TOWARDS EFFECTIVE AND EFFICIENT ZERO-SHOT LEARNING BY FINE-TUNING WITH TASK DESCRIPTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

While current machine learning models have achieved great success with labeled data, we have to deal with classes that have little or no training data in many real-world applications. This leads to the study of zero-shot learning. The typical approach in zero-shot learning is to embed seen and unseen classes into a shared space using class meta-data, and construct classifiers on top of that. Yet previous methods either still require significant manual labor in obtaining useful meta-data, or utilize automatically collected meta-data while trading in performance. To achieve satisfactory performance under practical meta-data efficiency constraint, we propose **N^3** (**N**eural **N**etworks from **N**atural Language), a meta-model that maps natural language class descriptions to corresponding neural network classifiers. N^3 leverages readily available online documents combined with pretrained language representations such as BERT to obtain expressive class embeddings. In addition, N^3 generates parameter adaptations for pretrained neural networks using these class embeddings, effectively "finetuneing" the network to classify unseen classes. Our experiments show that N^3 is able to outperform previous methods across 8 different benchmark evaluations and we show through ablation studies the contribution of each model component. To offer insight into how N^3 "finetunes" the pretrained network, we also performed a range of qualitative and quantitative analysis. Our code will be released after the review period.

1 INTRODUCTION

Current deep learning models achieved great success with supervision. However, labeled data can be a luxury in many scenarios; take classification problem as an example: large number of classes, rare classes with too few samples, emerging new classes during online learning, or simply prohibitive costs of labeling can all lead to data sparsity. Hence, it is crucial to build models that can perform well even when applied to data from label categories unseen during training. This give rise to the field of zero-shot learning (Lampert et al., 2009; Larochelle et al., 2008; Palatucci et al., 2009).

Intuitively, to build models that can make predictions for classes with no training data (i.e., "unseen classes"), we still need to have some information about them. The information we are seeking usually comes from some metadata about the unseen classes and is used to embed both seen and unseen classes into a shared semantic space. An early and successful example of this approach is the use of manually engineered semantic spaces such as class attribute space (e.g color, shape, size of object) (Lampert et al., 2009) for image classification. While the engineered spaces are very effective in modelling relationships between seen and unseen classes, the metadata here, i.e. class attributes, require expert annotation. In seeking to automate the process, other works have proposed to construct semantic spaces from relevant linguistic resources. Early work in this regard constructed lexical relationship space (e.g relationship of classes as appeared in WordNet) (Akata et al., 2016) as well as text key-word space (e.g TF-IDF vectors of descriptive documents) (Lei Ba et al., 2015). More recent work explored learned representations as well. Some directly employ word embeddings of class names (Xian et al., 2016; Akata et al., 2015) while others train embeddings using descriptive documents (Reed et al., 2016; Elhoseiny et al., 2017; Zhu et al., 2017). After the embeddings in the semantic space are obtained, they are typically used to generate the last linear classifier layer on top of a pretrained feature extraction network (e.g ImageNet-pretrained ResNet for image classification).

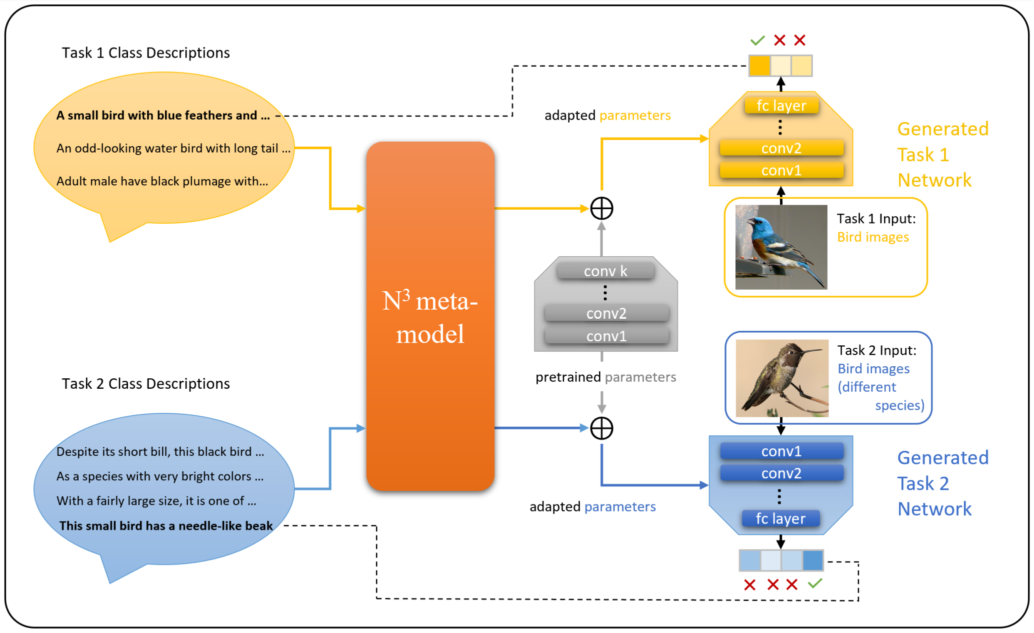

Task 1 Class Descriptions
A small bird with blue feathers and ...
An odd-looking water bird with long tail ...
Adult male have black plumage with...
N3 meta-model
adapted parameters
fc layer
conv2
conv1
Generated Task 1 Network
Task 1 Input: Bird images
conv k
conv2
conv1
pretrained parameters
Task 2 Class Descriptions
Despite its short bill, this black bird ...
As a species with very bright colors ...
With a fairly large size, it is one of ...
This small bird has a needle-like beak
Task 2 Input: Bird images (different species)
adapted parameters
conv1
conv2
fc layer
Generated Task 2 Network

Figure 1: Overview of N^3: class descriptions for a new classification task are fed to the N^3 meta-model and it generates parameters for task-specific model that classifies the designated objects. Dashed lines indicates the correspondence between dimension in the output logits and the input class descriptions.

This would allow the pretrained model to make predictions for unseen classes.

However, for previous models there seems to be an inevitable trade-off between good performance and metadata efficiency. Some models achieve satisfying metadata efficiency by only utilizing readily available online documents to construct semantic spaces (Lei Ba et al., 2015; Xian et al., 2016; Akata et al., 2015), yet when compared with concurrent state-of-the-art models (Reed et al., 2016) they fall short in performance. On the other hand, the recent models that achieve superior performance either require collecting additional corpora with hundreds of sentences per class (Reed et al., 2016) or extra fine-grained annotation for all training samples (Elhoseiny et al., 2017; Zhu et al., 2017), leading to arguably impractical metadata efficiency.

In order to achieve both superior performance and metadata efficiency, we propose a meta-model, **N^3** - **N**eural **N**etworks from **N**atural Language, mapping a set of class object descriptions to neural network parameters to create zero-shot classifiers. N^3 has two modules. The first module is a BERT-based (Devlin et al., 2018) **Semantic Encoding Module** (SEM). With the help of such large-scale pretrained representations, SEM can leverage class object descriptions that are readily available online to bridge the gap between seen and unseen classes. Compared to previous approaches that train their own embeddings with metadata, the pretrained language representations transfer knowledge learned from the large-scale pretraining corpora to our application domain. This allows us to exploit the nuances in class descriptions. Secondly, we push beyond generating only the final classifiers and propose a novel **Parameter Adaptation Module** (PAM), generating parameter adaptations to "fine-tune" **all layers** in the pretrained network based on the class embeddings given by SEM. In short, N^3 converts natural language descriptions into readily functioning neural network classifiers.

We experimented with 4 popular zero-shot learning benchmark datasets, each in 2 different settings. These datasets are Caltech-UCSD Birds (CUB), Animal with Attributes (AWA), North American Birds (NAB) and a filtered version of Oxford Flowers we refer to as Flowers-Species (FS) and we evaluated our models using settings adopted from a widely-acknowledged guideline (Xian et al., 2017). N^3 outperforms previous methods significantly across all evaluations while maintaining practical metadata requirements. We also perform a series of ablation studies to disentangle the gains from different modules, and demonstrated that each one of them offer solid contributions toward the final performance. Eventually, to gain additional insight as to how parameter adaptations affect the pretrained model, we visualized activations, model parameters and model predictions. These analysis quantitatively and qualitatively demonstrates that the parameter adaptations indeed change the behavior of pretrained models significantly.

2 Related work

Our work can be placed in the context of zero-shot learning and dynamic parameter generation for neural networks. We will discuss our work in both contexts.

2.1 Zero-shot Learning

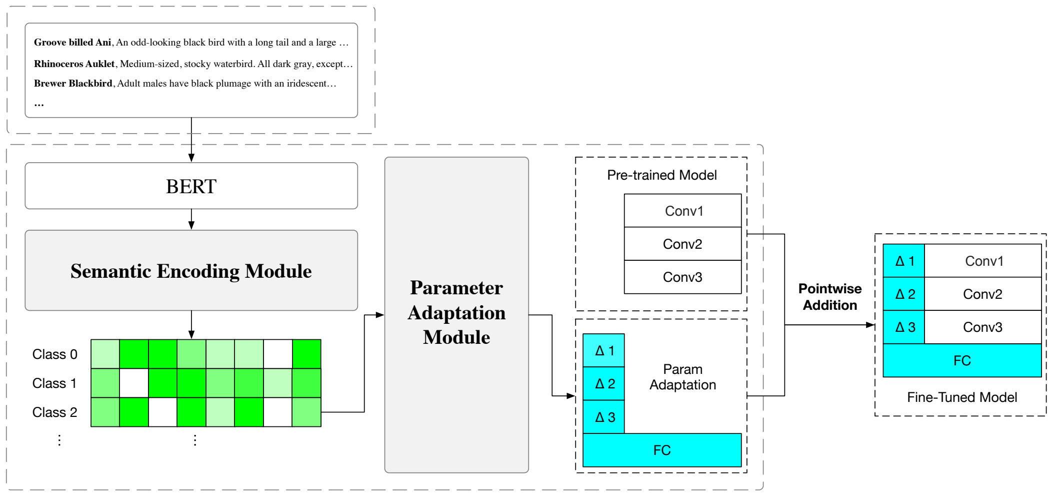
Zero-shot learning studies how we can generalize our models to perform well for tasks without any labeled training data at all. Achieving performance above chance-level for classification in such scenarios requires modelling the relationships between the classes seen during training and the unseen classes 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 & Grauman (2011); Zhang & Saligrama (2016); Akata et al. (2016). Yet this 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 text data for class object 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 reformulated 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 are a handful of work that tries to modify the parameters in-place during test time, they either rely on very simple representations of text (Lei Ba et al., 2015) or require significant amount of text 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 generate parameters for entire neural networks to adapt to the new tasks using short descriptions of the class objects. This leads to better metadata efficiency as well as higher flexibility in the generated task model.

2.2 Dynamic Parameter Generation

As mentioned before, N^3 dynamically generates classification models for designated classes. Dynamic parameter generation has been explored in 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 2.1, 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, in parameter generation, the output of the parameter generation model are large matrices that are excessively high dimensional. To deal with this, 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 generation of weight matrices as sequence of vectors. This allows parameter generation for entire neural networks with little memory bottleneck.

3 Natural Language Guided Parameter Adaptation

Our proposed N^3 architecture consists of two main modules: a Semantic Encoding Module (SEM) followed by a Parameter Adaptation Module (PAM). As is illustrated in Figure 2, the SEM first encodes class descriptions into description embeddings, then the PAM utilizes these encoded embeddings to generate parameter adaptations for a pre-trained base network. Each module applies self-attention over its input, thus leading to a two-level hierarchical attention structure. On description level, the SEM applies self-attention over tokens in each description, which allows the model to attend to key words that are crucial to the task. On class level, the PAM applies attention on all class description embeddings when generating parameter adaptations for each layer. This allows the model to have both the capability of attending to particular words in descriptions that are crucial to the task and the capability to compare and contrast different class embeddings during parameter generation. We will discuss both modules in the upcoming subsections.



Groove billed Ani, An odd-looking black bird with a long tail and a large …
Rhinoceros Auklet, Medium-sized, stocky waterbird. All dark gray, except…
Brewer Blackbird, Adult males have black plumage with an iridescent…
…
BERT
Semantic Encoding Module
Class 0
Class 1
Class 2
Parameter Adaptation Module
Pre-trained Model
Conv1
Conv2
Conv3
Δ 1
Δ 2
Δ 3
Param Adaptation
FC
Pointwise Addition
Fine-Tuned Model

Figure 2: N^3 Model Architecture. N^3 utilizes a two-level attention mechanism to map task meta-data to adapted neural network parameters: the **Semantic Encoding Module** applies a first level of attention mechanism over contextualized word embeddings of class description to produce description embeddings for each class; the **Parameter Adaptation Module** applies a second level of attention mechanism over all of the encoded class description embeddings to generate a parameter delta, which is then combined with the corresponding pre-trained model parameters to produce a task-specific neural network with its parameters adapted toward the new task.

3.1 Semantic Encoding Module

The Semantic Encoding Module is responsible for encoding class descriptions into corresponding embeddings, resulting in a set of class embeddings $\{c_i\}_{i=1}^{k}$ for all k classes. This is achieved by first applying pre-trained BERT model (Devlin et al., 2018) to class descriptions to obtain contextualized word embeddings, and then transforming each sequence of embedded descriptions into a class embedding using a Transformer (Vaswani et al., 2017). The SEM is the first hierarchy of attention in N^3: it attends to words within each description with its Transformer architecture, thus allowing the meta-model to attend to key words in the description that are more important for the new task.

3.2 Parameter Adaptation Module

The Parameter Adaptation Module use two steps to generate a task-specific neural network from class embeddings and pretrained neural network (i.e. the base model). For each layer in the pre-trained network with parameter W_{pre}, PAM generates a parameter delta ΔW with the same shape as W_{pre} that gets added to W_{pre}. To transform the set of class embeddings $\{c_i\}_{i=1}^{k}$ into a parameter delta matrix ΔW, PAM takes an encoder-decoder approach [1], viewing $\{c_i\}_{i=1}^{k}$ as a sequence and the generation of ΔW as a sequence of m columns of the matrix $\{\Delta w_d\}_{d=1}^{m}$. Similar to SEM, we also

[1]We used a non-conventional decoder setup to suit our need, see detail in supplement ”Decoder Setup”.

use Transformer models as our encoder-decoder architecture. Concretely, PAM can be expressed as:

$$\mathbf{h}_{1:k} = \texttt{Encoder}(c_{1:k}) \tag{1}$$

$$\Delta w_d = \texttt{Decoder}(\Delta w_{<d}, \mathbf{h}_{1:k}), d = 1, ..., m \tag{2}$$

$$\Delta W = \texttt{Concatenate}([\Delta w_1; \Delta w_2; ...; \Delta w_m]) \tag{3}$$

$$W = W_{pre} + \mu \cdot \Delta W \tag{4}$$

where μ is a trainable scaling factor used to control the scale of parameter delta applied to the pre-trained parameters. Through the use of Transformers models, PAM forms the second level of attention in N^3, namely the attention over classes. When generating parameter deltas, PAM always examines all classes concurrently, allowing the model to consider the interactions between them, whereas previous work almost always encode and make use of class representations independentlyLei Ba et al. (2015); Akata et al. (2015). Intuitively, the class-level attention allows N^3 to encode *distinct* features of the classes that help set them apart, instead of simply encoding each class without seeing others.

3.3 Training Methodology

In this section, we provide details about the procedures used for training N^3. Recall that, training N^3 requires a base model $\mathcal{F}$ and a set of pre-trained parameters of $\mathcal{F}$, denoted as W_{pre}. N^3 generates a set of parameter deltas ΔW to be combined with pre-trained parameters using Equation 4, such that for input images X, $\mathcal{F}(X; W_{pre}, \Delta W)$ predicts the correct labels of X.

We formulate the training of N^3 as an optimization problem. Optimal parameters θ for N^3 meta-learning model $\Phi(\cdot; \theta)$ should map a collection of natural language class descriptions $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_k\}$ to a set of parameter deltas $\Delta W = \Phi(\mathcal{D}; \theta)$, such that the cross entropy loss between ground truth label Y and model prediction $\mathcal{F}(X; W_{pre}, \Phi(\mathcal{D}; \theta))$ is minimized for all image-label pairs (X, Y) in the training set.

Therefore, to train N^3 on a image dataset $\mathcal{I}$ with labels $\mathcal{L}$, label descriptions $\mathcal{D}$ and a base model $\mathcal{F}$ to produce a k-class 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 from $\mathcal{I}_{\mathcal{L}_k}$. For each mini-batch $\mathcal{B}$, an independent set of parameter deltas ΔW is generated by evaluating $\Phi(\cdot; \theta)$ at $\mathcal{D}_{\mathcal{L}_k}$. Batch loss is then calculated as $\frac{1}{|\mathcal{B}|} \sum_i \ell(Y_i, \mathcal{F}(X_i, W_{pre}, \Delta W))$ where $\ell(\cdot, \cdot)$ refers to cross-entropy loss. Since the meta-model $\Phi(\cdot; \theta)$ and the base model $\mathcal{F}$ are fully differentiable, gradients can be propagated back to meta-models to optimize meta model parameter θ.

4 Experimental Evaluation and Discussion

In this section, we will introduce our experimental evaluation setup: the datasets we experimented with, the prior arts we compared with, and the evaluation protocol we developed. We will then move on to provide qualitative and quantitative analysis into N^3's working mechanism. Finally, several ablation studies will be provided to further characterize the impacts of several of our design choices on the model performance.

4.1 Datasets

In this section, we will introduce two types of datasets: the first type is regular datasets used to train the task model that makes the predictions; the second type is meta-datasets, consisting of natural language descriptions for classes in their corresponding regular datasets. The meta-datasets are used as meta-data to train the N^3 meta model.

- **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, 11,788 images are present in this dataset.
- **Animal with Attributes** (AWA) (Lampert et al., 2014; Xian et al., 2017) is another popular 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').
- **Flowers-Species** (FS) is a dataset we built based on Oxford Flowers (Nilsback & Zisserman, 2006). 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 every regular dataset, we build a meta-dataset by mapping each class label to a short excerpt of visual description of objects in this class. We manually find these visual descriptions from websites like Wikipedia. When such meta-dataset is fed into our BERT-based word embedding modules, we truncate these text excerpts to be within 512 tokens each; such length constraint is imposed by BERT embedding module due to its high GPU memory requirements.

4.2 Evaluation Protocol

Recent study(Xian et al., 2017) 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. Although both our tasks and datasets differ, we nevertheless developed our own set of evaluation protocols following the guiding principles of the *Rigorous Protocol* (Xian et al., 2017). Below our evaluation protocol is introduced with reference to *Rigorous Protocol*: (a) unlike *Rigorous Protocol* which uses ResNet-101 model, we use ResNet-18 as our base model; such choice helps reduce the output dimensionality of N^3 by reducing the number of parameters N^3 fine-tunes; (b) similar to *Rigorous Protocol*, we used two meta-splits **Standard Splits** and **Proposed Splits** to evaluate all methods in our experiments; while 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; (c) due to the class imbalance of several datasets, *Rigorous Protocol* proposes to use per-class averaged accuracy for more meaningful evaluation. Specifically, to evaluate meta-model on a meta-split containing the set of classes C, we calculate the per-class averaged accuracy as illustrated in Equation 5; (d) our method is unique in that permutations of the classes belonging to the same meta-split are treated as distinct tasks and therefore, to account for potential variations of the models being generated, we test our models on 10 different permutations of test set classes and report the medium of the aforementioned evaluation metrics. To summarize, we have tabulated the number of class labels used for training, validation and testing in Table. 1.

$$Acc_C = \frac{1}{|C|} \sum_{c \in C} \frac{\text{number of correctly predicted samples in c}}{\text{number of samples in c}} \tag{5}$$

4.3 Comparison with Prior work

Our work is most closely associated with prior method (Lei Ba et al., 2015) which also uses only natural language descriptions of class labels to generate deep neural networks capable of distinguishing between objects described. We will refer to it as PDCNN (**P**redicting **D**eep **C**onvolutional **N**eural **N**etworks). PDCNN (Lei Ba et al., 2015) is distinct from our work in that it dynamically generates fully connected layers and/or additional convolutional layers to be **appended** to a pretrained

Dataset	Total	Standard Split/Proposed Split		
		Training	Validation	Testing
$CUB(Wahet\,al.,\ 2011)$	200	100	50	50
$AWA2(Zhuet\,al.,\ 2017;\ Elhoseinyet\,al.,\ 2017)$	50	30	10	10
$NAB(Zhuet\,al.,\ 2017)$	404	324	40	40
$FS(Nilsback\&Zisserman,\ 2006)$	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.

deep neural network (VGG-19) whilst ours generates a set of parameter adaptations to be **combined directly** with all existing layers within deep neural network models that effectively fine-tunes the base model. To make our works comparable, we replaced the TF-IDF feature extractor in PDCNN with a BERT-based document embedding (specifically, a BERT word embedding followed by max-pooling) and changed the base model from VGG-19 to ResNet-18. Two best performing variants of PDCNN (Lei Ba et al., 2015) are reproduced.

Due to the sparsity of prior work that leverages natural language descriptions for zero-shot classifications in a metadata-efficient way, we also tried to adapt less metadata-efficient methods to function with stricter metadata-efficiency requirements. Specifically, ZSLPP (Elhoseiny et al., 2017), GAZSL (Zhu et al., 2017) and CorrectionNetwork (Hu et al., 2019) all utilizes natural language 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 prior 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 the help of 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 natural language processing module from TF-IDF to BERT-based document embedding 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 with the same exact algorithms (random search) and for the same number of runs (10).

Method	CUB-50		AWA2-10		NAB-40		FS-10	
	SS	PS	SS	PS	SS	PS	SS	PS
$PDCNN_{FC}$	6.1	6.1	12.8	18.4	6.7	7.4	13.1	11.9
$PDCNN_{FC+CONV}$	7.5	6.5	22.6	17.2	8.9	5.6	7.7	13.6
$MEGAZSL$	2.9	1.8	14.0	10.4	2.8	3.7	10.3	12.3
$N^3(Ours)$	**17.6**	**9.5**	**34.0**	**37.5**	**14.1**	**20.7**	**16.4**	**17.6**

Table 2: Zero-Shot Classification Accuracy On Various Datasets/Meta-Splits Combinations; our method is compared with 2 versions of PDCNN (Lei Ba et al., 2015) and MEGAZSL (Zhu et al., 2017); number of unseen class labels in the test set are recorded next to the name of each dataset.

All experimental results characterized by per-class averaged accuracy are tabulated in Table 2. For readers wondering how we compare with these methods in their original set up, we have attached additional comparisons in Appendix. A.2 with prior methods in their original setup; although such evaluation may inevitably be less rigorous. 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

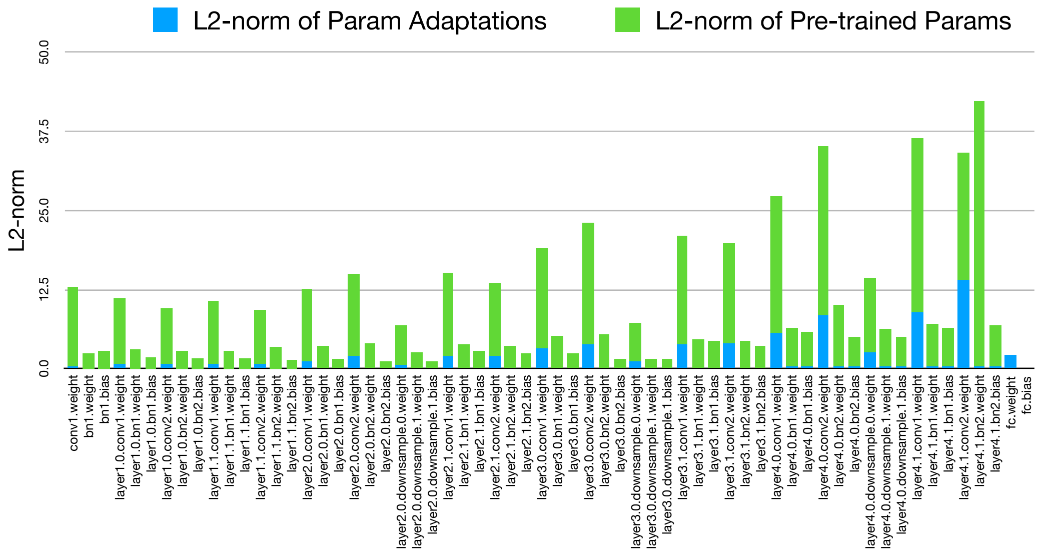


Figure 3: Composition of Fine-Tuned ResNet18 Parameters Produced by N^3, Measured by L2 Norm (the image enters through the leftmost layer and the prediction exits from the rightmost layer)

we replaced its image feature embedding module, which is trained with the help of per-sample parts annotations, the performance dropped significantly (from 11.3% to 3.3%), confirming our conjecture that such methods cannot be easily adapted to work under strict metadata-efficiency constraint.

4.4 Quantitative and Qualitative Analysis

We will provide detailed analysis into the inner workings of N^3. To begin with, it is instrumental to understand **the extent of parameter adaptations N^3 has applied to the base model** to quantify the role of N^3 in the fine-tuning process. To this end, we visualized the composition of parameters of each layer of the fine-tuned model trained on CUB Standard Split. Recall from Equation 4 that each layer's parameter tensor is the sum of two tensors: the pre-trained base model parameters and the parameter adaptations generated by N^3 meta-model; we quantified the magnitude contributed by each tensor by calculating their respective L2-norm. Such quantification is performed on every layer of the fine-tuned model and the result is visualized in Figure. 3. It demonstrates that the extent of parameter adaptations is significant in many layers and increases towards the later layers of the model; such phenomenon is likely due to the fact that deep CNNs capture generic features in earlier layers and task-specific ones in later layers. Therefore, fine-tuning base models to tackle new tasks naturally demands higher magnitude of parameter adaptations for deeper layers.

Seeking to understand **how parameters from different types of layers are adapted differently**, we also plotted the per-layer average magnitude (measured in L2 norm) of parameter adaptations grouped by layer types in Figure. 5(a). It can be observed that parameter adaptations are predominantly applied to convolutional and fully-connected layers, while batch normalization layer parameters remain largely unchanged; this is likely due to the fact that former types of layers are more likely to be correlated with the semantics of task descriptions whilst batch normalization layer parameters are unlikely to be induced by task semantics.

To understand **how parameter adaptations affect task-specific feature extraction process of base models**, we visualized the distribution of L2-norms of penultimate layer activation vectors obtained on the test split of CUB dataset, before and after parameter adaptation is applied in Figure. 5(b). It can be observed that the magnitude of base model's penultimate layer activation vector tends to either become more prominent or pushed further towards zero. It is likely that parameter adaptations have made feature extraction process more task-specific, so that task-related features are extracted with higher magnitude while less pertinent features are more actively discarded.

Furthermore, even though N^3 outperformed prior methods with significant margin; we are nevertheless interested in **understanding why N^3 performs less desirably on specific dataset (e.g., Flowers-Species)**; to this end, we plotted the normalized confusion matrix computed on the test set in Figure. 5 (c). Interestingly there appear to be vertical clusters of points of confusion, which implies that many samples are falsely classified to be a a select few categories. We therefore examined the textual descriptions of such labels, and that of the worst offender (windflower) is shown below in Figure 4. The textual descriptions indeed does not capture the distinguishing features of windflowers and therefore confused N^3 to fine-tune base models such that many samples are mis-classified as windflowers. This also lead us to conclude that the quality of metadata plays a critical role dictating the performance of the fine-tuned model; and a single poorly constructed metadata entry can have disproportionate effect on fine-tuned model accuracy; nonetheless, such problem is not unique to our approach but rather a consequence of an efficiency-robustness trade-off inherent in our problem setup. Moreover, our natural language based metadata makes such problem exceedingly easy to diagnose. On the other hand, we also found that the metadata procured from online sources vary in quality and tend to focus on facets that are most appealing to human readers; such aspects of visual descriptions tend to constitute only necessary, but not sufficient conditions for classification decisions. To conclude, we believe that N^3's performance can be increased further with higher quality task descriptions.

```
Windflowers grow underground
from tubers or rhizomes to form
small colonies.  Depending on
the variety, the flower stalks
grow from six inches tall to
nearly six feet.  Flower color
varies widely, but the blossoms
are generally two to three
inches in diameter with thin,
delicate petals.  The deeply
divided foliage is dark green
and clustered near the base of
the plants.
```

Figure 4: Textual Descriptions for "Windflowers"

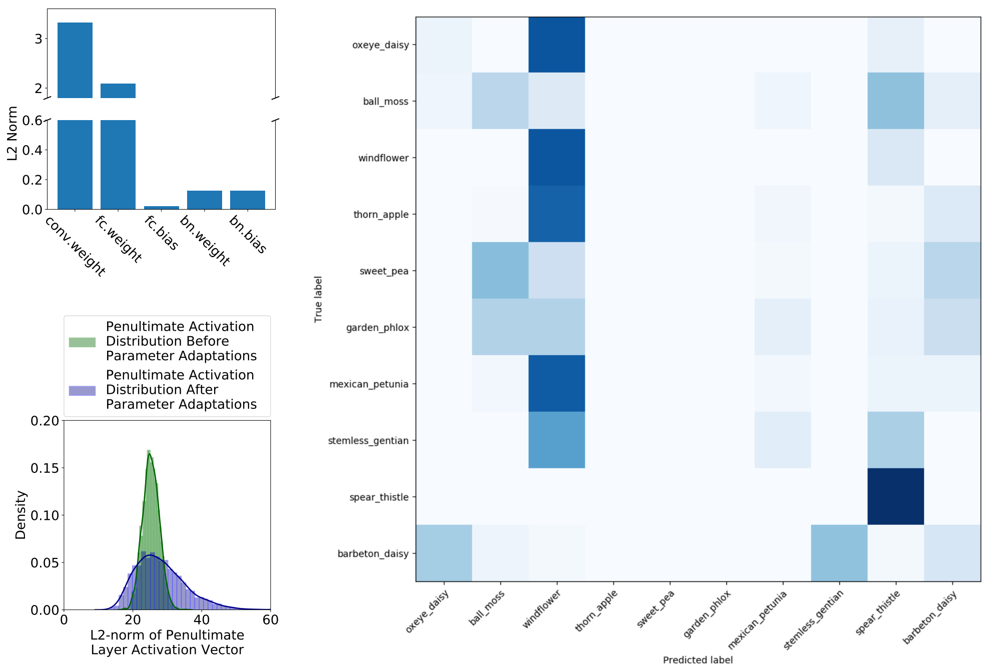


Figure 5: Upper Left: (a) Per-Layer Average L2 Norm of Parameter Adaptations Grouped By Layer Types. Lower Left: (b) Penultimate Layer Activation Vector Magnitude Distribution Before and After N^3 Fine-Tuning. Right: (c) Normalized Confusion Matrix for Flowers-Species.

4.5 Importance of Pretrained Model

We can interpret the parameter adaptation process of N^3 as a process of mixing the pre-trained base model parameters with a set of generated model parameter deltas, using a trainable mixing ratio.

For the purpose of studying the importance of pre-trained model parameters, we experiment with a slightly modified version of Equation 4:

$$W = \gamma \cdot W_{pre} + \mu \cdot \Delta W$$

Where γ is a fixed scalar weight given to the pre-trained model, μ is still the trainable scaling factor for the Parameter Adaptation Module. The choice of γ and the initial value of μ affects how much influence the pre-trained model have. 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 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-class classification model with varying γ and the resultant zero-shot classification performance are shown in Table 3. Such 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 helping the base model achieve better performance.

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 completely ignored in the case where $\gamma = 0$.

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.

4.6 Impact of Different Word Embeddings

To understand how much of N^3's efficacy can be attributed to the 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 performances 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 below GloVe. We hypothesize that ELMo might have been impacted by unexpectedly long sequences (here each description is a short paragraph that can contain up to 512 tokens), especially that LSTM-based models like ELMo are known to be worse at capturing long-range dependencies than self-attention based models such as BERT. To further investigate, we trained a variant with ELMo to perform only sentence level embedding and aggregate these sentence embeddings together to form the paragraph-level embeddings. As expected, the performance increases noticeably, but the resulting performance is still not on par with other methods. As for GloVe, since it is used together with Semantic Encoding Module inside N^3, they act together like a shallower version of BERT, and is also capable of achieving decent performance.

5 Conclusion

In this paper, we have demonstrated that small amount of unstructured natural language descriptions for class objects 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.

REFERENCES

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

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

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

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

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

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

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

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

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

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

Mohamed Elhoseiny, Yizhe Zhu, Han Zhang, and Ahmed M. Elgammal. 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*, pp. 6288–6297, 2017. doi: 10.1109/CVPR.2017.666. URL https://doi.org/10.1109/CVPR.2017.666.

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

R. Lily Hu, Caiming Xiong, and Richard Socher. Correction networks: Meta-learning for zero-shot learning, 2019. URL https://openreview.net/forum?id=r1xurn0cKQ.

P. Kankuekul, A. Kawewong, S. Tangruamsub, and O. Hasegawa. Online incremental attribute-based zero-shot learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3657–3664, June 2012. doi: 10.1109/CVPR.2012.6248112.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. 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, June 2011.

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

B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR 2011*, pp. 1785–1792, June 2011. doi: 10.1109/CVPR.2011.5995702.

C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3): 453–465, March 2014. doi: 10.1109/TPAMI.2013.140.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 951–958. IEEE, 2009.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, pp. 3, 2008.

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

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

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pp. 1410–1418, 2009.

Devi Parikh and Kristen Grauman. Relative attributes. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pp. 503–510, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126281. URL `http://dx.doi.org/10.1109/ICCV.2011.6126281`.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

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

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 425–435. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/D18-1039`.

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

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

Bernardino Romera-Paredes and Philip H. S. Torr. 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, pp. 2152–2161. JMLR.org, 2015. URL `http://dl.acm.org/citation.cfm?id=3045118.3045347`.

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. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

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

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

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

Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 69–77, 2016.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning - the good, the bad and the ugly. *CoRR*, abs/1703.04394, 2017. URL `http://arxiv.org/abs/1703.04394`.

Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6034–6042, June 2016. doi: 10.1109/CVPR.2016.649.

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

A Appendix

A.1 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.

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

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 we have noted in Section. 4.4, parameters or batch normalizations are hardly fine-tuned and GoogleNet in fact does not make use of Batch Normalization, thus avoiding the potential performance degradations caused by lack of fine-tuning on Batch Normalization layers.

A.2 Comparison with Prior Work in Their Own Setup

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). But 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 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.

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

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.

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 & 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 6: Zero-Shot Classification Performance on CUB-2011/NAB with SCE-split.

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 6. 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.

A.3 Decoder Setup

In N^3, we used a slightly different decoder setup and in this supplemental material, we will explain in detail how we deviate from the standard decoder setup and provide motivations for such design choice.

Concretely, the N^3's decoder is not auto-regressive. Usually, transformer encodes a sequence of inputs $X = (x_1, \cdots, x_n)$ to a latent representation of the input sequence $Z = (z_1, \cdots, z_n)$, where n is the input sequence length. The latent representation is then decoded to $Y = (y_1, \cdots, y_m)$ one symbol at a time, where m is the output sequence length. This is because usually, when transformer decoder generates y_k, attention mechanism is applied to $y_1, \cdots, y_{k-1}$ as well. Since in tasks like language translation, output symbols correlate closely with one another to form a coherent sentence, this auto-regressive feature of transformer decoder is very natural to be employed.

In the context of N^3, however, we believe simultaneous decoding without auto-regressive correlation to be more suitable. Consider an example for N^3 to generate a convolutional layer parameter of shape $K \times C \times kH \times kW$, where K is the output channel size, C is the input channel size and kH, kW are filter height and width, respectively. We generate such parameter as a sequence of flattened filters, with sequence length being K and hidden dimension size (which is, in this case, also the flattened filter size) being $C \times kH \times kW$; we do not expect the k-th filter to correlate too much with the filters generated before it, since it is fine for them to be extracting unrelated semantic features from input images. Therefore we do not believe the decoder needs to be auto-regressive when generating parameters in N^3; we thus generate all K filters at once. Moreover, generating K filters at once, instead of one at a time can result in some training and inference speedup. Therefore in N^3, we do not use decoder in an auto-regressive way.