Semantic Supervision: Enabling Generalization over Output Spaces

Ameet Deshpande♦ **Austin W. Hanjie**♦ **Karthik Narasimhan**

Department of Computer Science
Princeton University
{asd, hjwang, karthikn}@cs.princeton.edu

Abstract

In this paper, we propose *semantic supervision* (SEMSUP) – a unified paradigm for training classifiers that generalize over output spaces. In contrast to standard classification, which treats classes as discrete symbols, SEMSUP represents them as dense vector features obtained from descriptions of classes (e.g., "The cat is a small carnivorous mammal"). This allows the output space to be unbounded (in the space of descriptions) and enables models to generalize both over unseen inputs and unseen outputs. Specifically, SEMSUP enables four types of generalization, to – (1) unseen class descriptions, (2) unseen classes, (3) unseen super-classes, and (4) unseen tasks. Through experiments on four classification datasets, two input modalities (text and images), and two output description modalities (text and JSON), we show that our SEMSUP models significantly outperform baselines. For instance, our model outperforms baselines by 40% and 15% precision points on unseen descriptions and classes, respectively, on a news categorization dataset (RCV1). SEMSUP can serve as a pathway for scaling neural models to large unbounded output spaces and enabling better generalization and model reuse for unseen tasks and domains.

1 Introduction

Most approaches to supervised classification have traditionally considered different output classes as abstract symbols devoid of meaning (e.g., 0, 1, 2). This predefines a rigid output space that inhibits models from generalizing to unseen classes (e.g., "moth"), even if it is similar to a class seen during training (e.g., "butterfly"). Some prior works have aimed to tackle this problem by predicting classes based on semantic class attributes (Palatucci et al., 2009), word vectors of class names (Frome et al., 2013), or textual class descriptions (Lei Ba et al., 2015). These works provide solutions with a specific capability in mind, such as zero-shot generalization or domain generalization.

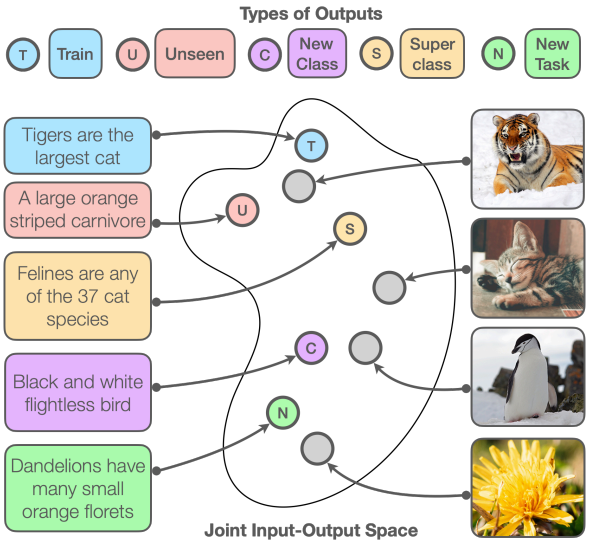

Figure 1: High-level overview of SEMSUP and different types of generalization it enables. Instances (shown as images) and outputs in the form of class descriptions are embedded into a joint input-output space. This allows models to generalize to unseen descriptions ("a large orange striped carnivore"), new classes (*penguin*), super-classes (*felines*), and new tasks (flower classification).

In this paper, we develop a general unifying paradigm for supervised classification called *semantic supervision* (SEMSUP) to leverage rich semantic information about classes to enable better generalization over the output space. SEMSUP allows models to learn better representations of output classes using multiple "descriptions" that capture their semantics and neural output encoders to represent them in vector space. SEMSUP can handle different types of descriptions, and we experiment with textual descriptions (e.g., "The cat is carnivorous mammal") and JSONs (e.g., {`size:small, legs:4`}).

Training models to predict over semantically informative choices has several advantages: (1) the number of choices can be varied during inference, (2) the choices can be described in several different ways (e.g., by different end-users), (3) new concepts can be provided as choices by describing them in known terms, and (4) the choices can span varying levels of granularity (e.g., classify between descriptions of *vehicle*, *bus* and *wheel*).

We demonstrate the general applicability of SEM-

♦Equaopl contribution – order decided over a game of snake.

SUP to any standard classification task by considering four existing benchmarks spanning text and image inputs, two different types of 'descriptions' (English text and structured JSON), and two paradigms – multi-class and multi-label classification. We show that SEMSUP models generalize over output spaces in various ways, including (1) new descriptions of seen classes, (2) unseen classes, (3) unseen high-level superclasses, and (4) unseen tasks (Figure 1). In all tasks and scenarios SEMSUP outperforms existing systems developed for zero-shot generalization to unseen classes while also remaining competitive with standard classifiers on seen classes. For instance, SEMSUP achieves absolute improvements of 40% on unseen descriptions in RCV1, 15% on unseen classes in CIFAR, and 10% on unseen superclasses in 20 Newsgroups. We recognize the importance of using multiple descriptions and pre-trained models for encoding semantic supervision (§ A), which provides a recipe for users to adopt our work.

2 Related Work

SEMSUP unifies these works by using multiple rich descriptions, multiple input (e.g., image and text) and output (e.g., text and JSON) modalities, and exhibiting numerous generalization capabilities to – unseen descriptions, classes, superclasses, and tasks. We summarize SEMSUP's capabilities and prior work in Table 1.

Zero-shot learning with auxiliary information The goal of zero-shot learning is to predict novel classes not encountered at train time by using class specifications (auxiliary information) of some form, like representative images (Larochelle et al., 2008). At their core, these works define a joint embedding space for inputs and auxiliary information that represents classes. Subsequent papers like DeViSE (Frome et al., 2013) and others Socher et al. (2013); Pappas and Henderson (2019); Dauphin et al. (2014) used shallow averaged word embeddings corresponding to class names, which have their shortcomings because they are oblivious to the word order. Other papers have employed text descriptions of classes, either crowdsourced or scraped from Wikipedia. Some of these use simple bag-of-words features to encode them (Nam et al., 2016; Lei Ba et al., 2015; Qiao et al., 2016), which has the issue of ignoring word order and a portion of semantics, while others use deeper models (Reed et al., 2016; Bujwid and Sullivan, 2021). However, all these papers focus only on a single input modality (images), a single output modality (text), or a single generalization scenario (to unseen classes), whereas SEMSUP handles a variety of these attributes.

Prompting pre-trained models SEMSUP is partly related to a recent area of research which uses natural language to prompt large pre-trained models (Liu et al., 2021). Prompting contrastive models like CLIP (Radford et al., 2021) involves providing class names at inference time (e.g., "Image of a dog") and choosing the description with the highest similarity to the instance. However, SEMSUP differs from CLIP in major ways. CLIP requires a large about of paired image-caption data, which necessitates that different images (potentially belonging to the same category) have different captions. However, SEMSUP requires very inexpensive *class* level descriptions which can be re-used for all the images belonging to that class. Further, SEMSUP can also operate on multiple types of output descriptions like NL and JSON, whereas CLIP uses only NL descriptions.

Learning with natural language descriptions The body of work around natural language explanations (Srivastava et al., 2017, 2018; Murty et al., 2020; Hancock et al., 2018; Mu et al., 2020) aims to induce classifiers with the help of explanations that describe rationales for instances belonging to specific classes. Some weak supervision studies use class-specific auxiliary information (natural language descriptions or from knowledge bases) to generate labeling functions which are used to augment the training data by annotating unsupervised data (Hancock et al., 2018; Ratner et al., 2017). But unlike our paper, both these lines of work aim to improve few-shot learning on a *bounded* set of classes, whereas we enable even zero-shot learning on an unbounded set. Our work is also related to studies that learn classifiers (Andreas et al., 2018), reinforcement learning (RL) agents (Andreas et al., 2018; Zhong et al., 2019; Narasimhan et al., 2018; Hanjie et al., 2021), and programs (Acquaviva et al., 2021; Wong et al., 2021) by "reading" natural language descriptions of the task. In contrast to our work, these studies typically evaluate on synthetic domains and only consider text descriptions.

3 Methodology

3.1 Background

Our paradigm (SEMSUP) is a modification of the standard supervised classification paradigm (SUP), which involves using *data* to learn a *model* by minimizing a *loss function*. The training data can be represented as $\mathcal{D}_{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ denote the input and the categorical output ($\mathcal{Y} = \{1, \ldots, K\}$) sampled from a hidden underlying distribution $(x_i, y_i) \sim P_{true}(\mathcal{X}, \mathcal{Y})$. We construct a *model* $\mathcal{M}_\theta$ which can predict the conditional probability of outputs given the input – $P_{pred}(y|x_i)$ and learn it by minimizing a *loss function* $\mathcal{L}\left(P_{pred}(y|x_i), y_i\right)$ like cross-entropy or max margin loss.

Without loss of generality, let us assume $\mathcal{M}_\theta$ to be a neural network with a hidden representation of dimensionality d and the number of output classes to be K. Then, we can factorize $\mathcal{M}_\theta$ to consist of (1) an input encoder $f(\cdot)$ which encodes the input as $f(x_i) \in \mathbb{R}^d$ and an (2) output matrix $\mathcal{O} \in \mathbb{R}^{K \times d}$. This allows the model to represent the conditional distribution over output classes as:

$$P_{\text{SUP}}(y|x_i) = \text{softmax}\left(\mathcal{O} \times f\left(x_i\right)\right) \quad (1)$$

Categorization	Papers	Descriptions		Multiple modalities		Capabilities: Generalization to unseen			
		Multiple	Rich	Inputs	Outputs	Descriptions	Classes	Superclasses	Tasks
Semantic supervision using multiple descriptions ***(Both text and JSON)***	**Ours (SEMSUP)**	✓	✓	✓	✓	✓	✓	✓	✓
Shallow bag-of-vectors of class names ***(Text only)***	Frome et al. (2013)	✗	✗	✗	✗	✗	✓	✓	✗
	Mittal et al. (2021); Dauphin et al. (2014) Wang et al. (2018); Socher et al. (2013)	✗	✗	✗	✗	✗	✓	✗	✗
	Zhang et al. (2018)	✗	✗	✗	✗	✗	✓	✗	✓
Deep embeddings of class descriptions ***(Text only)***	Reed et al. (2016)	✓	✓	✗	✗	✗	✓	✗	✗
	Zhang et al. (2017); Qiao et al. (2016) Bujwid and Sullivan (2021) Pappas and Henderson (2019)	✗	✓	✗	✗	✗	✓	✗	✗
Attribute annotation ***(JSON only)***	Koh et al. (2020) Akata et al. (2015); Demirel et al. (2017)	✗	✓	✗	✗	✗	✓	✗	✗

Table 1: Summary of related work. SEMSUP provides a unified framework to handle multiple and rich descriptions (referencing different attributes of the class), input modalities (image and text), and output modalities (text descriptions and JSON). Unlike prior work SEMSUP handles multiple modalities for representing the output (text and JSON) and exhibits several types of generalization (to unseen descriptions, classes, superclasses, and tasks).

3.2 Semantic Supervision

Our semantic supervision (SEMSUP) paradigm uses the same *data*, *loss functions*, and *input encoder* ($f(\cdot)$) used in SUP, but changes the output matrix ($\mathcal{O}$). Instead of representing the i^{th} class (say "cat") using a randomly initialized vector, SEMSUP encodes semantic information about the class into $\mathcal{O}[i]$ (e.g., the sentence "The cat is a small carnivorous mammal"). Hereon, we will use the term "description" to refer to any kind of semantic information (text or JSON).

SEMSUP requires access to descriptions for each class $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_K)$, where the i^{th} element contains a set of descriptions corresponding to the i^{th} class: $\mathcal{C}_i = \{c_{i1}, \ldots, c_{iL}\}$. Our SEMSUP model independently samples a description for each class (c_{ij}) and employs an output encoder $g(\cdot)$ to encode it as $g(c_{ij}) \in \mathbb{R}^{d_{out}}$, which make up rows of the output matrix $\mathcal{O}^{\text{SEMSUP}}[i] = g(c_{ij})$. Since the last dimension of $\mathcal{O}^{\text{SEMSUP}} \in \mathbb{R}^{K \times d_{out}}$ doesn't necessarily match the dimensionality of the input encoder $f(x_i) \in \mathbb{R}^d$, we learn an intermediate projection matrix $\mathcal{P} \in \mathbb{R}^{d_{out} \times d}$. The final prediction is obtained as:

$$P_{\text{SEMSUP}}(y|x_i) = \text{softmax}\left(\mathcal{O}^{\text{SEMSUP}} \times \mathcal{P} \times f\left(x_i\right)\right) \tag{2}$$

Training and testing with SEMSUP During training, we sample a description for each class uniformly at random and construct an output matrix ($\mathcal{O}^{\text{SEMSUP}} \in \mathcal{R}^{K \times d_{out}}$) such that the i^{th} row representing the i^{th} class is $\mathcal{O}^{\text{SEMSUP}}[i] = g(c_{ij}), c_{ij} \sim \mathcal{U}(\mathcal{C}_i)$. We learn the input encoder ($f(\cdot)$), the output encoder ($g(\cdot)$), and the label projection matrix ($\mathcal{P}$) together. Descriptions are sampled uniformly at random for each class and batch. During testing, we predict the class corresponding to the class description with the highest softmax probability.

4 Experimental Setup

4.1 Datasets

We evaluate SEMSUP on four diverse datasets chosen to test generalization on different scenarios (§ 3). **20 Newsgroups** (20NG) (Lang, 1995) consists of 20,000 newsgroup documents in 20 classes. We partition similar classes into 5 superclasses. **CIFAR-100** (Krizhevsky et al., 2009) consists of 60K images in 100 classes, each assigned to one of 20 superclasses (e.g. *aquatic mammals*). **Animals with Attributes 2** (AWA2) (Xian et al., 2018) is an animal classification dataset with 37K images and 50 classes. The dataset also includes 85 animal attributes (e.g. *fur*, *swims*). Each class is annotated with binary values indicating whether the attribute is present in the class. **RCV1** (Lewis et al., 2004) is a multi-label news classification dataset (multiple correct classes possible) with 103 classes. We withhold 17 parent classes in the provided hierarchy to test unseen superclass generalization. We evaluate using the label ranking average precision (LRAP) metric (Pappas and Henderson, 2019).

4.2 Collecting output supervision

We collect **textual output descriptions** for RCV1, 20NG, and CIFAR-100 by converting class names into queries: "what is a *class*" and issuing them to two popular search engines, Google and DuckDuckGo. We scrape the resulting preview snippets, automatically remove scraper artifacts and partial descriptions, and manually filter any remaining off-topic descriptions. To construct **JSON descriptions** of an animal in AWA2, we use types of attributes as the keys (e.g. `color`) and the actual attribute lists as values (e.g., `{color:[orange, black]}` for a tiger). To improve the robustness of the model, we automatically augment the descriptions by randomly removing attribute values and permuting the key and value list orders, and divide them into training, validation, and test sets. We provide examples of textual and JSON descriptions and

collection and filtering in Appendix C.

4.3 Models

For text datasets, we encode input features ($f(\cdot)$) using the `[CLS]` representation from a pretrained `BERT-small` model (Turc et al., 2019). For image datasets, we use the activations of a ResNet-18 model (He et al., 2016) immediately preceding the fully-connected final layer. For each dataset, the input encoders are identical across our models and baselines. While the standard supervised baseline (SUP) uses an output matrix to obtain the logits, for our **SEMSUP** models, we encode output features ($g(\cdot)$) using the `[CLS]` representations from a pretrained `BERT-small` model for text descriptions and a pretrained CodeBERTa-small (HuggingFace) model for AWA2 JSONs. We evaluate three SEMSUP variations: **SEMSUP-ALL** samples from all available class descriptions during training. **SEMSUP-SINGLE** uses a single fixed randomly selected description for each class during training. **SEMSUP-NAMES** is a SEMSUP variant using class names instead of descriptions (e.g. "computer graphics"). We consider two strong **baselines** from prior work. **DEVISE** (Frome et al., 2013) and **GILE** (Pappas and Henderson, 2019) which use word-embeddings of class names. For all models, we use the cross-entropy loss for multi-class datasets and the binary cross-entropy loss for the multi-label dataset. We provide additional details about training in Appendix E.

5 Results

We now report our main results and refer the reader to Section A for ablation experiments.

5.1 Generalizing to unseen descriptions (S1)

In this scenario, the input classes are identical between train and test time, but the model is given unseen descriptions at test time. The ability to generalize to novel descriptions enables users to define classification problems over subsets of train classes simply by providing their own list of class descriptions. For each dataset, we keep train and test descriptions consistent for our models and baselines. Notably, for DEVISE and GILE we use the same train and test descriptions as SEMSUP rather than class names in this scenario.

We present the results in Table 2. For all models other than SEMSUP-ALL, there is a large gap between the performance when unseen descriptions (**UN**) and seen descriptions (**S**) are used, with similar trends on all datasets. Interestingly, SEMSUP-SINGLE also performs poorly with a drop of 25 points compared to SEMSUP-ALL on CIFAR-100. This result suggests that training with a multiple descriptions is important for generalization in the output encoder.

5.2 Generalizing to unseen classes (S2)

In this case, classes are partitioned into training ($\mathcal{Y}_{\text{train}}$) and test ($\mathcal{Y}_{\text{test}}$) classes, where ($\mathcal{Y}_{\text{train}} \cap \mathcal{Y}_{\text{test}} = \emptyset$). The models generalize to unseen classes in $\mathcal{Y}_{\text{test}}$ by using corresponding class descriptions for the test classes.

Model	RCV1		20 NG		CIFAR		AWA2	
	UN	S	UN	S	UN	S	UN	S
SUP	×	96	×	92	×	74	×	94
DEVISE	52	91	84	91	55	73	**92**	92
GILE	53	91	85	92	53	73	**93**	93
SEMSUP-SINGLE	38	96	71	92	47	72	61	93
SEMSUP-ALL	**91**	96	**93**	93	**71**	73	**93**	93

Table 2: **(S1)** Model performance on seen descriptions (**S**) and unseen descriptions (**UN**) at test time. SUP – supervised learning baseline. SEMSUP-ALL drops ≤ 5 points across all datasets when switching from seen to unseen class descriptions whereas the next best model (GILE) loses almost 40 points on RCV1. We report performance on the test set (all classes included) for all datasets. RCV1 uses the LRAP metric and other datasets use ACCURACY. Decimal places removed for clarity.

Model	RCV1	20 NG	CIFAR	AWA2
	LRAP	ACC.	ACC.	ACC.
DEVISE	27.1 (±3.0)	**71.2** (±1.5)	46.6 (±3.5)	25.0 (±3.1)
GILE	35.2 (±0.2)	**70.4** (±0.2)	51.0 (±1.3)	✗
SEMSUP-NAMES	44.6 (±0.9)	**74.0** (±4.5)	**58.5** (±2.3)	27.6 (±1.9)
SEMSUP-SINGLE	29.8 (±3.5)	63.8 (±1.0)	47.7 (±3.5)	33.6 (±2.1)
SEMSUP-ALL	**48.0** (±2.4)	**72.5** (±0.9)	**61.0** (±1.7)	**40.0** (±4.1)

Table 3: **(S2)** Mean performance on unseen test classes, which are a subset of classes not seen during training. The test classes are at the same level of hierarchy as train classes. SEMSUP models consistently outperform DEVISE and GILE. Bracketed numbers are standard deviation over 3 seeds and bolded numbers are statistically significantly higher than other numbers ($p < .05$). The GILE baseline is invalid for AWA2 because the word vectors used cannot handle JSON descriptions.

We present the results in Table 3. SEMSUP models significantly outperform DEVISE and GILE on three out of the four datasets, with performance improvements of 13 points on RCV1 and 10 points on CIFAR. SEMSUP-ALL is also able to generalize better to unseen classes in AWA2 while using JSON attributes, beating DEVISE and GILE which use the class name, validating that our framework can use different kinds of output descriptions effectively.

5.3 Generalizing to unseen superclasses (S3)

In this scenario, the classes at test time are unseen *superclasses* of the classes from the training set. For example, if $\mathcal{Y}_{\text{train}}$ includes *foxes*, *lions*, and *frogs*, $\mathcal{Y}_{\text{test}}$ can consist of *mammals* and *reptiles*. This task is more challenging because of the change in the granularity of classification.

We present the results in Table 4. Like in the previous scenarios, our SEMSUP models consistently outperform DEVISE and GILE, with improvements ranging from 10 points on 20 NG to 16 points on CIFAR-100.

Model	RCV1	20 NG	CIFAR
	LRAP	Acc.	Acc.
DEVISE	47.0 (±4.1)	76.1 (±2.7)	43.1 (±2.6)
GILE	46.2 (±0.1)	74.3 (±2.5)	41.9 (±1.9)
SEMSUP-NAMES	**56.1** (±3.0)	80.5 (±1.8)	54.8 (±2.9)
SEMSUP-SINGLE	44.6 (±2.4)	80.4 (±2.4)	53.2 (±2.5)
SEMSUP-ALL	**56.2** (±1.4)	**86.1** (±0.8)	**59.4** (±1.9)

Table 4: **(S3)** Mean performance of models on test superclasses, which consist of unseen supersets of train classes. SEMSUP-ALL outperforms all other models by significant margins on 20 NG and CIFAR and our SEMSUP models outperform baselines on all datasets. We do not report AWA2 scores because it is not provided with a hierarchical arrangement of the classes.

Model	RCV1→20 NG
MAJORITY	5.0 (±0.0)
DEVISE	7.9 (±1.2)
GILE	10.9 (±1.0)
SEMSUP-NAMES	**20.5** (±1.3)
SEMSUP-SINGLE	14.8 (±0.9)
SEMSUP-ALL	**19.5** (±1.0)

Table 5: **(S4)** Model transfer performance (Acc.) when trained on RCV1 and tested on 20NG (unseen task). SEMSUP outperforms DEVISE and GILE by approx $2\times$. MAJORITY always predicts the majority class.

5.4 Generalizing to unseen tasks (S4)

In this final scenario, we evaluate models under the condition of task transfer by training on the source task RCV1 and evaluating on the target task 20NG.

Table 5 shows that SEMSUP achieves over $2\times$ the accuracy of DEVISE and GILE, demonstrating the strong transfer performance of our model even though RCV1 and 20NG are very different types of classification tasks (multi-label and multi-class respectively) and contain different sets of classes. While there is scope for improvement, SEMSUP's unseen task generalization which will lead to model reuse in related tasks and domains.

5.5 Analysis: The effect of number of descriptions

We study four variants of SEMSUP which use $n = 1, 5, 10$ text descriptions at train time each. The models use the same set of descriptions at test time, and we evaluate them on two different scenarios, unseen descriptions and unseen classes, and report the results in Table 7.

Using a single description leads to poor performance for both scenarios, around 50 and 15 points lower than when 10 descriptions are used. This is likely because the output encoder learns spurious discriminative features from a single description that do not generalize. However, the likelihood of learning spurious features decreases as we increase the number of descriptions. Indeed, for both scenarios, we observe that the performance steadily increases as we increase the number of descriptions.

No. of desc.	Unseen descriptions	Unseen classes
$n = 1$	38.3 (±2.6)	35.6 (±1.2)
$n = 5$	81.9 (±1.1)	**49.3** (±3.0)
$n = 10$	**90.5** (±1.0)	**50.5** (±2.4)

Table 6: Mean performance on RCV1 when we vary the number of descriptions used to train SEMSUP models. We evaluate on two scenarios – (a) unseen descriptions and (b) unseen classes. Increasing the number of training descriptions significantly improve generalization performance, especially to unseen descriptions.

6 Discussion

We proposed semantic supervision (SEMSUP), a unified paradigm for providing semantic supervision to enable generalization over output spaces. SEMSUP represents output classes as dense feature vectors obtained from class 'descriptions', which allows models to generalize over unseen output spaces during training. Our results demonstrate the ability of models trained with SEMSUP to generalize to unseen descriptions, classes, superclasses, and tasks, while significantly outperforming prior work across four different datasets and two variants of supervision (text and JSON).

We view SEMSUP as a generalization of the standard supervised learning setup currently prevalent in the field (since classes can always be 'described' as abstract numbers). We believe this approach will enable better re-use of trained models for new tasks, new downstream applications, and by new end users, without requiring expensive re-training or fine-tuning procedures.

References

Samuel Acquaviva, Yewen Pu, Marta Kryven, Catherine Wong, Gabrielle E Ecanow, Maxwell Nye, Theodoros Sechopoulos, Michael Henry Tessler, and Joshua B Tenenbaum. 2021. Communicating natural programs to humans and machines. arXiv preprint arXiv:2106.07824.

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2015. Label-embedding for image classification. IEEE transactions on pattern analysis and machine intelligence, 38(7):1425–1438.

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2166–2179.

Sebastian Bujwid and Josephine Sullivan. 2021. Large-scale zero-shot image classification from rich and diverse textual descriptions. arXiv preprint arXiv:2103.09669.

Yann N Dauphin, Gökhan Tür, Dilek Hakkani-Tür, and Larry P Heck. 2014. Zero-shot learning and clustering for semantic utterance classification. In ICLR (Poster).

Berkan Demirel, Ramazan Gokberk Cinbis, and Nazli Ikizler-Cinbis. 2017. Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning. In Proceedings of the IEEE international conference on computer vision, pages 1232–1241.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Andrea Frome, Greg S Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: a deep visual-semantic embedding model. In Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2, pages 2121–2129.

Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In Proceedings of the conference. Association for Computational Linguistics. Meeting, volume 2018, page 1884. NIH Public Access.

Austin W. Hanjie, Victor Y Zhong, and Karthik Narasimhan. 2021. Grounding language to entities and dynamics for generalization in reinforcement learning. In Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 4051–4062. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

HuggingFace. Codeberta.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. 2020. Concept bottleneck models. In International Conference on Machine Learning, pages 5338–5348. PMLR.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In Machine Learning Proceedings 1995, pages 331–339. Elsevier.

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

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.

David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. Journal of machine learning research, 5(Apr):361–397.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021. Decaf: Deep extreme classification with label features. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pages 49–57.

Jesse Mu, Percy Liang, and Noah Goodman. 2020. Shaping visual representations with language for few-shot classification. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4823–4830.

Shikhar Murty, Pang Wei Koh, and Percy Liang. 2020. Expbert: Representation engineering with natural language explanations. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2106–2113.

Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In Thirtieth AAAI Conference on Artificial Intelligence.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2018. Grounding language for transfer in deep reinforcement learning. Journal of Artificial Intelligence Research, 63:849–874.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In NIPS.

Nikolaos Pappas and James Henderson. 2019. Gile: A generalized input-label embedding for text classification. Transactions of the Association for Computational Linguistics, 7:139–155.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

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. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2249–2257.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases, volume 11, page 269. NIH Public Access.

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 49–58.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In Advances in Neural Information Processing Systems, pages 935–943.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In Proceedings of the 2017 conference on empirical methods in natural language processing, pages 1527–1536.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 306–316.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. arXiv preprint arXiv:1908.08962v2.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2321–2331.

Catherine Wong, Kevin M Ellis, Joshua Tenenbaum, and Jacob Andreas. 2021. Leveraging language to learn program abstractions and search heuristics. In International Conference on Machine Learning, pages 11193–11204. PMLR.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. 2018. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE transactions on pattern analysis and machine intelligence, 41(9):2251–2265.

Honglun Zhang, Liqiang Xiao, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Multi-task label embedding for text classification. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4545–4553.

Li Zhang, Tao Xiang, and Shaogang Gong. 2017. Learning a deep embedding model for zero-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2021–2030.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. 2019. Rtfm: Generalising to new environment dynamics via reading. In International Conference on Learning Representations.

No. of desc.	Unseen descriptions	Unseen classes
$n = 1$	38.3 (±2.6)	35.6 (±1.2)
$n = 5$	81.9 (±1.1)	**49.3** (±3.0)
$n = 10$	**90.5** (±1.0)	**50.5** (±2.4)

Table 7: Mean performance on RCV1 when we vary the number of descriptions used to train SEMSUP models. We evaluate on two scenarios – (a) unseen descriptions and (b) unseen classes. Increasing the number of training descriptions significantly improve generalization performance, especially to unseen descriptions.

Output encoder	Pre-trained	Fine-tuned	Unseen classes
SEMSUP **(Linear)**	✓	✓	34.7 (±0.5)
SEMSUP ($\mathbf{L = 2}$)			**47.2** (±1.5)
SEMSUP ($\mathbf{L = 4}$) ⋆			**48.0** (±2.4)
SEMSUP ($\mathbf{L = 8}$)			**45.4** (±2.9)
SEMSUP **(R. I.)**	✗	✓	37.4 (±1.5)
SEMSUP **(Frozen)**	✓	✗	36.0 (±1.0)

Table 8: Mean performance on RCV1 when we vary the output encoder and evaluate on unseen classes. $\mathbf{L}$ represents the number of layers, **R. I.** means that the model is randomly initialized, and **Frozen** indicates that the output encoder's weights are fixed. Jointly optimizing the input and output encoders and using pretrained output encoders are both important for strong performance.

A Analysis

We corroborate in the following sections that the reasons for strong performance of SEMSUP are three-fold – *multiple* diverse descriptions of classes, *pre-trained* output encoders, and *fine-tuning* the output encoder. We use the RCV1 dataset for the following ablation experiments (§ A.1,A.2). We also compare the effect of different output modalities (natural language text v.s. JSONs with attributes) on AWA2 (§ A.3).

A.1 The effect of number of descriptions

We study four variants of SEMSUP which use $n = 1, 5, 10$ text descriptions at train time each. The models use the same set of descriptions at test time, and we evaluate them on two different scenarios, unseen descriptions and unseen classes, and report the results in Table 7.

Using a single description leads to poor performance for both scenarios, around 50 and 15 points lower than when 10 descriptions are used. This is likely because the output encoder learns spurious discriminative features from a single description that do not generalize. However, the likelihood of learning spurious features decreases as we increase the number of descriptions. Indeed, for both scenarios, we observe that the performance steadily increases as we increase the number of descriptions.

A.2 The effect of the output encoder

The output encoder, which encodes the descriptions corresponding to classes, is crucial to the success of SEMSUP, because it should learn discriminative yet generalizable features. We evaluate different encoders and report the results in Table 8. We use a 4 layer pre-trained model as our reference SEMSUP ($\mathbf{L = 4}$) and consider generalization to unseen classes (§ 5.2).

Varying the encoder complexity In the first segment of the table Table 8, we notice that using a linear model SEMSUP **(Linear)**, which computes a bag-of-vectors representation of the class descriptions, performs poorly when compared to non-linear models like SEMSUP ($\mathbf{L = 2}$). We experiment with three differently sized (non-linear) BERT (Devlin et al., 2019) models released by (Turc et al., 2019) which have 2, 4, and 8 layers, respectively, and notice no significant difference between their performance. While increasing the depth of the model typically leads to gains in performance, we speculate that the small number of class descriptions used (10 per class for 103, $10 \times 103 = 1030$) when compared to the number of instances (480,000 for RCV1) means that a very deep model is unnecessary. Thus, we conclude that while it is extremely crucial to use a non-linear output encoder, its depth is less important.

Pre-trained v.s. randomly initialized In the second segment, we notice that a 4 layer output encoder which is initialized using random weights – SEMSUP **(R. I.)** performs 10 points lower than our reference – SEMSUP ($\mathbf{L = 4}$), a pre-trained model of a comparable size. This underscores the importance of pre-training, which gives models a *better* semantic understanding of sentences than their randomly initialized counterparts.

Freezing the output encoder In the third segment, we consider SEMSUP **(Frozen)**, which initializes the output encoder using a pre-trained model, but freezes all the weights throughout training (no gradients are passed). We notice that even though this variant has the benefits of a pre-trained model, it is worse than our reference by over 10 points. This highlights the importance of fine-tuning, which adapts the weights of the pre-trained models to the task at hand. Interestingly, SEMSUP **(Frozen)** performs similar to the randomly initialized model – SEMSUP **(R. I.)**. Both these models are missing a key ingredient required for good generalization, with the former missing fine-tuning and the latter missing pre-training, and our reference (SEMSUP ($\mathbf{L = 4}$)) which includes both these aspects performs the best.

These experiments validate that the following three aspects are useful for strong generalization in SEMSUP models – (a) *multiple* diverse descriptions of classes, (b) *pre-trained* output encoders, and (c) *fine-tuning* the output encoder.

Supervision	Model	AWA Heldout
NL	SEMSUP-NAMES	20.5 (±1.3)
	SEMSUP-SINGLE	21.9 (±3.7)
	SEMSUP-ALL	32.3 (±2.42)
JSON	SEMSUP-SINGLE	33.6 (±2.1)
	SEMSUP-ALL	**40.0** (±4.1)

Table 9: Mean performance (Acc.) on test classes for SEMSUP models trained on natural language (NL) and (JSON). Numbers in brackets are stddev. over 3 seeds.

A.3 Comparing output supervision: NL vs. JSON

To compare the effect of different types of output supervision for SEMSUP, we evaluate its performance on AWA2 test classes using natural language and JSON supervision (Table 9). The best performing JSON model (SEMSUP-ALL) outperforms its corresponding model trained using natural language model by 8 points, demonstrating the utility of using structured data formats to provide class descriptions on this particular task. For AWA2, this performance gap between natural language and JSON could be due to two reasons: (1) the JSON class descriptions directly list all of the most salient features of each class making it more information dense than natural language descriptions, and (2) its structure is more consistent between training and test descriptions (since the keys are identical for example).

These results demonstrate the flexibility of the SEMSUP framework in allowing users to pick a type of semantic supervision that suits their task needs the best and hint at the possibility of developing more sophisticated forms of semantic supervision, which may even be a combination of both freeform text and more structured representations, to maximize generalization performance.

B Experimental Setup

B.1 Datasets

We evaluate SEMSUP on four diverse datasets chosen to test generalization on different scenarios (§ 3). **20 Newsgroups** (20NG) (Lang, 1995) consists of 20,000 newsgroup documents in 20 classes. We partition similar classes into 5 superclasses [1]. We further partition the classes into 12 train, 4 validation and 4 test classes, and evaluate using accuracy. **CIFAR-100** (Krizhevsky et al., 2009) consists of 60K images in 100 classes, each assigned to one of 20 superclasses (e.g. *aquatic mammals*). We partition the dataset into 80 train, 10 validation, and 10 test classes, and evaluate using accuracy. **Animals with Attributes 2** (AWA2) (Xian et al., 2018) is an animal classification dataset with 37K images and 50 classes. The dataset also includes 85 animal attributes (e.g. *fur*, *swims*). Each class is annotated with binary values indicating whether the attribute is present in the class. We follow the split of classes into 27 train, 13 validation, and 10 test classes provided in the dataset. **RCV1** (Lewis et al., 2004) is a multi-label news classification dataset (multiple correct classes per instance) with over 800,000 articles. We hold out 25 of the 103 niche classes to test unseen class generalization. We withhold 17 parent classes in the provided hierarchy to test unseen superclass generalization. We evaluate using the label ranking average precision (LRAP) metric (Pappas and Henderson, 2019). Further details about the experimental setup are provided in

B.2 Collecting output supervision

Text output supervision We collect textual output descriptions for RCV1, 20NG, and CIFAR-100 by converting class names into queries: "what is a *class*" and issuing them to two popular search engines[2]. We scrape the resulting preview snippets, automatically remove scraper artifacts and partial descriptions, and manually filter any remaining off-topic descriptions.[3]

JSON output supervision To construct JSON descriptions of an animal in AWA2, we use types of attributes as the keys (e.g. `color`) and the actual attribute lists as values (e.g., {`color:[orange, black]`} for a tiger). To improve the robustness of the model, we automatically augment the descriptions by randomly removing attribute values and permuting the key and value list orders, and divide them into training, validation, and test sets. We provide examples of textual and JSON descriptions and details regarding their collection and filtering in Appendix C.

B.3 Models

For text datasets, we encode input features ($f(\cdot)$) using the `[CLS]` representation from a pretrained `BERT-small` model (Turc et al., 2019). For image datasets, we use the activations of a ResNet-18 model (He et al., 2016) immediately preceding the fully-connected final layer. For each dataset, the input encoders are identical across our models and baselines, to ensure fair comparison. While the standard supervised baseline (SUP) uses an output matrix to output the logits, the other models use output encoders, which we describe below.

For our **SEMSUP** models, we encode output features ($g(\cdot)$) using the `[CLS]` representations from a pretrained `BERT-small` model (Turc et al., 2019) for text descriptions and a pretrained CodeBERTa-small[4] model for AWA2 JSONs. We propose three SEMSUP models:

1. **SEMSUP-ALL** samples from all available class descriptions during training.

[1] We follow the division from: `http://qwone.com/~jason/20Newsgroups/`

[2] `www.google.com` and `www.duckduckgo.com`

[3] We do not manually filter RCV1 descriptions due to a large number of descriptions and classes.

[4] `https://huggingface.co/huggingface/CodeBERTa-small-v1`

2. **SEMSUP-SINGLE** uses a single fixed randomly selected description for each class during training.

3. **SEMSUP-NAMES** is a SEMSUP variant using class names instead of descriptions (e.g. "computer graphics").

We consider two strong **baselines** from prior work.

1. **DEVISE** (Frome et al., 2013) uses the mean of the word vectors of the *class names*.

2. **GILE** (Pappas and Henderson, 2019) uses the mean of the word vectors of *class descriptions*, with modeling similar to DEVISE other than the $\tanh$ activation applied to the output embeddings.

For both the models, we use GloVe (Pennington et al., 2014) vectors which are fixed throughout training (Frome et al., 2013) to represent the output.

Training We train all models end-to-end and back-propagate gradients both through the input and output encoder. For all models, we use the cross-entropy loss for multi-class datasets and the binary cross-entropy loss for the multi-label dataset. We provide additional details about training in Appendix E.

C Output Supervision

We show example class descriptions for the datasets in table 10. The results were scraped from `www.google.com` and `www.duckduckgo.com` using a third-party scraping tool[5]. Data collection was conducted between September 2021 and January 2022. To reduce variability, personalized results were turned off and regions were fixed to United States. Safe search was enabled for `www.google.com` and set to moderate on `www.duckduckgo.com`. The number of search returns for `www.google.com` was varied between 10 and 50. While we obtained more descriptions using a higher number of search returns, we found that the quality and relevance was often lower.

[h]

An example scraping target is presented in Figure 2. We automatically filter the scraped preview blocks by removing any incomplete sentences. For multi-sentence descriptions, we only take the first sentence. Sentences that are less than 5 words are discarded. After automatic filtering, we manually inspect the descirptions and remove irrelevant descriptions. The mean number and lengths of the collected descriptions is presented in Table 11. On all datasets, we divide the class descriptions into a 60-20-20 train-val-test split.

D Dataset Details

D.1 RCV1

RCV1 contains $800,000$ articles and we create a 60:20:20 split for train, validation, and test respectively. It contains 103 classes.

[5] `www.webscraper.io`

Dataset (class)	Description
RCV1 (Consumer Prices)	A consumer price index is a price index, the price of a weighted average market basket of consumer goods and services purchased by households.
20-NG (Cryptography)	Cryptography is the study and practice of sending secure, encrypted messages between two or more parties
CIFAR-100 (Flatfish)	A category of fish that are characterized by their narrow bodies that are flat and oval-shaped.
AWA2 (Killer Whale)	{appendages: [flippers, tail], behavior: [fierce, smart, group], color: [black, white], diet: [fish, meat, plankton, hunter], habitat: [arctic, coastal, ocean, water], mobility: [swims, fast, strong, active, agility], shape: [big, bulbous, lean], skin: [patches, spots, hairless, toughskin], teeth: [meatteeth, strainteeth]}

Table 10: Randomly selected example class descriptions for RCV1, 20-NG, CIFAR-100, and AWA2 for a randomly selected class in each dataset.

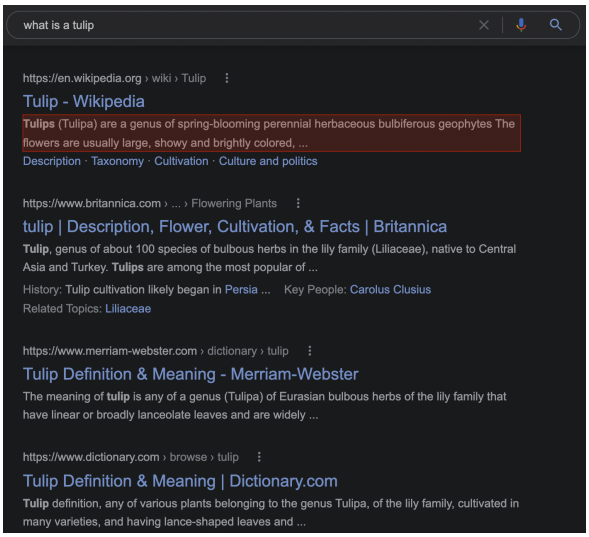

Figure 2: Example scraper selection from the search query for the class *tulip* in CIFAR100.

Dataset	Num Descriptions	Description Lengths
RCV-1	17.9 ± 4.3	17.4 ± 9.7
20 NG	19.2 ± 3.8	17.7 ± 7.1
CIFAR-100	20.3 ± 5.9	16.7 ± 7.1
AWA2	1250 ± 0.0	30.6 ± 5.9

Table 11: Statistics of the collected class descriptions including mean number of descriptions per class and mean lengths per description. Note that on AWA2 we automatically augment the descriptions, so there is no variance in the number of descriptions between classes.

Val Classes	alt.atheism comp.sys.mac.hardware rec.motorcycles sci.electronics,
Test Classes	comp.os.ms-windows.misc rec.sport.hockey sci.space talk.politics.guns
Val Superclasses	recreation religion
Test Superclasses	computer science politics

Table 12: Details for the 20 NG dataset. Training classes are the remaining 12 classes not in val classes or test classes.

Val Classes	streetcar, rabbit, man lamp, forest, otter crab, crocodile, house orchid
Test Classes	motorcycle, pine_tree, bottle trout, chair, butterfly chimpanzee, orange, leopard possum
Val Superclasses	large_omnivores_and_herbivores medium_mammals, people large_man-made_outdoor_things insects, household_electrical_devices food_containers, fish flowers, vehicles_2

Table 13: Details for CIFAR-100. Training classes are the remaining 80 classes not in val classes or test classes. The test superclasses are the remaining 10 superclasses not listed in the val superclasses above.

D.2 20NG

We use the 18828 variant for each newsgroup. Since the original dataset does not define train-test splits, we construct our own 80-20 train test split. We further divide the training set into training and validation sets with a porportion of 80-20.

We present details of the 20 NG dataset splits in table 12. When evaluating generalization to superclasses on 20 NG we remove the `misc.forsale` class since it is its own superclass.

D.3 CIFAR-100

. We use the provided train-test split, but divide the train set 80-20 into training and validation examples.

D.4 AWA2

We use the predefined train-val-test splits of classes provided in the paper (Xian et al., 2018). We use only the second of the three train-val splits provided. We split the instances into train and test examples 80-20 and further divide the training set 80-20 into training and validation examples.

To construct the JSON, we first assign each attribute to a parent attribute. The final class-level JSON consists of the parent attributes as keys, and the values are attributes that are present in the class. We augment this dataset by first adding 50 samples per class of corrupted examples, by randomly deleting attributes independently with probability 0.15, and then further multiplying this by 25 permutations.

E Model Training and Evaluation

All models are end-to-end differentiable and we train them using the AdamW optimizer (Loshchilov and Hutter, 2017). We use a constant learning rate of 1×10^{-4} for all the vision experiments on AWA2 and CIFAR-100 and a constant learning rate of 2×10^{-5} for all experiments on 20 NG. For efficiency, the class descriptions are encoded into the output matrix $\mathcal{O}^{\text{SemSup}}$ at each minibatch, so that all instances in the batch share the same output matrix. We use the validation set for early stopping, and test checkpoints saved at the point of highest validation accuracy. All implementation was done in PyTorch and PyTorch Lightning and experiments were run on either a single NVIDIA RTX2080 or a single NVIDIA RTX3090.