Learning Image Representations to Understand and Predict Semantic Hierarchies

Lisa Wang
Stanford University
450 Serra Mall, Stanford, CA 94305
lisa1010@cs.stanford.edu

Ajay Sohmshetty
Stanford University
450 Serra Mall, Stanford, CA 94305
ajay14@stanford.edu

Abstract

Humans are very good at recognizing and labeling seen objects at different levels of the semantic hierarchy. If a human is not sure about what particular breed of dog she sees, she would just recognize it as a dog, the finest label she is confident about. To get closer to general human visual recognition especially with fewer training data and on images from unseen classes, we hypothesize that designing deep neural network architectures to learn representations which lend themselves to predictions at different levels is a crucial stepping stone. We defined a novel hierarchical object classification task where a model should be able to predict the most fine-grained label that the model is confident about. To tackle this task, we investigated several model architectures and proposed a chained convolutional and recurrent neural net architecture (CNN-RNN) as shown in Figure 1. We evaluated our models on the CIFAR-100 dataset of color images, each of which belongs to one of 100 fine-grained classes and one of 20 coarse-grained superclasses.

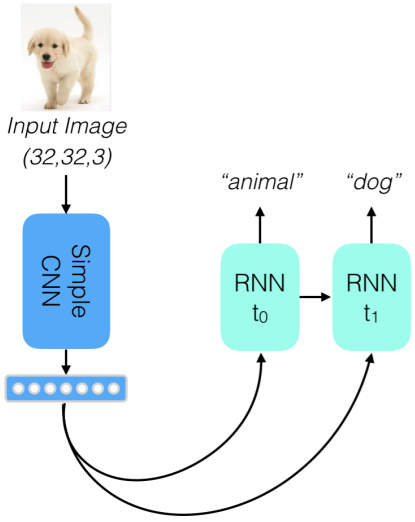


Figure 1. Proposed CNN-RNN model which predicts the semantic hierarchy for a given input image.

1. Introduction and Motivation

When a human examines an object, she does not simply identify it by its most specific semantic label (e.g. "Golden Retriever"). She recognizes the object in the context of its semantic hierarchy: "animal" → "dog" → "Golden Retriever". Each object she identifies further contains a dynamic hierarchical structure, the depth of which depends on a confidence, determined based on prior knowledge, and differs from person to person. For example, while one person may predict "animal"→ "dog" → "Golden Retriever", another may identify the object as simply "animal" → "dog", having never seen a Golden Retriever before. Even though this (poor) human has never seen a Golden Retriever before, she is still able to make a decent prediction about the object at hand, which is better than guessing some other, completely incorrect fine-grained label (like "Yorkie"). While Convolutional Neural Networks (CNNs) have achieved notable performance on visual recognition and detection tasks, they are nevertheless subject to this pitfall, and do not mimic the natural paradigm of hierarchical predictions. Inspired by models originally designed for image captioning, this paper attempts to bridge this gap.

The traditional method of training feed-forward CNNs for object recognition is to provide a large dataset with images of objects and corresponding labels. However, every object is part of a semantic hierarchy and could be labeled at different levels of the hierarchy. It thus makes sense to include this contextual information in training to improve image classification tasks and the learned image representations, but is often omitted, and classes are treated as fully discrete and non-overlapping labels.

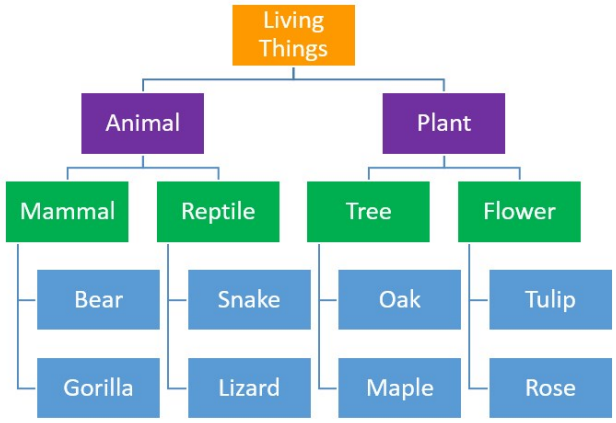


Figure 2. An example of a semantic hierarchy tree. Source: `http://thepeakperformancecenter.com/wp-content/uploads/2016/08/Hierarchies.jpg`

2. Definitions and Proposed Task

2.1. Semantic Hierarchy

A semantic hierarchy categorizes objects into classes and subclasses, resulting in a semantic hierarchy tree. Figure 2 shows an example of a semantic hierarchy tree with four levels. The most coarse-grained class *Living Things* is located at the root, while each child of a node contains a finer-grained subclass of its parent. For example, *Animal* is a subclass of *Living Things* while *Mammal* is a subclass of *Animal*. A semantic hierarchy path is then a path from a coarser-grained node to a finer-grained node in one of its subtrees. In Figure 2, valid paths would be:

1. *Living Things* $\rightarrow$ *Animal* $\rightarrow$ *Mammal* $\rightarrow$ *Gorilla*.
2. *Living Things* $\rightarrow$ *Plant*.
3. *Reptile* $\rightarrow$ *Snake*.

Given the CIFAR-100 data set which contains two levels of the semantic hierarchy (coarse-grained and fine-grained), we will focus on predicting two levels of the hierarchy. We will describe the dataset in more detail under Experiments.

2.2. Hierarchical Image Classification Task

The specific task we propose is to predict a single label for a given input image, which can either be a coarse-grained label or a fine-grained label, depending on what the model has trained on. The fine-grained label is preferred whenever the model has previously trained on that fine-grained label before. We call this task the Hierarchical Image Classification Task.

The model first trains on a set of images that are labeled with both their coarse-grained and fine-grained classes. Then the model is also provided with some images that are only labeled with their coarse-grained class, so the model can learn when to output just the coarse-grained label.

If the model has trained on the fine-grained class of the image before, then it should ideally predict the fine-grained label. Otherwise, the model should predict the coarse-grained label given that the model has at least trained on other images from the same coarse-grained class.

To illustrate, consider the following three cases with examples:

1. The input image shows an *orchid* (fine-grained class), which is part of the coarse-grained class *flowers*. The model has trained on images of orchids labeled with both the fine-grained class *orchid* and the coarse-grained class *flowers* before. Hence, for this image, the correct prediction would be the fine-grained class *orchid*.

2. The input image shows a *rose* (fine-grained class),which is part of the coarse-grained class *flowers*. The model has trained on images of *rose* before, but those images of *rose* where only labeled with the coarse-grained class *flowers*. So even though the model has seen roses before, it has never trained on images labeled with the fine-grained class *rose*. Thus, the correct prediction for this input image would be the coarse-grained class *flowers*.

3. The input image shows a *tulip* (fine-grained class), which is also part of the coarse-grained class *flowers*. The model has never trained on images of *tulip* before. However, since the model has trained on other types of flowers like *orchid*, the correct prodiction for this input image would be the coarse-grained class *flowers*. This is comparable to a human who has never seen or heard a description of tulips before, but is usually still able to classify a tulip as a flower.

3. Related Work

From our review of related work, our proposed task hasn't been addressed before. Nevertheless, our work does build upon previous efforts to incorporate semantic hierarchical information in order to achieve more generalizable image representations and better results on image recognition tasks. We also took inspiration from caption-generating models which combine convolutional neural nets with recurrent neural nets [3].

3.1. Semantic Hierarchies for Visual Object Recognition

Prior work has been done utilizing the semantic structure of the dataset to improve image tasks. For example, Marszalek et al. uses semantic hierarchies for visual object detection, and used their network for inference for part

detectors [7]. Key features of this model include combining a binary detector with the extracted semantic graph from WordNet, and then training multiple SVM classifiers on exemplars representing relationships within the semantic graph.

While this model is able to capture inter-class relationships and integrates it into the visual appearance learning procedure, and reduces the classifier complexity in the number of classes, it does not use deep learning as an approach, instead opting for a bag-of-features model and a set of SVMs with Gaussian kernels.

3.2. Hierarchical Deep CNN (HD-CNN)

Yan et al. proposed a hierarchical deep CNN (HD-CNN) architecture, which embeds CNNs into a category hierarchy [9]. Specifically, the HD-CNN distinguishes coarse superclasses with a coarse category classifier and uses multiple fine category classifiers to distinguish between fine classes. The HD-CNN model is pretrained component-wise, and fine-tuned afterwards.

The authors were able to achieve state-of-the-art results on CIFAR-100 and ImageNet 1000-class. With different variations of HD-CNN, they were able to improve the top-1 classification error by 1.1% to 2.65%.

For their experiments, the authors adopted a Network-in-Network (NIN) [5] with three layers. They chose the NIN architecture since it reduces the number of parameters significantly compared to other models with similar performance. The fine category components share the first three convolutional layers. A few layers forming the "coarse component" are stacked on top of the shared layers, and trained to predict the coarse class. For each coarse class, additional independent layers are stacked on top of the shared layers and output predictions over fine classes for the respective coarse class. The additional layers are also referred to as the "fine component." To obtain the final prediction, the outputs of the coarse component and all fine components are taken into account. Figure 3 shows a an illustration of the HD-CNN architecture.

3.3. Generating Image Descriptions

Karpathy et al. proposed an alignment model combining Convolutional Neural Networks with Bidirectional Recurrent Neural Networks with the goal to generate descriptions of image regions [3]. During training, the model is provided an input image together with a sentence description. The image is first passed through a CNN. The embedding output of the CNN subject to a linear transformation is then passed into the RNN to initialize its hidden state, as illustrated in Figure 4. Additionally, the words in the sentence are sequentially passed into the RNN, and the prediction at each timestep is supposed to be the next word in the sentence which describes the image. The main idea here is that the RNN uses the context from the CNN together with the word from the previous timestep to generate the word at the next timestep.

We took inspiration the combined architecture containing both a CNN and an RNN to design our CNN-RNN model for semantic hierarchy predictions. However, there are important differences between our proposed CNN-RNN architecture and the one described by Karpathy et al., e.g. we do not initialize the hidden state using the CNN embedding, but we pass the embedding itself as an input at every timestep. Our model is described in further detail under section **4**.

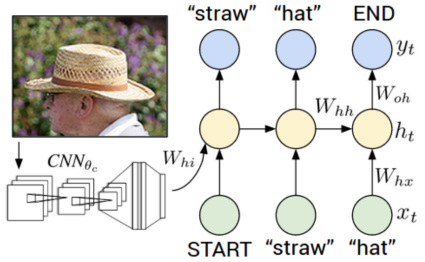


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model proposed by Karpathy et al. in [3].

3.4. Show, Attend, and Tell: Attention Mechanism

Kelvin Xu et. al. in [8] describe two novel approaches, for the task of image captioning: a "soft" but deterministic attention mechanism, trainable via backpropogation, and a "hard" stochastic attention mechanism. This method has been shown to improve image captioning BLEU scores, so we have gained inspiration from this paper to use similar attention mechanisms to not only 1) improve model results, but also 2) use this framework to visualize what the model is seeing at each stage in the iteration of the RNN and use it to make further conclusions and directions.

Figure 5. Example of attention mechanism in action.

4. Our Models

Both models we propose below contain a CNN as part of their architectures. Since our focus is mainly on new model architectures that go beyond CNN variations, we wanted

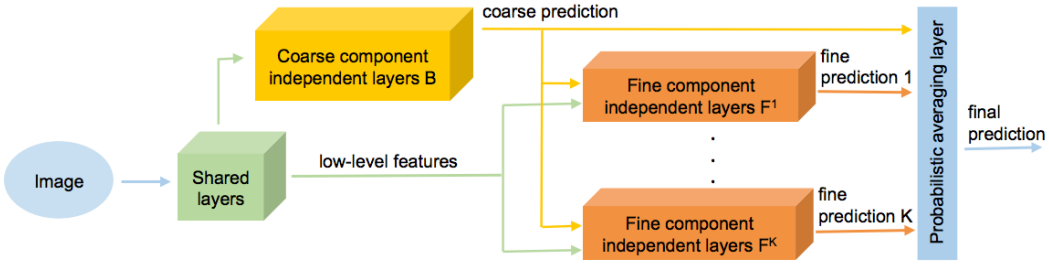


Figure 3. Hierarchical Deep Convolutional Neural Network (HD-CNN) Architecture, as proposed by Yan et al. Source: Yan et al. [9]

to choose a CNN architecture which performs reasonably on simple classification tasks but is also small enough to converge quickly during training. To decide on the exact structure of this core CNN, we experimented with several types of existing convolutional neural network architectures, inspired by frameworks that performed well in CIFAR-100 competitions in the past. To converge in a reasonable amount of time during training, we removed a number of the final convolutional layers in these preexisting architectures.

We ran three baselines: a LeNet-style CNN, a VGGNet-style CNN, and our own designed CNN, which combines features of the other two networks. We will refer to the latter architecture as "SimpleCNN". See 6 for an overview of this network. Based on the classification accuracies on CIFAR-10 and CIFAR-100 fine as well as the convergence speed, we decided to go with the "SimpleCNN" architecture.

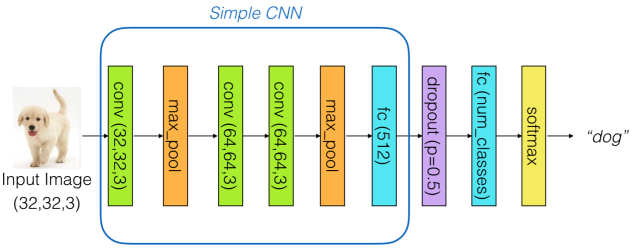


Figure 6. SimpleCNN Architecture. It has three convolutional layers with max pooling and dropout. The layers in the box are independent from the number of classes and can therefore be easily shared.

4.1. Branching Lever Model

This model is our baseline model to predict labels at different levels of the semantic hierarchy. The Branching Lever model is implemented on top of the SimpleCNN architecture, as shown in Figure 7. After the shared SimpleCNN layers, the network splits into coarse layers on one branch and fine layers on the other branch. At the end of the coarse branch, the model makes a coarse prediction out of 20 coarse-grained classes, and at the end of the fine branch, the model makes a fine prediction out of 100 fine-grained classes. To train the model, we compute the respective losses L_{coarse} and L_{fine} and minimize the combined loss $L_{joint} = L_{coarse} + L_{fine}$.

To tackle the Hierarchical Image Classification Task as described in section **2.2**, we add a lever on top of the two branches, which decides whether to output the fine-grained or the coarse-grained prediction, as illustrated in Figure 8. If the confidence on the fine-grained prediction is above a certain threshold, the lever decides that the model should output its fine-grained prediction, otherwise the model outputs the coarse-grained prediction.

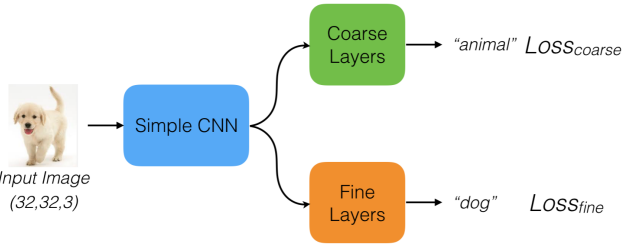


Figure 7. Branching Lever model. During training, we compute both the coarse and the fine loss, and minimize their combined loss.

4.2. CNN-RNN

Recurrent neural networks (RNNs) have the powerful the ability to make dynamic-sized predictions. It is thus quite natural to use RNNs for our task of making hierarchical predictions, which are also dynamic in structure. Of course, current state-of-the-art image classification frameworks use CNNs, not RNNs, where a final fully connected layer makes the final prediction. It is thus not prudent to forgo the CNN architecture completely, but instead intuitive to replace the final layers with a recurrent network layer architecture. These types of models have been used previously in the context of image captioning. However, while image captions are inherently subjective and open-ended, for this task, hierarchies are objective and explicit. Further, since our model is fully end-to-end differentiable, our

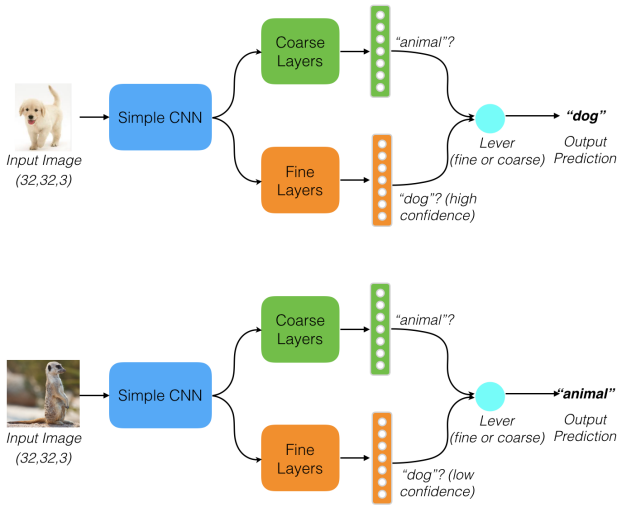


Figure 8. Branching Lever model. These two examples show the desired behavior for two input images. If the confidence score fine prediction is below a certain treshold, the model outputs the fine-grained prediction, e.g. "dog" in the example at the top, otherwise it outputs the coarse-grained prediction, e.g. "animal" in the example at the bottom.

model has the ability to learn these hierarchies.

An overview of this model is shown in Figure 1. Our convolutional layer, SimpleCNN, does the initial processing to produce a 512 dimensional feature vector, which is then fed into each input of the RNN. For objects that contain a shorter hierarchy length than others, we designate end by replacing the fine label with an **END** token. For example, observe that in Figure 9: Left, the input image makes both "animal" and "dog" predictions, since the model was previously trained on dogs. However, while the model was trained on several different types of animals, the model did not see any "meerkats". We can thus train our model to predict the **END** token and thus to dynamically stop predicting at various points of an object's semantic hierarchy.

We used Long-Short Term Memory (LSTM) RNN network. In our implementation, the initial hidden vector state is randomly initialized, and each time step the RNN receives exactly same input: the features outputted from the SimpleCNN. Each hidden vector representation after each step of the RNN is then attached to a fully connected layer that makes a prediction across the size of the vocabulary, which, in this case is 100 fine labels + 20 coarse labels + 1 **END** token = 121 total.

Another paradigm we examined was to pass a single feature representation from the CNN as input to the RNN as its initial state, and have no more inputs to the RNN across the rest of the time steps. However, we believed this paradigm would not work well, after discussions with Amir and Silvio, especially for longer hierarchies.

Since there are multiple predictions being made per image, we defined our loss to be a weighted average of the cross entropy loss for each prediction made. We also experimented with different weightings of the cross entropy loss, and describe our findings in the results section below.

5. Dataset:CIFAR-100

The CIFAR datasets including CIFAR-10 and CIFAR-100 were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [4]. The CIFAR-100 dataset has 100 classes with 600 RGB images per class. Each class is split into 500 training images and 100 test images. In addition, the 100 classes are grouped into 20 superclasses, each of which has 5 fine classes. For example, "trout" is fine class that belongs to the "fish" coarse super class. Table 1 describes the coarse classes and their corresponding fine classes. Due to the existence of coarse and fine classes, the CIFAR-100 dataset us suited for our task, which requires hierarchical semantic information.

We preprocessed the images by normalizing them and augmented the training set by randomly cropping and flipping images to prevent the network from overfitting on biases.

5.1. Training Set Split

We want our model to learn how to predict at multiple levels of the semantic hierarchy (both coarse-grained and fine-grained), but also learn when it should only predict the coarse-grained label. Hence, we decided to restructure the training set as follows:

1. Training set A: This set contains training images from 40 fine-grained classes, so 20000 images. When the model trains on images from this set, both the coarse-grained and the fine-grained labels are provided. Hence, the images in this set are "fully-labeled". In the Branching Lever model, training set A is used to train the model by minimizing both the coarse and fine losses together. In the CNN-RNN model, training set A is used to pretrain the CNN portion for the warm-start.

2. Training set B: This set contains training images from 80 fine-grained classes, so 40000 images, containing all of the images in the **fully-labeled** training set. All images in the 40 fine-grained classes in training set A are still fully-labeled, but all images in the remaining 40 fine-grained classes are only labeled with their coarse-grained class. The purpose of training set B is to teach the model when it should predict the fine label or the coarse label during the Hierarchical Image Classification task. In the Branching Lever model, training set B is used to tune the lever threshold. In

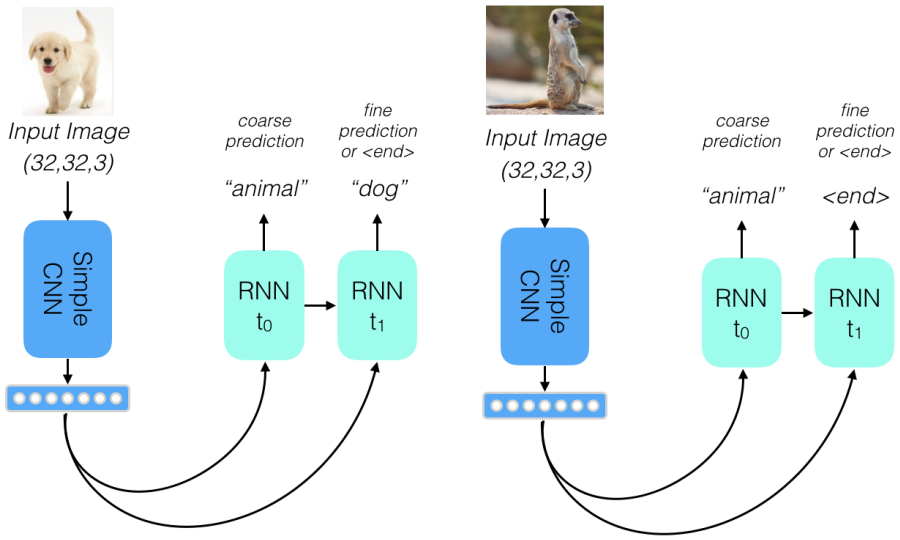


Figure 9. CNN-RNN network architecture. Left: Classifying a previously seen image ("dog"). Right: Classifying a previously unseen fine grained class image ("meerkat").

Table 1. CIFAR-100 Coarse and Fine Classes. There are 20 coarse superclasses, each of which with 5 fine subclasses.

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

the CNN-RNN model, training set B is used to train the CNN-RNN model end-to-end. For images without fine-grained labels, the model should predict coarse-grained label and the ¡end¿ token, as shown on the right side of Figure 9.

Note that our models never train on the training images in the remaining 20 fine-grained classes. We purposefully held these images out, so we can evaluate how well our models can generalize to images from unseen fine-grained classes and still predict their coarse-grained classes correctly.

5.2. Test Set Split

Since there are three cases in which test images can fall into (as described in section **2.2** as well), we split the

CIFAR-100 test set of 10000 images into three subsets A,B, and C respectively. Test subsets A, B and C do not overlap. Together, images in A,B and C form the entire test set. The split might seem very complex at first, but is necessary to to evaluate our models rigorously.

1. **Test set A** (*orchid* example in **2.2**): This set contains all test images belonging to 40 fine-grained classes the model has previously trained on (same 40 fine-grained labels as training set A). Specifically, for all the training images in these 40 fine-grained classes, the images were labeled with both the coarse-grained and the fine-grained classes during training. Hence, the correct hierarchical prediction for any image in test set A should be its fine-grained label. Since test set A contains images from 40 out of 100 fine-grained classes, it makes up 40% of the test set, containing 4000 test images.

2. **Test set B** (*rose* example in **2.2**): This set contains all test images belonging to another 40 fine-grained classes the model has previously trained on (additional 40 fine-grained classes contained in training set B). However, in contrast to test set A, for all the training images in these 40 fine-grained classes, the images were labeled only with the coarse-grained but **not** the fine-grained classes during training. So the model never trained on images explicitly labeled with these fine-grained classes. Hence, the correct hierarchical prediction for any image in test set B should be its coarse-grained label. Since test set B contains images from 40 out of 100 fine-grained classes, it makes up 40% of the test set, containing 4000 test images.

3. **Test set C** (*tulip* example in **2.2**): This set contains all test images belonging to the remaining 20 fine-grained classes the model has never trained on. This means that the model has never seen any image from any of these 20 fine-grained classes before during training, which was not the case with test set B. Thus, test set C poses a more challenging task to the model than test set B. The correct hierarchical prediction for any image in test set C is thus the coarse-grained label. Since test set C contains images from 20 out of 100 fine-grained classes, it makes up 40% of the test set, containing 2000 test images.

6. Evaluation

6.1. Quantitative Metrics

Based on the Hierarchical Image Classification task description under section **2.2**, we will use two metrics to compare our models:

1. *Coarse Accuracy*: This is the accuracy on the coarse-grained predictions. Both models always predict a coarse label for each input image. Since every image has a true coarse label, the *coarse accuracy* is simply the fraction of correct-grained label predictions.

2. *Hierarchical Accuracy*: This metric measures the accuracy on the Hierarchical Image Classification Task, as defined in section **2.2**. Since for this task, each input image exactly one "correct" output, which is either a coarse or fine label depending on what the model has previously trained on, the *hierarchical accuracy* is just the fraction of correct hierarchical predictions.

6.2. Qualitative Evaluation

In addition to quantitative metrics, we also look at predictions on randomly chosen images in the three different test sets to gather information about samples the models do well or poorly on.

7. Results and Discussion

We implemented our models using TensorFlow and TFlearn [1], running on AWS GPU P2 instances, equipped with NVIDIA Tesla K80 GPUs. Our experimental results are summarized in Table 2. Note that we are most interested in hierarchical accuracy. Here, we see that the Branching Lever model outperforms the CNN-RNN model in Test Set A, but CNN-RNN outperforms on all other test sets and the overall hierarchical accuracy is 10.63% better than Branching Lever. The most interesting result is for Test Set C, since it is the most challenging set, as discussed in section **5.2**. The results indicate that the CNN-RNN model outperforms the Branching Lever model, by about 4% on Test Set C, though both of these accuracies are still quite low. However, it is also worth emphasizing that the CNN-RNN model is generalizable to arbitrary semantic hierarchies with more than two levels, which cannot be said for the Branching Lever model. We postulate that with additional architectural features as described in the future work section, we are confident we will see a more drastic increase in performance of the CNN-RNN model. We also hypothesize that both of these models are limited by their SimpleCNN substructure, resulting in their poor hierarchical accuracy performance on the unseen training examples, and we believe we can see better results by using a more state-of-the-art convolutional layer.

In order to train the CNN-RNN model, we had to "warm start" the SimpleCNN. Otherwise, the model was getting stuck within a local minima. We first pretrained our SimpleCNN on Training Set A, and then transferred these weights over to our CNN-RNN model architecture, which resulted in much higher accuracies. Our best CNN-RNN model results are displayed in Figure 10 and the specific optimal metrics are given in Table 3. Here, we define joint accuracy as the accuracy of the model getting BOTH coarse

and fine grain predictions correct. Fine accuracy is defined as the accuracy of the model correctly predicting the second token, whether it is the **END** token or a specific fine grained class. Finally, average accuracy is defined as the average accuracy between the coarse and fine accuracy.

Note that, this model begins to overfit as evident from the loss graph, so we stopped training at an appropriate time. In Figure 10, we show the results for the end-to-end trained CNN-RNN. During training, we see that the model first optimizes for coarse loss, before turning back and optimizing for fine loss. In addition, fine accuracy begins at around 50% because half of the fine labels are marked with **END** token.

As mentioned earlier, a fully end-to-end CNN-RNN model was found difficult to train. Initially, we used our best SimpleCNN framework to produce static feature representations of input images, which were then used as inputs to the RNN layer. These results are displayed in Figure 11. Note that the same trends as described above in the end-to-end model appear in this model as well, though each accuracy metric is lower.

7.1. Weighted loss exploration

We finally experimented with different weightings assigned to each loss (coarse and fine). This was motivated by the fact that, an incorrect coarse prediction should be penalized far more than a correct coarse prediction, but an incorrect fine prediction. Contrary to our expectations, our initial results did not indicate any improvement upon the original equal weighted average loss method. These results are summarized in Figure 12. While coarse accuracy is prioritized to train faster, we see that there is no change in hierarchical accuracy, though joint accuracy sees a curious jump. We believe this may be due to the fact that this occurs because coarse accuracy is now slightly higher, and since 50% of all fine labels are **END** tokens, it is now easier to get both labels correct. However, since there is no change in hierarchical accuracy, our metric of interest, this exploration proves fruitless in attempting optimize our hierarchical accuracy. We tried many different weightings of coarse accuracy, from 0.6-0.9, and found that 0.6 had the best results (depicted in Figure 12).

7.2. Qualitative Analysis

By examining actual predictions of our two models, we can make a few empiric observations. The coarse and fine predictions of the Branching Lever model are not always aligned, meaning the predicted fine label is not always a subclass of the predicted coarse label, e.g. in Figure 24, the fine prediction is *keyboard* while the coarse prediction is *large_man-made_outdoor_things*. Due to the architecture of the branching lever model, this observation is not too surprising, since the fine branch doesn't use the output of the coarse branch. Interestingly, we weren't able to make this observation on the CNN-RNN model. When it predicted a fine-grained label, it was always a subclass of the previously predicted coarse-grained label. This shows that the sequential nature predictions in the RNN helps the fine-grained prediction make use of the coarse-grained context from the previous timestep, similar to what we saw in the caption-generating model proposed by Karpathy et al. [3]. The Appendix contains a list classification examples for both the CNN-RNN model as well as the branching lever model, with further observations.

8. Future Work

8.1. Attention Model

An immediate next step we would like to take is to implement an attention model that would attend over relevant sections over the input image, instead of feeding in the entire image into the LSTM RNN. One method of carrying out this would be the Show, Attend, and Tell model we described earlier in the Related Works section. We believe that this would provide two main benefits: 1) we would be able to visualize where in the input image the model attends to, as it iterates through the semantic hierarchy. This could prove quite interesting, and shed light into how the model makes decisions. 2) We believe this mechanism would improve accuracy as it would add complexity and structure to our model. We postulate that this attention mechanism could help to filter out irrelevant portions of the image, while focusing on relevant portions.

8.2. Exploring More Complex Semantic Hierarchies

So far, we have run our experiments on CIFAR-100 with two levels of semantic hierarchy and a fixed number of fine-grained classes per coarse-grained class. However, this structure only captures a tiny fraction of real-world semantic hierarchies. Semantic hierarchy trees even in a domain like "animals" are usually asymmetric trees, with vastly different number of subtrees per tree node and different depths for the leaf nodes. Hence, we would like to explore how well our model can generalize to more degenerate semantic hierarchies.

While the Branching Lever model doesn't lend itself easily to this generalization since we would have to add one branch for each additional hierarchy level, the architecture of our proposed CNN-RNN model should be able to handle any semantic hierarchy path. This is a big advantage of the CNN-RNN model over the Branching Lever model.

As defined earlier, the semantic hierarchy path is the tree path from coarser-grained node of the semantic hierarchy tree to a more fine grained node. The most fine-grained nodes are leaf nodes, but a semantic hierarchy path can also

Table 2. Coarse and Hierarchical accuracies of the Branching Lever model and the CNN-RNN model on the test sets.

Model	Test Set	Coarse Accuracy	Hierarchical Accuracy
Test Set A	Branching Lever	65.08%	**48.05%**
	CNN-RNN	61.08%	40.35%
Test Set B	Branching Lever	29.15%	14.25%
	CNN-RNN	53.48%	46.43%
Test Set C	Branching Lever	32.5%	13.75%
	CNN-RNN	30.15%	**18.05%**
Overall	Branching Lever	44.19%	27.69%
	CNN-RNN	51.85%	**38.32%**

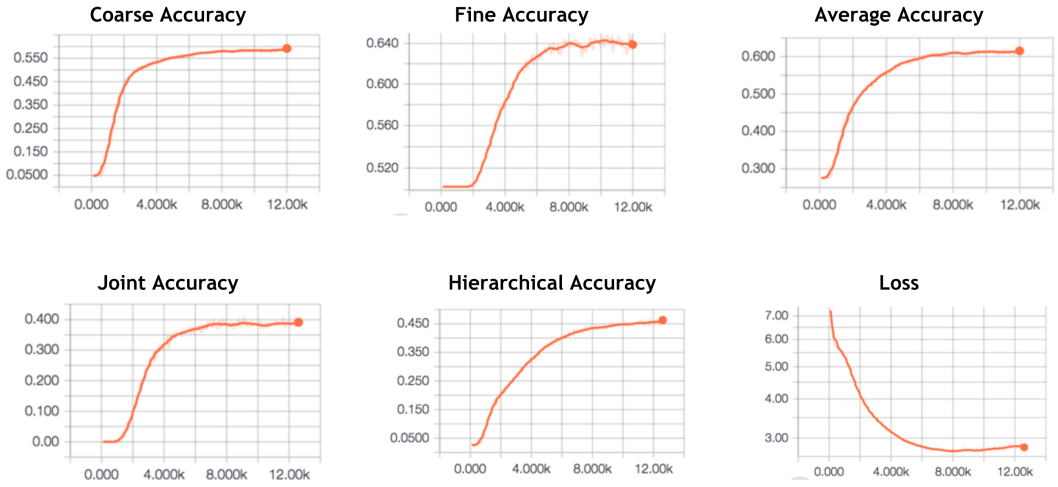


Figure 10. CNN-RNN end to end results

Table 3. CNN-RNN End-To-End Results

Metric	Accuracy
Coarse	58.15%
Fine	64.28%
Average	61.21%
Joint	39.29%
Hierarchical	45.43%

end at node on the path between the root and the most fine-grained node.

The generalized task would then be defined as follows:

> Given an image x, predict its semantic hierarchy path to the most fine-grained level the model is confident about.

In our experiments, the CNN-RNN model predicted only over two "timesteps", one for the coarse-grained prediction on a CIFAR-100 image, and one for the fine-grained prediction. Since the architecture of RNNs allows an RNN cell to be unrolled any number of times, we can use a dynamic length RNN to predict semantic hierarchy paths of any length, similar to how caption-generating CNN-RNNs can create variable length captions. To "end" the path, the RNN predicts the **END** token. We hypothesize that the CIFAR-100 dataset with only two hierarchy levels might not be able to showcase the full potential of the CNN-RNN model, and thus hope to experiment with the model on more complex semantic hierarchies, e.g. using the ImageNet dataset [2] or the CompCars dataset [6]. The CompCars dataset contains 163 car makes with 1,716 car models, perfect for establishing a nice semantic hierarchy. For example, we could establish a hierarchy as follows: 1) car type 2) car make 3) car model 4) model year. We are especially interested in observing how the attention mechanism ties in with this dataset; we would be interested to observe which features of the input image the model attends to as it iterates through the above described hierarchy.

9. Conclusion

We proposed a novel image classification task inspired by human learning behavior to predict semantic hierarchies with the additional objective to encourage finer-grained predictions, while avoiding those the model is not confident about. We believe that this is an important task since real-world objects can be classified at different levels of the semantic hierarchy. In addition, a model might not have trained on all the fine-grained classes, but should still be

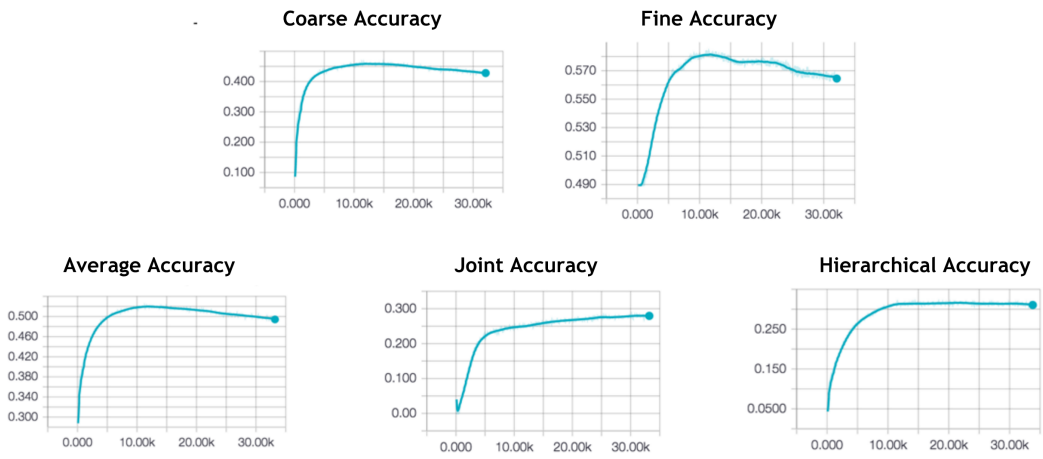


Figure 11. CNN-RNN Pre-Featurized Results

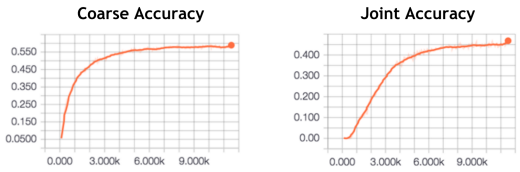


Figure 12. Results of weighted loss exploration.

able to classify an object from an unseen fine-grained class at a coarser level, similar to what humans would do. This would allow a model to be more flexible and generalizable. We also designed two network architectures for this task and demonstrated that the more advanced CNN-RNN model is able to produce better hierarchical coarse-grained predictions than the simpler branching lever model on images from fine-grained classes which the model has never trained on before. Furthermore, the CNN-RNN architecture is generalizable to semantic hierarchy paths of any lengths, an area worth of further exploration.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[3] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[4] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[5] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[6] X. Liu, W. Liu, H. Ma, and H. Fu. Large-scale vehicle reidentification in urban surveillance videos. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[7] M. Marszalek and C. Schmid. Semantic hierarchies for visual object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007.

[8] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

[9] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2740–2748, 2015.

10. Appendix

10.1. Example Predictions by the CNN-RNN Model

	Predicted Label	True Label
coarse	small_mammals	large_natural_outdoor_scenes
fine	<END>	forest

	Predicted Label	True Label
coarse	large_omnivores_and_herbivores	fruit_and_vegetables
fine	<END>	mushroom

	Predicted Label	True Label
coarse	medium_mammals	large_omnivores_and_herbivores
fine	<END>	camel

	Predicted Label	True Label
coarse	flowers	fruit_and_vegetables
fine	<END>	apple

Figure 13. Maintains hierarchy in predictions, as in the fine prediction exists, the fine prediction is subclass of coarse most of the time

	Predicted Label	True Label
coarse	fish	fish
fine	<END>	flatfish

	Predicted Label	True Label
coarse	household_electrical_devices	household_electrical_devices
fine	<END>	clock

	Predicted Label	True Label
coarse	vehicles_1	vehicles_1
fine	<END>	bus

Figure 14. Here we have cases where coarse is correct and no fine label predicted, which might be a sign that the model is not confident enough, but is at least getting partial credit.

	Predicted Label	True Label
coarse	aquatic_mammals	vehicles_2
fine	dolphin	rocket

	Predicted Label	True Label
coarse	large_natural_outdoor_scenes	reptiles
fine	cloud	crocodile

	Predicted Label	True Label
coarse	large_omnivores_and_herbivores	large_man-made_outdoor_things
fine	camel	castle

Figure 15. Fine prediction made: where model is wrong, wrong for reasonable reasons. Rarely is it unreasonably wrong when it makes a fine prediction. Also, we see that it retains hierarchy predictions.

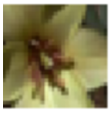
	Predicted Label	True Label
coarse	people	flowers
fine	<END>	<END>

	Predicted Label	True Label
coarse	aquatic_mammals	fish
fine	<END>	<END>

	Predicted Label	True Label
coarse	household_electrical_devices	reptiles
fine	<END>	<END>

Figure 16. Most predicted are just coarse, which is good. Incorrect predictions are reasonable. No clear patterns. Though retains hierarchy as above.

	Predicted Label	True Label
coarse	large_natural_outdoor_scenes	large_natural_outdoor_scenes
fine	<END>	<END>



	Predicted Label	True Label
coarse	non-insect_invertebrates	flowers
fine	bee	<END>

	Predicted Label	True Label
coarse	fish	large_natural_outdoor_scenes
fine	<END>	<END>

Figure 17. Examples from Test Set C. There are no clear patterns.

10.2. Example Predictions by the Branching Lever Model

	Predicted Label	Model Choice	True Label
coarse	large_omnivores_and_herbivores		large_omnivores_and_herbivores
fine	camel	X	camel

	Predicted Label	Model Choice	True Label
coarse	vehicles_2		vehicles_2
fine	rocket	X	rocket

	Predicted Label	Model Choice	True Label
coarse	medium_mammals		medium_mammals
fine	porcupine	X	porcupine

	Predicted Label	Model Choice	True Label
coarse	large_natural_outdoor_scenes		large_natural_outdoor_scenes
fine	cloud	X	cloud

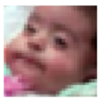
Figure 18. Branching Lever Model predictions on test subset A, where both the coarse and the fine predictions are correct, and the model correctly chose to predict the fine-grained label.

	Predicted Label	Model Choice	True Label
coarse	insects	X	insects
fine	bee		bee

	Predicted Label	Model Choice	True Label
coarse	large_natural_outdoor_scenes	X	large_natural_outdoor_scenes
fine	forest		forest

Figure 19. Branching Lever Model predictions on test subset A, where both the coarse and the fine predictions are correct, but the model chose to predict the coarse label instead of the fine label.

	Predicted Label	Model Choice	True Label
coarse	trees	X	trees
fine	maple_tree		oak_tree



	Predicted Label	Model Choice	True Label
coarse	people	X	people
fine	boy		baby

Figure 20. Branching Lever Model predictions on test subset A, where the the coarse prediction was correct and the fine prediction was wrong, and the model chose to predict coarse. Since fine was wrong, it is good that the model decided to predict the correct coarse label.

	Predicted Label	Model Choice	True Label
coarse	large_natural_outdoor_scenes	X	fish
fine	cloud		flatfish

	Predicted Label	Model Choice	True Label
coarse	large_carnivores	X	large_omnivores_and_herbivores
fine	cattle		camel

Figure 21. Branching Lever Model predictions on test subset A, where both the coarse and the fine predictions are wrong. Interestingly, the model chose the predict coarse rather than fine in most of these cases.

	Predicted Label	Model Choice	True Label
coarse	large_carnivores	X	large_carnivores
fine	leopard		lion

	Predicted Label	Model Choice	True Label
coarse	food_containers	X	food_containers
fine	bowl		can

Figure 22. Branching Lever Model predictions on test subset B, where the coarse prediction was correct and the model chose to predict the coarse label. Note that since the model never trained on the fine labels of subset B, the fine label is expected to always be wrong. Hence, this is an ideal outcome on test subset B.

	Predicted Label	Model Choice	True Label
coarse	flowers		flowers
fine	poppy	X	sunflower

	Predicted Label	Model Choice	True Label
coarse	fruit_and_vegetables		fruit_and_vegetables
fine	apple	X	pear

Figure 23. Branching Lever Model predictions on test subset B, where the coarse prediction was correct, but the model chose to predict the incorrect fine label. This is not an ideal outcome, since predicting coarse would have led to a correct prediction.

	Predicted Label	Model Choice	True Label
coarse	large_man-made_outdoor_things	X	large_natural_outdoor_scenes
fine	keyboard		mountain

	Predicted Label	Model Choice	True Label
coarse	aquatic_mammals	X	fish
fine	dolphin		ray

Figure 24. Branching Lever Model predictions on test subset B, where both the coarse and the fine predictions where wrong. Again, as we saw in subset A, the model chose to predict fine in most examples of this case.

	Predicted Label	Model Choice	True Label
coarse	vehicles_1	X	vehicles_1
fine	bear		train

	Predicted Label	Model Choice	True Label
coarse	large_natural_outdoor_scenes	X	large_natural_outdoor_scenes
fine	dolphin		sea

Figure 25. Branching Lever Model predictions on test subset C, where the coarse prediction was correct and the model chose to predict the coarse label. Note that for the Branching Lever Model, there is no difference between test subset B and C, since the network part of the model has not trained on any images belonging to the fine classes in subsets B or C.

	Predicted Label	Model Choice	True Label
coarse	fish		fish
fine	mushroom	X	trout

	Predicted Label	Model Choice	True Label
coarse	large_man-made_outdoor_things		large_man-made_outdoor_things
fine	castle	X	skyscraper

Figure 26. Branching Lever Model predictions on test subset C, where the coarse prediction was correct, but the model chose to predict the incorrect fine label. This is not an ideal outcome, since predicting coarse would have led to a correct prediction.

	Predicted Label	Model Choice	True Label
coarse	fish	X	reptiles
fine	dinosaur		turtle

	Predicted Label	Model Choice	True Label
coarse	non-insect_invertebrates	X	insects
fine	crab		cockroach

Figure 27. Branching Lever Model predictions on test subset C, where both the coarse and the fine predictions where wrong. Again, as we saw in subsets A and B, the model chose to predict fine, which at least shows that the model is not confident about its fine prediction.