy of transformer blocks, as illustrated in 2. The first level of transformers, named the **Tokens2Label Encoder**, encodes the natural language descriptions of each object class to a label embedding vector. Intuitively, this level summarizes the described visual features of an ob-

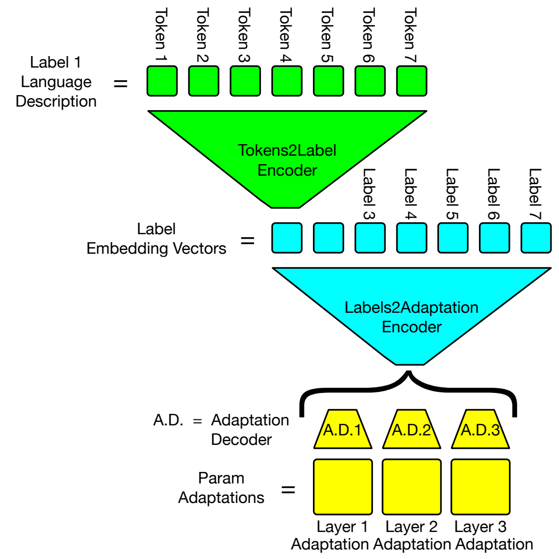

Figure 2: Hierarchical encoder for descriptions and decoder for parameter adaptations.

ject class to a single, fixed-sized embedding vector. Thus, attention in this level has a maximum span of L spanning all tokens in each description. The second level, named the **Labels2Adaptation Encoder-Decoder**, encodes the sequence of label embedding vectors and decodes them into parameter adaptations of multiple layers. This level of transformer blocks examines the characteristics of **all** encoded object classes and determines how to adapt the pre-trained model parameters to classify images corresponding to these object labels. In this level, the model attention has a maximum span of K spanning all the object classes. Moreover, due to the sheer number of layers in state-of-the-art CNN models, we initialize layer-specific adaptation decoders for each layer and decode from shared hidden states encoded by the adaptation encoder. Only through these measures can we materialize the high-dimensional mapping Φ under reasonable memory constraints.

Putting everything together, for a pre-trained network with parameter Θ, N^3 applies the mapping Φ to input descriptions to generate a parameter adaptation $\Delta\Theta$ with the same shape as Θ. The adaptation $\Delta\Theta$ is multiplied with a trainable scaling factor μ to control its impact on the pre-trained model. The scaled parameter adaptation $\mu \cdot \Delta\Theta$ is then combined with its pre-trained counterpart Θ via point-wise addition.

The mapping Φ from object descriptions D to $\Delta\Theta$ contains two parts. The Tokens2Label Encoder transforms the set of tokens contained in K entries of object class descriptions, each with length up to L, denoted as $\{d_{k,l}\}$, to a set of K object label embedding vectors $\{c_k\}_{k=1}^{K}$:

$$c_k = \texttt{Encoder}_{\texttt{T2L}}(d_{k,1:L}), k = 1, ..., K \quad (2)$$

Subsequently, Labels2Adaptation Encoder-Decoders translate the object label embedding vectors into a parameter adaptation matrix $\Delta\Theta$. Parameters in a typical neural network often have more than two-dimensions, but for simplicity in our setup, they are always viewed as a two-dimensional matrix consisting of a sequence of parameter columns. [1] Viewing the object-label embeddings $\{c_i\}_{i=1}^{k}$ as an input sequence and the parameter adaptation $\Delta\Theta$ as an output sequence of M columns $\{\Delta\theta_m\}_{m=1}^{M}$, Labels2Adaptation Encoder-Decoders can be expressed as:

$$\mathbf{h}_{1:K} = \texttt{Encoder}_{\texttt{L2A}}(c_{1:K}) \quad (3)$$
$$\Delta\theta_m = \texttt{Decoder}^{\texttt{m}}_{\texttt{L2A}}(\mathbf{h}_{1:K}), m = 1, ..., M \quad (4)$$
$$\Delta\Theta = \texttt{Concat}([\Delta\theta_1; \Delta\theta_2; ...; \Delta\theta_M]) \quad (5)$$

Finally, pre-trained parameter Θ is adapted to Θ' using the following equation, with μ being a trainable scaling factor:

$$\Theta' = \Theta + \mu \cdot \Delta\Theta \quad (6)$$

3.3 Training Methodology

We formulate the training of N^3 as the following optimization problem. Optimal parameters Γ for N^3 model $\Phi(\cdot; \Gamma)$ should map a list of language descriptions for K class objects $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$ to parameter adaptations $\Delta\Theta = \Phi(\mathcal{D}; \Gamma)$, such that the cross entropy loss between ground truth label Y and model prediction $\mathcal{F}(X; \Theta, \Phi(\mathcal{D}; \Gamma))$ is minimized for all image-label pairs (X, Y) in the training set.

Thus, to train N^3 on a image dataset $\mathcal{I}$ with labels $\mathcal{L}$, class descriptions $\mathcal{D}$ and a pre-trained model $\mathcal{F}$ to produce a K-way classification model, we first draw meta-batches of K class labels $\mathcal{L}_K \in \mathcal{L}$. Then, a subset of the image dataset $\mathcal{I}_{\mathcal{L}_K}$ and description dataset $\mathcal{D}_{\mathcal{L}_K}$ corresponding to the drawn labels $\mathcal{L}_K$ are constructed. We then draw mini-batches of images $\mathcal{B}$ and ground-truth labels Y

[1]For instance, the parameter of a convolutional layer W of shape $[K, C, kH, kW]$ where K is the number of output channels, C the number of input channels, kH, kW the kernel height and width, is viewed as a sequence of K parameter columns, with a column size of $C \times kH \times kW$.

from $\mathcal{I}_{\mathcal{L}_K}$. For each mini-batch $\mathcal{B}$, distinct parameter adaptations $\Delta\Theta$ are generated by evaluating $\Phi(\cdot;\Gamma)$ at $\mathcal{D}_{\mathcal{L}_k}$. Batch loss is then calculated as $\frac{1}{|\mathcal{B}|}\sum_i \ell(Y_i, \mathcal{F}(X_i;\Theta,\Delta\Theta))$ where $\ell(\cdot,\cdot)$ refers to cross-entropy loss. Since the meta-model $\Phi(\cdot;\Gamma)$ and the pre-trained model $\mathcal{F}$ are fully differentiable, gradients can be propagated back to meta-models to optimize meta-model parameters Γ.

4 Experimental Setup

We evaluate N^3 by comparing its efficacy in solving natural language guided zero-shot learning problems with prior state-of-the-art methods.

In this section, we introduce our training method, datasets and evaluation protocols.

4.1 Datasets

To evaluate the N^3, we select 4 standard zero-shot image classification datasets and collected natural language descriptions for their object classes.

Caltech-UCSD-Birds 200-2011 (CUB) (Wah et al., 2011) contains images of 200 species of birds. Each species of bird forms its own class label. In total, there are 11,788 images in this dataset.

Animal with Attributes (AWA) (Lampert et al., 2014; Xian et al., 2017) is another dataset to evaluate zero-shot classification methods. It consists of 50 classes of animals with a total of 37322 images.

North America's Birds (NAB) is a dataset used by prior state-of-the-art methods related to our task. Following the established practices (Zhu et al., 2017; Elhoseiny et al., 2017), we consolidated the class labels into 404 distinct bird species. The consolidation process combines closely related labels (e.g., 'American Kestrel (Female, immature)' and 'American Kestrel (Adult male)') into a single label (e.g., 'American Kestrel'). We end up with 48,000 images of 404 classes of bird-species.

Flowers-Species (FS) is a dataset we built based on Oxford Flowers (Nilsback and Zisserman, 2006), another commonly used zero-shot dataset. The original contains label categories that are a mixture of species and genera. Some genus includes thousands of species, yet the dataset examples only cover a fraction of them. Such mismatch creates biases in the dataset that fundamentally cannot be addressed through learning from external descriptions. This hence undermines its utility as a test of our proposed method: for instance, when N^3 is asked to generate classifier to decide whether an object is of label "anthurium", which is a genus of around 1000 species of varying visual appearance, the efficacy of our generated model can only be evaluated based on a representative samples that cover most of the species within the genus "anthurium". However, the dataset only contains a tiny number of (i.e., 105) correlated (species-wise) samples, making such evaluation neither comprehensive nor conclusive in the context of our task objective and may introduce unexpected noise in evaluation results. Therefore, we decided to filter out the genera from the original Oxford Flowers dataset, leaving only the species as class labels, as an effort towards homogenizing the sample spaces implied by the class labels and the image dataset. This leaves us with 55 classes and 3545 images.

For each dataset, we collect language descriptions for object classes from websites like Wikipedia. To collect language descriptions from Wikipedia, we use the python package Wikipedia (Goldsmith) to access structured representation of Wikipedia pages, and extract section content under "Description" to be used as object class descriptions. If no Wikipedia entry exists for a specific object class, we resort to manually searching for the object class description on Google. Furthermore, we truncate these textual excerpts to a maximum length of 512 tokens to avoid excessively long descriptions and the accompanying computational issues.

4.2 Evaluation Protocol

Recent study showed that previous zero-shot learning evaluation protocols are inadequate and proposed a set of rigorous evaluation protocols for attribute-based zero-shot learning methods (Xian et al., 2017). Although both our tasks and datasets differ, we nevertheless followed Xian et al.'s guiding principles of their *Rigorous Protocol* and developed our evaluation protocol:

- Similar to *Rigorous Protocol*, we used two meta-splits **Standard Splits (SS)** and **Proposed Splits (PS)** to evaluate all methods; the **Standard Splits** are established meta-splits and **Proposed Splits** are meta-splits that guarantees the exclusion of ImageNet-1K classes from the test set.

- Due to class imbalance, *Rigorous Protocol* proposes to use per-class averaged accuracy for more meaningful evaluation. Thus, to evaluate meta-model on a meta-split containing

Datasets	Total	Standard Split/Proposed Split		
		Training	Validation	Testing
CUB (Wah et al., 2011)	200	100	50	50
AWA2 (Zhu et al., 2017; Elhoseiny et al., 2017)	50	30	10	10
NAB (Zhu et al., 2017)	404	324	40	40
FS (Nilsback and Zisserman, 2006)	55	35	10	10

Table 1: Number of Class Labels in Our Meta-Split (both Standard Split and Proposed Split). Within each meta-split, training, validation and testing class labels are disjoint.

the set of classes C, we calculate the per-class averaged accuracy as shown in Equation 7.

$$Acc_C = \frac{1}{|C|}\sum_{c\in C}\frac{\text{\#correctly predicted samples in c}}{\text{\#total samples in c}} \tag{7}$$

- Unlike *Rigorous Protocol* which uses ResNet-101 model, we use ResNet-18 as our pre-trained model; such choice helps reduce the output dimensionality of N^3 by reducing the number of parameters N^3 adapts.

- Our method is unique in that permutations of the classes belonging to the same meta-split count as distinct tasks and therefore, to account for variations, we test our models on 10 different permutations of test set classes and report the medium value of the relevant evaluation metric.

We have tabulated the number of class labels used for training, validation and testing in Table. 1.

4.3 Baseline Models

PDCNN PDCNN (Lei Ba et al., 2015) is the most relevant prior method as it use the natural language descriptions of class labels to generate classification layers capable of distinguishing between objects described. Note that PDCNN is distinct from our work in that it dynamically generates fully connected layers and/or additional convolutional layers to be **appended** to a pre-trained deep neural network (VGG-19) whilst ours generates parameter adaptations to be **combined directly** with all existing layers within deep neural network models, effectively "fine-tuning" the pre-trained model. We compare with two variants of PDCNN, with PDCNN_{FC} generating a fully connected layer only for classification and $\text{PDCNN}_{\text{FC+Conv}}$ producing an additional convolutional layer to help with classification. To make our works comparable, we replaced the TF-IDF feature extractor in PDCNN with a BERT-based document embedding (specifically, a BERT token embedding followed by max-pooling) and changed the pre-trained model from VGG-19 to ResNet-18.

MEGAZSL Due to the scarcity of prior work leveraging natural language descriptions for zero-shot classifications in a metadata-efficient way, we also adapted less metadata-efficient methods to function with stricter metadata-efficiency requirements. Specifically, ZSLPP (Elhoseiny et al., 2017), GAZSL (Zhu et al., 2017) and Correction-Network (Hu et al., 2019) all utilize natural language object class descriptions to produce classifiers capable of distinguishing between images belonging to unseen categories during training. However, all of them require significantly more metadata: specifically, parts annotations of each sample image are used to provide extra supervision of the training procedure. Among these methods, GAZSL (Zhu et al., 2017) stands out as the most cited work; therefore we adapted the code released by its authors to learn from only natural language metadata, without using parts annotations; to distinguish our modified version from the original, we will refer to our modified version as MEGAZSL (**M**etadata-**E**fficient GAZSL). To make our works comparable, we also updated its language representation from TF-IDF to BERT-based ones and used ResNet-18 as the image feature embedding module. It is worth noting the CorrectionNet(Hu et al., 2019) is orthogonal to our work as it is designed to improve any existing zero-shot classification task modules, and in its original setup, GAZSL (Zhu et al., 2017) was used as the main task module, which we do include in our experimental comparison.

For all experiments, hyper-parameters are tuned

Method	CUB-50		AWA2-10		NAB-40		FS-10	
	SS	PS	SS	PS	SS	PS	SS	PS
PDCNN_{FC}(Lei Ba et al., 2015)	6.1	6.1	12.8	18.4	6.7	7.4	13.1	11.9
$\text{PDCNN}_{\text{FC+CONV}}$(Lei Ba et al., 2015)	7.5	6.5	22.6	17.2	8.9	5.6	7.7	13.6
MEGAZSL(Zhu et al., 2017)	2.9	1.8	14.0	10.4	2.8	3.7	10.3	12.3
N^3 (Proposed)	**17.6**	**9.5**	**34.0**	**37.5**	**14.1**	**20.7**	**16.4**	**17.6**

Table 2: Zero-Shot Classification Accuracy (Defined by Eq. 7) On Various Datasets/Meta-Splits Combinations; number of classification labels used in the test set are recorded next to the name of each dataset.

with the same exact algorithms (random search) and for the same number of runs (10).

5 Results and Discussion

In this section, we compare N^3 method with prior work on 4 zero-shot image classification benchmark datasets. Furthermore, we provide a simple approach to help interpret what part of the language input is being taken as evidence by the synthesized visual classifier to make predictions. We then show through ablation studies the importance of adaptation of all parameters in the pre-trained network, the necessity of pre-trained networks, and the effect of language representation choices on the success of N^3.

5.1 Benchmark Evaluations

We report performance in per-class averaged accuracy, as shown in Table 2. In these experiments, we standardized the language representations, dataset-splits, and other factors orthogonal to our model design to ensure fairness of comparison. We also include experimental results comparing N^3 to models in their respective original settings in the Appendix. From Table 2, we can clearly observe that N^3 outperforms all competing methods by a significant margin on all 8 dataset/meta-split combinations. Noticing the large performance gap between MEGAZSL and the original GAZSL, we performed additional investigations to pinpoint the cause: we reproduced one of their experiments (on CUB dataset with SCE meta-split), and replaced their TF-IDF module with BERT document embedding module. The modified module performed noticeably better (from 10.3% to 11.3%); however, when we replaced its image feature embedding module, which is trained with additional parts annotations of bird images, the performance dropped significantly (from 11.3% to 3.3%), confirming our conjecture that such methods cannot be easily adapted to work without extra supervision in the form of additional data annotation.

5.2 Interpreting Model Predictions

In this section, we explore how N^3 model architecture can help interpret the predictions made by the adapted model. Specifically, the design of N^3 is unique in that it examines **all** object class descriptions in order to adapt neural network parameters. This means that N^3 can adapt neural networks to seek both positive and negative evidence for an image to be classified. Naturally, we want to understand how the model is using these object class descriptions. In Figure. 3, we present our findings by visualizing the magnitude of $\frac{\partial E}{\partial D_{ij}}$ where E is the loss value computed on a test example (in our experiment, this example is correctly predicted to be an Acadian Flycatcher) and D_{ij} is the BERT representation of the j-th word in the i-th object class description. We present two patterns in the data that are indicative of the model behavior:

Top Positive Evidence: top positive evidence is identified as tokens in the description of the ground truth label with large gradients. Intuitively, these tokens are top supporting evidence that encourages the prediction of the correct label. We locate them by ranking all tokens in the description of the ground truth label by their magnitude of gradients and take the top few. **Top Negative Evidence**: top negative evidence is identified as tokens with large gradients descriptions of the negative labels that also has a low predicted probability. Intuitively, these tokens are the keywords deemed as important evidence when rejecting to predict a label. We locate such evidence by first ranking the descriptions by the ratio between their largest token gradients and the softmax probability of the corresponding label. Then, within the set of class descriptions with the largest aforementioned ratios, we locate the tokens with the largest gradients.

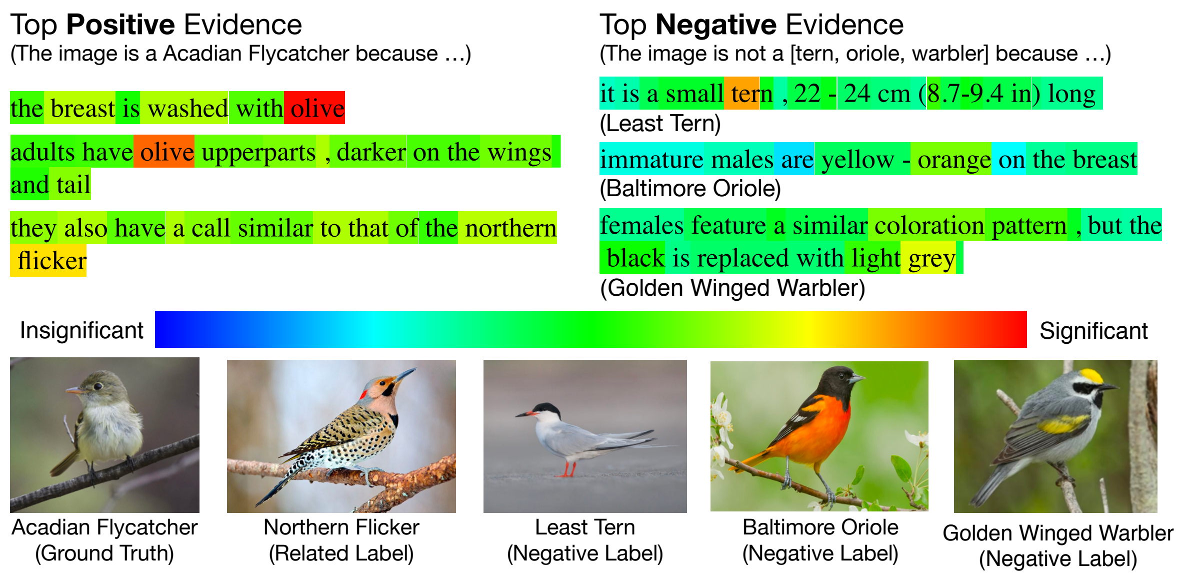
Top **Positive** Evidence
(The image is a Acadian Flycatcher because ...)
the breast is washed with olive
adults have olive upperparts , darker on the wings and tail
they also have a call similar to that of the northern flicker
Top **Negative** Evidence
(The image is not a [tern, oriole, warbler] because ...)
it is a small tern , 22 - 24 cm (8.7-9.4 in) long
(Least Tern)
immature males are yellow - orange on the breast
(Baltimore Oriole)
females feature a similar coloration pattern , but the black is replaced with light grey
(Golden Winged Warbler)
Insignificant
Significant
Acadian Flycatcher
(Ground Truth)
Northern Flicker
(Related Label)
Least Tern
(Negative Label)
Baltimore Oriole
(Negative Label)
Golden Winged Warbler
(Negative Label)

Figure 3: When classifying an image of Acadian Flycatcher, we visualized the magnitude of gradients of description tokens w.r.t the loss. Higher magnitude is colored red and lower magnitude blue. Intuitively, words with higher magnitude of gradient represent textual features that are more important for the classification decision.

Examples for both types of keyword evidence are presented in 3 with their context. Several observations can be drawn about how N^3 uses textual descriptions to make classification decisions. Firstly, we can observe that distinguishing features described with keywords such as "olive", "darker", "yellow" and "grey" are used to support both positive and negative identifications, which support our conjecture that N^3 examines descriptions from all object class descriptions to make classification decisions, sometimes employing the process of elimination. Secondly, we can observe that some label (or label word piece) like "flicker" and "tern" is used as evidence to support or refute a classification decision, which seems to suggest that some task-specific knowledge is learned and employed in the language representations.

5.3 Ablation Study: Task-specificness of Synthesized Models

To account for the improved accuracy of models synthesized with whole-model parameter adaptations, we performed additional analysis to understand the features models are utilizing when making predictions. Two variants of N^3 meta-models sharing the same set of hyper-parameters are trained on CUB-50 with the standard split. One is allowed to adapt parameters of every layer whilst the other is ablated only to generate the classification layer. Saliency map (as described in (Simonyan et al., 2014)) visualizing the importance of each pixel w.r.t. predictions is plotted as a heatmap in Figure. 4. Clearly, fully adapted models show a greater concentration on task-specific regions.

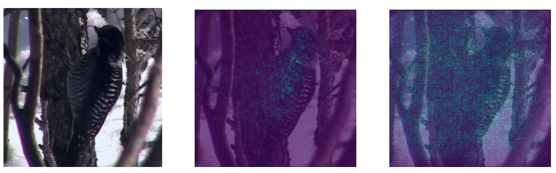

Figure 4: Saliency map showing magnitude of gradients of loss w.r.t. every pixel in the input image. Left: original image. Middle: saliency map from a model with every layer adapted by N^3. Right: saliency map from a model where N^3 is restricted to act on only its fc layer. Purple indicates small values while blue/green indicates large values.

5.4 Ablation Study: Importance of pre-trained Model

To adapt generic pre-trained model parameters to task-specific ones, N^3 mixes the pre-trained model parameters with a set of generated model parameter adaptations, using a trainable mixing ratio as illustrated in Equation 6. It is natural to question whether the pre-trained parameters are necessary. In other words, can N^3 generate parameters for a task-specific model from scratch and achieve reasonable classification accuracy? To study the importance of pre-trained model parameters, we ex-

γ	initial μ	Acc@1
0.999	0.001	16.2 %
0.99	**0.01**	**18.5** %
0.9	0.1	16.2 %
0.5	0.5	6.2 %
0.1	0.9	2.1 %
0.01	0.99	2.3 %

Table 3: Comparison of different γ and initial μ values. The pre-trained model is ignored when $\gamma = 0$.

Embedding Method	Acc @ 1
ELMo (paragraph)	4.1 %
ELMo (sentence)	5.0 %
GloVe	8.9 %
BERT	**17.6 %**

Table 4: Comparison of different embedding methods for N^3 in zero-shot classification task.

periment with a modified version of Equation 6:

$$\Theta' = \gamma \cdot \Theta + \mu \cdot \Delta\Theta$$

Where γ is a fixed scalar weight given to the pre-trained model, μ is still the trainable scaling factor. The choice of γ and the initial value of μ affects the extent to which the pre-trained model contribute to the parameter-adapted one.

In our main experiments, we have fixed γ to be 1 and initialized μ to be 1^{-3}, such choice of value initialization proves to work well, yet it remains unclear to what extent the superior performance of N^3 depends on these two hyperparameters. To answer this question, we decide to vary γ and μ; In order to keep the magnitude of parameters relatively stable, we always set the initial value of μ to be $1 - \gamma$ across different settings here. We trained N^3 to produce a 50-way classification model with varying γ, and the resultant zero-shot classification performance are shown in Table 3. Such an ablation experiment demonstrates that setting a small initial μ is crucial for performance, yet it is not the smaller the better, re-affirming the crucial role of parameter adaptation in achieving good performance.

5.5 Ablation Study: Impact of Different Word Embeddings

To understand the importance of pre-trained BERT module, and whether N^3 can be generalized to be used with pre-trained word embedding modules other than BERT, we compared the classification accuracy of models adapted by variants of N^3 using different word embeddings. Concretely, we experimented with BERT (Devlin et al., 2018), ELMo (Peters et al., 2018) and GloVe (Pennington et al., 2014) embeddings. Results are shown in Table 4. While BERT prevails as expected, ELMo seems to under-perform GloVe. We hypothesize that ELMo might have been impacted by unexpectedly long sequence lengths (here each description is a paragraph up to 512 tokens), since LSTM-based models are known to be worse at capturing long-range dependencies. To further investigate, we trained a variant of N^3, where we limit the context of ELMo to each sentence instead of the entire paragraph. As expected, the performance increases noticeably, but the resulting performance is still not on par with other methods. In contrast, GloVe embeddings worked better since such static embeddings are not affected by the long context windows.

6 Conclusion

In this paper, we have demonstrated that small amount of unstructured natural language descriptions for object classes can provide enough information to fine-tune an entire pretrained neural network to perform classification task on class labels unseen during training. We have achieved state-of-the-art performance for natural language guided zero-shot image classification tasks on 4 public datasets with practical metadata requirements. In addition, we presented in-depth analysis and extensive ablation studies on various aspects of the model functioning mechanism and architecture design, showing the necessity of our design contribution in achieving good results.

Acknowledgment

This material is based upon work partially supported by the National Science Foundation (Awards #1750439 #1722822) and National Institutes of Health. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of National Science Foundation or National Institutes of Health, and no official endorsement should be inferred.

References

2019. albanie/convnet-burden :memory consumption and flop count estimates for convnets. `https://github.com/albanie/convnet-burden`.

2019. torchvision.models. `https://pytorch.org/docs/stable/torchvision/models.html`.

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2016. Label-embedding for image classification. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1425–1438.

Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. 2015. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936.

Maruan Al-Shedivat, Avinava Dubey, and Eric P. Xing. 2017. Contextual explanation networks. *CoRR*, abs/1705.10301.

Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. 2016. Synthesized classifiers for zero-shot learning. *CoRR*, abs/1603.00550.

Sasank Chilamkurthy. Transfer learning for computer vision tutorial¶.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. 2012. Discovering localized attributes for fine-grained recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3474–3481. IEEE.

Mohamed Elhoseiny, Ahmed M. Elgammal, and Babak Saleh. 2016. Write a classifier: Predicting visual classifiers from unstructured text descriptions. *CoRR*, abs/1601.00025.

Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. 2013. Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2584–2591.

Mohamed Elhoseiny, Yizhe Zhu, Han Zhang, and Ahmed M. Elgammal. 2017. Link the head to the "beak": Zero shot learning from noisy text description at part precision. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6288–6297.

Jonathan Goldsmith. Wikipedia¶.

David Ha, Andrew M. Dai, and Quoc V. Le. 2016. Hypernetworks. *CoRR*, abs/1609.09106.

R. Lily Hu, Caiming Xiong, and Richard Socher. 2019. Correction networks: Meta-learning for zero-shot learning.

P. Kankuekul, A. Kawewong, S. Tangruamsub, and O. Hasegawa. 2012. Online incremental attribute-based zero-shot learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3657–3664.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. 2011. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO.

Elyor Kodirov, Tao Xiang, and Shaogang Gong. 2017. Semantic autoencoder for zero-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

B. Kulis, K. Saenko, and T. Darrell. 2011. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pages 1785–1792.

C. H. Lampert, H. Nickisch, and S. Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4247–4255.

M-E. Nilsback and A. Zisserman. 2006. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454.

Devi Parikh and Kristen Grauman. 2011. Relative attributes. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 503–510, Washington, DC, USA. IEEE Computer Society.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435. Association for Computational Linguistics.

Ruizhi Qiao, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. 2016. Less is more: zero-shot learning from online textual documents with noise suppression. *CoRR*, abs/1604.01146.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bernardino Romera-Paredes and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2152–2161. JMLR.org.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 6000–6010.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. 2010. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning - the good, the bad and the ugly. *CoRR*, abs/1703.04394.

Z. Zhang and V. Saligrama. 2016. Zero-shot learning via joint latent similarity embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6034–6042.

Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, and Ahmed M. Elgammal. 2017. Imagine it for me: Generative adversarial approach for zero-shot learning from noisy texts. *CoRR*, abs/1712.01381.

A Impact of Different Base Model Architectures

In this section, we provide an additional ablation experiment, we will be using the Standard Split on CUB dataset. With the superior results of N^3 obtained on the above datasets using ResNet-18 as the base architecture, it is reasonable to wonder whether the power of N^3 can extend to other base architectures and whether N^3 can work effectively on other base model architectures. To answer this question, we trained N^3 with 2 additional base model architecture: GoogleNet and SqueezeNet. We compared them in terms of zero-shot classification accuracy. Results are tabulated in Table 5.

Based on the results shown in Table. 5, we were surprised to find out that SqueezeNet performs as well as ResNet as N^3's base model and GoogleNet can even out-perform ResNet-18 as a better base model. One explaination for the superior performance of GoogleNet is that parameters of batch normalizations are not known to corrolate with task semantics and therefore can hardly be fine-tuned. GoogleNet, on the other hand, does not make use of batch normalization, thus avoiding the potential performance degradations caused by lack of fine-tuning on Batch Normalization layers.

B Additional Comparison with Prior Work

In our main experimental evaluation section, we performed extensive experiments comparing our methods with prior arts using newly established rigorous zero-shot learning evaluation guidelines (Xian et al., 2017). To adhere to the stricter guidelines, and to compare fairly with prior arts, we have to reproduce the results of prior methods ourselves; one may wonder how do we compared with each of these prior methods in their own experimental setup, and more importantly, how do we compare with their originally published performance numbers? In this section we seek to assure readers that our method continues to perform noticeably better than prior methods in their original experimental setup. To begin with, we compare the zero-shot classification performance of N^3 with prior work which also generates neural network parameters directly (Lei Ba et al., 2015).

To make our results comparable, we adopted the meta-training/testing splits from (Lei Ba et al., 2015), where in CUB-2010/CUB-2011, the 200

Model Arch	Acc @ 1	ImageNet Test Acc. (tor, 2019)	Param Size (MB) (con, 2019)
GoogleNet	**21.3 %**	**69.78 %**	**51**
ResNet18	17.6 %	69.76 %	45
SqueezeNetV1.1	17.7 %	58.19 %	5

Table 5: N^3 Zero-Shot Classification Accuracy on CUB2011 with Different Base Model Architectures

classes are randomly split into 160 ”seen” classes for training and validation (within which 120 classes are randomly selected for training, and 40 for validation) and 40 ”unseen” classes for testing, and a 40-class classification model is generated by N^3. Within the seen classes, 20% of the dataset is are excluded since (Lei Ba et al., 2015) used them for testing to report classification results on seen classes, which is irrelevant to our comparisons. Meta models are trained using the 160 seen classes only. Similarly, in Oxford Flowers, the 102 classes of flowers are split into 82 ”seen” classes and 20 ”unseen” classes randomly. We used a ResNet-18 pre-trained on ImageNet as our base model. Note that while we didn't adopt the larger VGG-19 base model as previous work, later we will show that we are still able to outperform them significantly with the lesser ResNet-18 architecture. During test time, N^3 generates neural networks based on test set class descriptions and these generated networks are used for zero-shot evaluation.

As shown in Table 6, N^3 generated ResNet18 out-performs prior arts by a significant margin both in terms of average precision and classification accuracy on both CUB and Oxford Flower dataset.

We then move on to evaluate the performance of N^3 on more rigorous and difficult zero-shot meta-splits on the CUB-2011 and NAB dataset. This split is called SCE (super category exclusive) split, which ensures that the parent categories of unseen classes are exclusive to those of the seen classes. Such split is used in many prior works (Zhu et al., 2017; Elhoseiny et al., 2017; Changpinyo et al., 2016) to evaluate their proposed method. We evaluated N^3 using such meta-split and results are shown in Table 5. As explained in the experimental evaluation section of our paper, **later prior works** (Zhu et al., 2017; Elhoseiny et al., 2017) **uses significantly more metadata (e.g., per sample parts annotation) than N^3** and yet N^3 is able to exceed their classification accuracy by using a generic ResNet-18 base model.

C Magnitude of Parameter Adaptation

In this section, we visualize in Fig. 6, the magnitude of N^3's parameter adaptation per layer (quantified by the L2 norm of each layer's parameter adaptation tensor generated by N^3) when adapting pre-trained ResNet-18.

D Hyper Parameters

We tabulated the set of hyper parameters used to produce our experimental results in Table. 7.

E Computational Resource Consumption

We used two types of machines - 4 x NVidia Pascal-100 and 4 x NVidia Volta-100 - depending on availability. It is worth noting that computation was not a bottleneck for our experiments; and a multi-GPU setup is primarily used for their larger total memory size available since transformers are known to consume a lot of memory when processing lengthy sequences. We conjecture that recent development of more efficient variants of transformer models would enable N3 to train with fewer GPUs. The training time ranged from a few hours to a single day depending on the size of the dataset and the computing power of the machine.

Dataset	Method	A.P.	Acc.@1	Acc.@5	Base Model
CUB-2010	DA (VGG) (Kulis et al., 2011)	0.037	-	-	VGG-19
	PDCNN(fc) (Lei Ba et al., 2015)	0.1	12.0%	42.8%	VGG-19
	PDCNN(conv) (Lei Ba et al., 2015)	0.043	-	-	VGG-19
	PDCNN(fc+conv) (Lei Ba et al., 2015)	0.08	-	-	VGG-19
	Ours(ResNet18)	**0.31**	**33.9%**	**77.0%**	ResNet-18
CUB-2011	PDCNN(fc) (Lei Ba et al., 2015)	0.11	-	-	VGG-19
	PDCNN(conv) (Lei Ba et al., 2015)	0.085	-	-	VGG-19
	PDCNN(fc+conv) (Lei Ba et al., 2015)	0.13	-	-	VGG-19
	Ours(ResNet18)	**0.27**	32.6%	68.6%	ResNet-18
Flowers	PDCNN(fc) (Lei Ba et al., 2015)	0.07	-	-	VGG-19
	PDCNN(conv) (Lei Ba et al., 2015)	0.054	-	-	VGG-19
	PDCNN(fc+conv) (Lei Ba et al., 2015)	0.067	-	-	VGG-19
	Ours(ResNet18)	**0.16**	10.4%	39.7%	ResNet-18

Table 6: N^3 Zero-Shot Classification Performance on CUB-2010, CUB-2011, Oxford Flower

Method	CUB Acc.@1	NAB Acc.@1
WAC-Linear (Elhoseiny et al., 2016)	5 %	-
WAC-Kernel (Elhoseiny et al., 2016)	7.7 %	6.0 %
ESZSL (Romera-Paredes and Torr, 2015)	7.4 %	6.3 %
ZSLNS (Qiao et al., 2016)	7.3 %	6.8 %
$\text{SynC}_{\text{fast}}$ (Changpinyo et al., 2016)	8.6 %	3.8
SynC_{OVO} (Changpinyo et al., 2016)	5.9 %	-
ZSLPP (Elhoseiny et al., 2017)	9.7 %	8.1 %
GAZSL (Zhu et al., 2017)	10.3 %	8.6 %
CorrectionNetwork (Hu et al., 2019)	10.0 %	9.5 %
Ours(ResNet18)	**11.9 %**	**11.1 %**

Figure 5: Zero-Shot Classification Performance on CUB-2011/NAB with SCE-split.

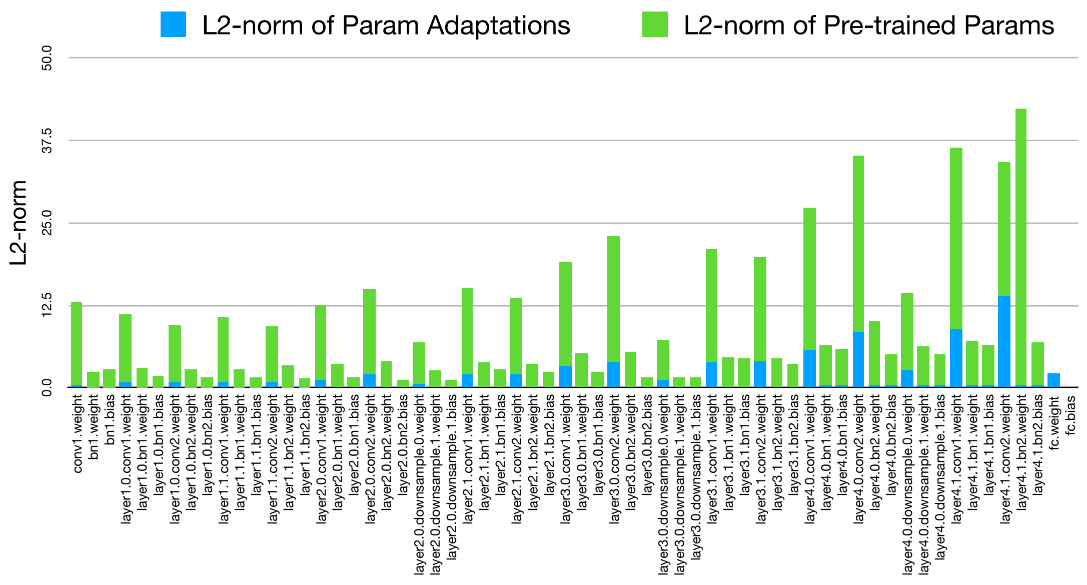

Figure 6: Magnitude of Parameter Adaptation

	(Max) Training Epoch	T2L Num Layer	L2A Num Layer	T2L LR	L2A LR
CUB-SS/PS	40	2/1	1/1	3e-5/1e-5	5e-5/1e-5
AWA-SS/PS	20	2/1	1/1	2e-6/9e-6	1e-5/2e-6
NAB-SS/PS	25	2/2	1/1	2e-5/1e-5	2e-5/2e-5
FS-SS/PS	40	2/2	1/1	2e-5/1e-6	1e-6/1e-5

Figure 7: Hyper-parameters