A Hierarchical Multi-output Deep Neural Network for Image Classification

Edwin S Yan Edwin.Yan@jhu.edu
Whiting School of Engineering
Johns Hopkins University
Baltimore, MD 21218

Abstract

A hierarchical data structure is ubiquitous in many real-world applications to organize and memorize information. Hierarchical classification is a particular type of classification problem to predict multiple class labels within a hierarchical tree. It has comprehensive use cases in image classification tasks. Simultaneously, the hierarchical classification problem is also very challenging to ensure hierarchical consistency among different labels. Historically, these hierarchical structures have widely been ignored. The model will either only predict the most granular level then recover the parent classes, or simply predict each class individually. Neither approach is ideal due to the trade-off between accuracy and hierarchical consistency. This paper proposes a novel Hierarchical Multi-output Deep Neural Network (HM-DNN) architecture to uncover hierarchical structure by concatenating embedding space between parent class node and child class node. The hierarchical structure is discovered in the fully connected layers so that HM-DNN can be used with any popular Deep Neural Network architectures for image classifications, such as VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2015). I implement HM-DNN on CIFAR-10 (Krizhevsky, 2012), CIFAR-100, and CAR-196 (Krause et al., 2013) datasets to compare with the traditional implementation of the multi-output DNN. The experimental results indicate that HM-DNN has a significant improvement in hierarchical consistency and a slight improvement in overall accuracy compared to the baseline approach.

1. Introduction

A hierarchical data structure is quite common in our daily life. As human beings, we classify almost everything hierarchically. We do so to make learning about and finding information more accessible. For example, in biology, animals are classified into two categories at their highest order - vertebrates and invertebrates. Vertebrates can be further categorized as warm-blooded or cold-blooded. Furthermore, warm-blooded animals can be further breakdown into mammals, birds, etc.

At the same time, hierarchical classifications can also be quite challenging. One of the challenges is to ensure hierarchical consistency across all predictions of different levels. For example, if an animal in the image is classified as a bird, it should also be classified as warm-blooded. Any mismatches between the predictions in different hierarchical levels will

introduce hierarchical inconsistency.

Another challenge is to share the information learned across different levels to improve the accuracy further. For example, the features learned from coarse-level classes might provide valuable information to the fine-grained level classes to improve their prediction accuracy. One common approach is to predict the most granular level and rebuild the hierarchical structure from the bottom up. This approach may lose many details from the coarse levels, resulting in further unsatisfactory performance, especially when there are a large number of classes in the lowest granular level.

Most of the existing approaches have several drawbacks. They either ignore the hierarchical structure completely or suffer hierarchical inconsistency across the predictions. The detailed pros and cons of existing approaches will be discussed in Section 2.

This paper proposes a novel **H**ierarchical **M**ulti-output **D**eep **N**eural **N**etwork (**HM-DNN**) architecture to uncover hierarchical structure by concatenating embedding space between parent class node and child class node. Although the same architecture may work in many different classification problems, such as image classification and text classification, this paper will only focus on hierarchical image classifications. Compared to existing approaches, this paper has the following contributions:

1. Proposing HM-DNN for hierarchical image classification problems. A single DNN model can train all of the attributes regardless of whether they are part of the hierarchical tree.
2. By concatenating the dense layers of different hierarchical levels, HM-DNN can uncover the underlying hierarchical structure through the training process, resulting in better hierarchical consistency across different output levels
3. HM-DNN can be used with any existing DNN structure to solve different hierarchical image classification problems.

The remainder of this paper is organized as follows. In Section 2, I introduce the related work. Section 3 discusses my hypothesis and research objectives. Section 4 presents the proposed HM-DNN architecture. Then, in Section 5, the performance of the proposed algorithm is analyzed. Finally, Section 6 concludes this paper and discusses the future work.

2. Related work

There are several common approaches to solve hierarchical classification problems. The most common approach is to classify the labels at the most granular level regardless of hierarchical structure. This approach is also described as the flat approach in (Ghazi et al., 2010) or multi-class approach in (Pereira et al., 2020). The parent class labels are reconstructed based on a predefined hierarchy and the predictions on leaf nodes. The flat approach is illustrated in Figure 1.

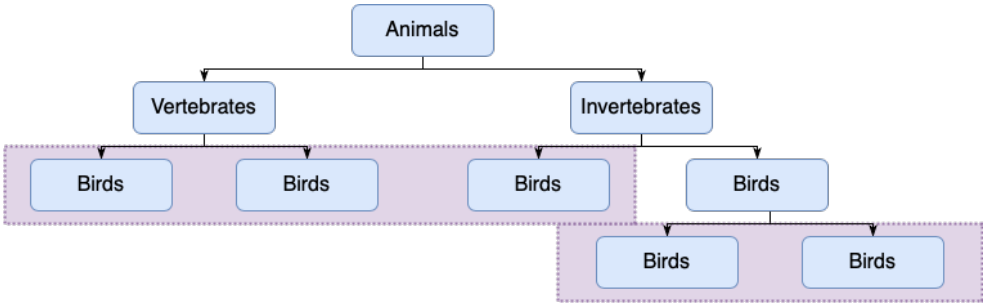

Figure 1: Flat approach classifies the leaf nodes regardless of hierarchical structure, then reconstructs the parent classes

The most significant benefit of using the flat approach is its simplicity. Any classification algorithm can be applied before reconstructing the parent class labels. The hierarchical consistency is also guaranteed by using this approach because parent classes are reconstructed based on a predefined hierarchy. However, on the other hand, it may perform poorly when there are a large number of classes in the leaf nodes because the information and similarities shared between different hierarchical levels are not considered by the model.

Because of its simplicity, it is the most widely used approach to tackle hierarchical classification problems. (Zimek et al., 2010) also claimed that the flat approach could perform equally well compared to other hierarchical approaches in protein classifications. A similar conclusion is also observed in (Pereira et al., 2020) to classify X-ray images related to Covid-19.

Another common approach is to create a local classifier for each node. The classifiers are normally built from a top-down fashion. Then, it ensembles all of the classifiers using a decision tree. Several research papers also refer to this approach as a top-down approach. Since local classifiers are created for each node, this approach also guarantees consistency across different levels. Figure 2 illustrates the local classifier for each node approach.

There are various research domains that implemented this approach. For example, (Burred and Lerch, 2003) used a local classifier for each node to classify musical genre. (Nakano et al., 2017) used the same approach to classify DNA sequences. There is also a notable amount of research to compare its performance to the flat approach in several different domains, such as (Zimek et al., 2010), (Ghazi et al., 2010) and (Pereira et al., 2020). Based on their results, the hierarchical approach generally performs slightly better than the flat approach.

However, this approach is also commonly criticized for its two most significant disadvantages. The biggest disadvantage is that its accuracy degrades from the top level to the bottom level because the accuracy of the child levels depends on the accuracy of their parent level prediction. Any incorrect predictions on the parent level will result in inaccurate

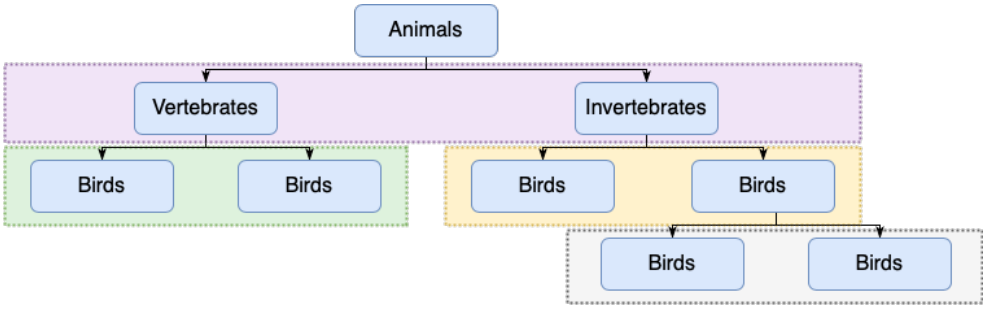

Figure 2: Local classifiers per node are created for each branch to classify all of its child nodes

predictions on all subsequent child levels. Besides, every node in the hierarchical tree will be trained as a separate classifier. Hence, the number of models in the ensemble may become unmanageable when there are many branches.

Last but not least, the other commonly used approach is called "local classifier per level". Different classifiers are created for each level, regardless of the hierarchical structure. Normally, a multi-output classifier is used to predict multiple levels at the same time. Figure 3 illustrates the local classifier for each level approach.

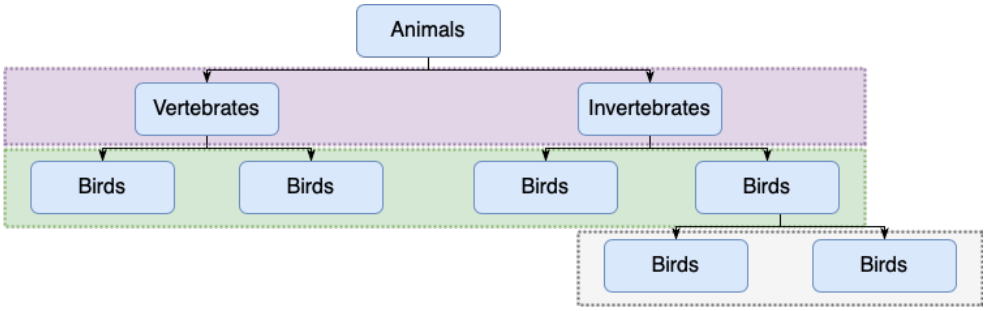

Figure 3: Local classifiers per level are created to classify different attributes

There are several benefits of using this approach. First of all, it aligns with the data structure because the original hierarchical classification problem can simply be turned into multiple simple classification problems using the input data and each attribute. Therefore, it is common to utilize a single multi-output classifier to predict multiple levels simultaneously. This approach also enables the possibility to predict other attributes that are not part of the hierarchical tree via the same model. For example, to classify a car image, its color can be predicted along with the brand and the model via the same model. Moreover, the accuracy of each level is independent, which means poor predictions on one level will not impact the predictions on other levels.

Simultaneously, this approach also undergoes extensive criticism on its hierarchical inconsistency of the predictions across different levels. This can be problematic in some scenarios when the class hierarchy needs to be strictly enforced. Several pieces of research tried to address this problem. For example, (Paes et al., 2012) recommended a voting mechanism based on their research to enforce consistency.

More recently, more and more researchers utilize multi-output DNN to improve the hierarchical consistency and accuracy of the local classifier per level approach. (Zhu and Bain, 2017) designed branch convolutions neural network (B-CNN) to predict multiple hierarchical attributes through the same model. Although hierarchical consistency is not guaranteed, the authors claimed the overall accuracy is slightly higher than the flat approach. A similar conclusion can be observed by another research performed by (Seo and shik Shin, 2019), which improved the prediction accuracy on fashion image classifications by creating branches for each level in the VGG16 and VGG19 model (Simonyan and Zisserman, 2014). There is also research performed by (Wu and Saito, 2017) that introduced a Hierarchical classification with a neural network (HiNet) model. The model is designed to perform transfer learning cross levels using a combined cost function during training and using a greedy downpour algorithm during inference. The authors proved that their model is significantly more computationally efficient than the DNN based on the flat approach.

3. Hypothesis and Research Objectives

Each of the existing approaches has its advantages and disadvantages. Since the local classifier per level approach requires a separate flat classifier for each attribute, it can be simply achieved via a single multi-output DNN model. Therefore, my research objectives come down to identifying a neural network architecture to bridge the output branches, so the features derived from hidden layers can be shared across different hierarchical levels to improve the hierarchical consistency.

Since the output from the convolutional layers represents the high-level features discovered from the input images, the dense layers, also known as fully connected layers, are typically used to discover the nonlinear relationships among these features. Therefore, I hypothesize that the network can uncover the hierarchical structure via the back-propagation step if the dense layer is shared between different hierarchical levels. Besides, I hypothesize that the same assumption should hold regardless of the overall accuracy.

To test the hypothesis, the experiments in Section 5 are designed to compare the **H**ierarchical **I**nconsistency HI and the classification **Err**or rate of each attribute Err_i of the proposed model against the standard implementation of local classifiers per level approach. The hierarchical inconsistency HI is defined as follows:

$$HI = \frac{1}{N} * \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{M_{ij}}{n} \tag{1}$$

where N is the total number of attributes, M_{ij} is the total number of predictions from i-th attribute and j-th attribute that are not in the same hierarchical tree, n is the total number of samples

the classification error rate of each attribute Err_i is defined as follows:

$$Err_i = f_i/n * 100 \tag{2}$$

where i is the i-th attribute, $f_i$is the total number of incorrect classifications in the i-th attribute, n is the total number of samples.

If there is no significant improvement on HI nor Err_i of each attribute, it disproves the hypothesis. Otherwise, the hypothesis will be accepted.

4. Proposed Approach

The DNN needs to uncover the hierarchical structure from the input data to lower the hierarchical inconsistency. Therefore, my research has focused on how to detect the hierarchical structure via the back-propagation step.

In the traditional implementation of the multi-output DNN, each branch will have multiple **F**ully **C**onnected (**FC**) layers before the softmax layer to learn the relations between the active features extracted from the base **C**onvolution **N**eural **N**etwork (**CNN**). See Figure 4. However, since FC layers among different branches are independent of each other, they will mainly learn the important features specifically for the branch. The only layers that will learn the hierarchical structure are the layers shared by multiple branches because they will update the weights based on the loss from multiple branches. If the shared layers are very far from the output layers, most of the information may be lost during the back-propagation stage, resulting in larger hierarchical inconsistency.

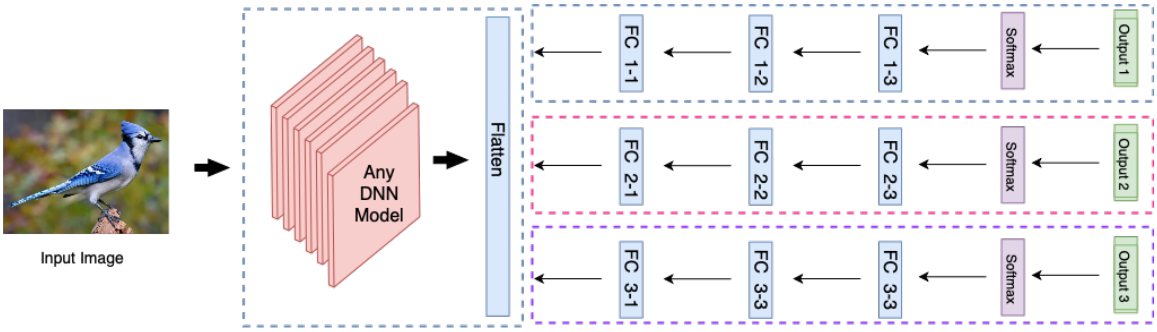

Figure 4: Traditional implementation of multi-output DNN. It is hard to uncover the hierarchical structure during the back-propagation stage because there are too many layers between output layers and the layer shared by multiple branches.

To ensure the DNN can capture the hierarchical structure at an early stage of the back-propagation process, HM-DNN concatenates the last FC layers between the superclass and subclass branches before the output and softmax layers.

My initial implementation of the HM-DNN uses the same concatenated layer as the input to the output and softmax layers of both parent and child classes. However, based on the experiments, it works better if the concatenated layer is only used in the child branch while keeping the parent class branch structure unchanged. This is probably because the superclass branch features are essential to the subclass branch, but not the other way around. Figure 5 illustrates the final implementation of HM-DNN when the child branch only depends on one parent branch. If multiple child branches are under each parent branch, the FC layer from the parent branch should be concatenated to the FC layers in each child branch. Similarly, if the child branch depends on multiple parent branches, then the FC layers from all parent branches should be concatenated to the FC layer in the child branch.

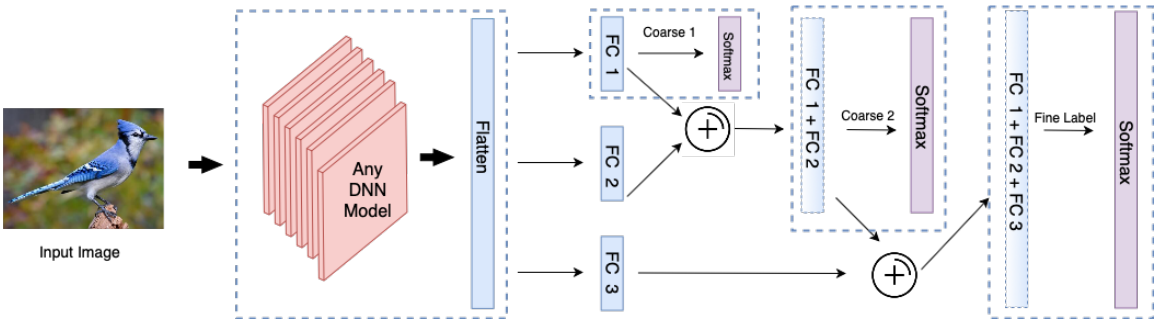

Figure 5: Architecture of HM-DNN, where FC layers are shared between the parent and child classes.

HM-DNN can use any type of loss function designed for classification problems. For the implementation in this paper, categorical cross-entropy is used. Since the concatenated layers will take the loss from multiple branches, a hyperparameter W_i is introduced to define the weight of the losses from each branch. Therefore, the total loss function $\mathcal{L}$ of the HM-DNN is defined as:

$$\mathcal{L} = \sum_{i=1}^{N} -W_i * \sum_{j=1}^{n} y_i * log\widehat{y}_i$$

where N is the total number of attributes, n is the total number of samples, y_i is the actual label of i-th sample, $\widehat{y}_i$ is the predicted label of i-th sample.

5. Experiments and Analysis

In this section, I study the performance of HM-DNN by making empirical comparisons against the standard implementation of the local classifier per node model on several bench-

mark datasets. HM-DNN is implemented the same way as Figure 5 illustrates. As a comparison, the base model is implemented as multi-output DNN as illustrate in Figure 6. Since there are no connections among different branches, it works the same way as several separate DNN classifiers with the same input data. The DNN architecture in both HM-DNN and base model is selected based on different datasets. All of the experiments are performed based on stratified five-fold cross-validation[1] to ensure the reliability of the outcome.

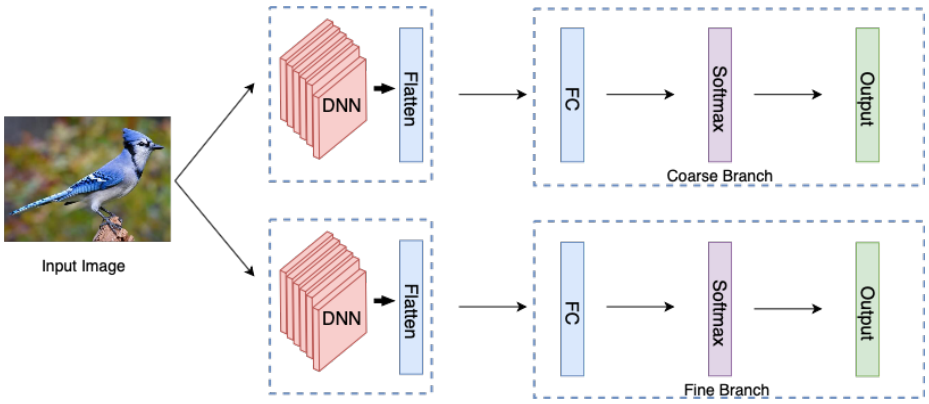

Figure 6: Baseline model uses separate DNNs and fully connected layers in each branch

5.1 Proposed Metrics

All comparisons in my experiments use Hierarchical Inconsistency (1) and Error Rates (2) to measure the performance of HM-DNN compared with other approaches. All experiments[2] are implemented using Tensorflow 2.0. I perform all of the model training on an Nvidia GTX 1080 Ti GPU. The **A**verage **R**unning **T**ime **P**er **E**poch $ARTPE$ is provided as a reference. The $ARTPE$ includes both training and validation steps. It also includes the image pre-processing step if data augmentation is used.

5.2 Test Datasets

The following datasets have been used. See Appendix A for sample inputs with corresponding labels.

1. **CIFAR-10 (Krizhevsky, 2012):** It consists of 50,000 images of the training set and 10,000 images of the test set. Each color image is sized 32 x 32 pixels and classified into ten classes. These ten classes are further grouped into two super-classes. The dataset is relatively small ($\sim$ 162MB). The hierarchical inconsistency is relatively low with the standard implementation of the local classifier per node approach given the small

1. Stratified five-fold cross-validation is based on the lowest granular level, so all of the parent classes will also get an equal amount of observations in both train and test splits.
2. Source code is available at `https://github.com/edwin-yan/hierarchical_classification`

Algorithms	Superclass Err	Subclass Err	HI	$ARTPE$
Baseline	6.22%	**23.71%**	5.72%	11s 148ms
HM-DNN	**4.60%**	24.93%	**0.59%**	**6s 82ms**

Table 1: Performance comparison between baseline algorithm and HM-DNN on CIFAR-10 dataset

number of classes. It is perfect to evaluate whether HM-DNN can further improve the hierarchical inconsistency.

2. **CIFAR-100 (Krizhevsky, 2012):** It is very similar to the CIFAR-10 dataset, except it has 100 classes containing 600 images each. The dataset is relatively small ($\sim$ 161MB). In the original dataset, the 100 classes are grouped into 20 superclasses. The 20 superclasses are further grouped into seven ultra-classes. The state-of-art image classification algorithms cannot reach high accuracy without data augmentation, so it is a perfect case to experiment whether HM-DNN can still maintain minimal hierarchical inconsistency even with fair overall accuracy.

3. **Cars-196 (Krause et al., 2013):** It contains 16,185 images with 196 classes with a combination of year, brand, model, and car type. The original class labels are decomposed into nine superclasses for car types, 48 superclasses of car brands, and 173 subclasses of car models. The dataset is relatively large ($\sim$ 1.8GB). Both the image sizes and the label hierarchy are very close to real-world problems. This is a perfect example to validate the performance of HM-DNN.

5.3 Study the Overall Performance and the Importance of Loss Weights

In this experiment, both HM-DNN and baseline models use VGG11 as their base CNN model. The VGG11 model uses the same configuration as the original paper ($Cov3-64 \rightarrow Maxpool \rightarrow Cov3-128 \rightarrow Maxpool \rightarrow Cov3-256 \rightarrow Cov3-256 \rightarrow Maxpool \rightarrow Cov3-512 \rightarrow Cov3-512 \rightarrow Maxpool \rightarrow Cov3-512 \rightarrow Cov3-512 \rightarrow Maxpool$). Both models are trained on CIFAR-10 datasets with five-fold cross-validation. Adam is used as the optimizer of both models, with an initial learning rate of 1e$-$4 and decay of 1e$-$6. The batch size is set to 512. For HM-DNN, the loss weights W_i are set to 0.4 for the superclass branch and 0.6 for the subclass branch.

The performance of base models and HM-DNN are shown in Table 1. HM-DNN demonstrates 9.69$\times$ better hierarchical consistency than the base model. It also further reduces the superclass error rate to 4.6% compared to 6.22% in the base model, although it performs slightly worse in the subclass error rate. It is also worth pointing out that HM-DNN takes 1.8x less time per epoche than the base model because all branches share the same CNN model.

The subclass loss weights control the amount of loss from the subclass branch back-propagate back to the dense layer in the superclass branch. Therefore, it is worth studying the impact of different loss weights between superclass and subclass branches. The experiment results are illustrated in Figure 7. The results are slightly disappointing because it is impossible to eliminate hierarchical inconsistency with any combinations of the loss weights. On the other hand, it also confirmed that the features learned from the superclass branch do have a positive impact on the subclass branch. Based on the results, setting superclass weights to 0.4 and subclass weights to 0.6 can yield the best outcome.

Figure 7: Impact of Loss Weights on Hierarchical Consistency, Superclass Error Rate and Subclass Error Rate

5.4 Stress Test HM-DNN on Hierarchical Inconsistency

To better understand the performance of HM-DNN on datasets with a large number of classes, I perform the same experiments on the CIFAR-100 dataset. In this experiment, both HM-DNN and baseline models use VGG13 as their base CNN models. The configuration of VGG13 is $Cov3-64 \rightarrow Cov3-64 \rightarrow Maxpool \rightarrow Cov3-128 \rightarrow Cov3-128 \rightarrow Maxpool \rightarrow Cov3-256 \rightarrow Cov3-256 \rightarrow Maxpool \rightarrow Cov3-512 \rightarrow Cov3-512 \rightarrow Maxpool \rightarrow Cov3-512 \rightarrow Cov3-512 \rightarrow Maxpool$. The image augmentation is not applied to the training images, so this experiment can stress test the hierarchical inconsistency of the HM-DNN model when it cannot achieve high overall accuracy[3]. Similar to the prior experiment, Adam is used as the optimizer of both models, with an initial learning rate of 5e$-$4 and decay of 1e$-$6. The batch size is set to 256. For HM-DNN, the loss weights W_i for ultraclass, superclass, and subclass branches are 0.2, 0.35, and 0.45, respectively.

3. Most of the papers use Top K accuracy on the CIFAR-100 dataset. The error rates in this experiment are still based on all classes to better align with the calculation on hierarchical inconsistency

Algorithms	Ultraclass Err	Superclass Err	Subclass Err	HI	$ARTPE$
Baseline	26.03%	33.66%	46.31%	29.57%	29s 194ms
HM-DNN	**21.74%**	**31.35%**	**44.82%**	**6.02%**	**10s 65ms**

Table 2: Performance comparison between baseline algorithm and HM-DNN on CIFAR-100 dataset

Table 2 summarizes the performance of both models. Without data augmentation, VGG13 has some challenges to maintain high overall accuracy on the CIFAR-100 dataset. However, despite the relatively high subclass error rate, HM-DNN can still ensure good hierarchical consistency with slightly better accuracy in all branches than the baseline model. Lastly, HM-DNN takes significantly less time than the baseline model for each epoch.

5.5 Demonstrate the Potential on Real World Applications

To further explore the potential of using HM-DNN in real-world applications, I perform the experiments on the CAR-196 dataset. Instead of using the original labels, I split them into three class labels, Brand, Model, and Car Type. The model depends on Brand, and Car Type depends on the combination of Brand and Model.

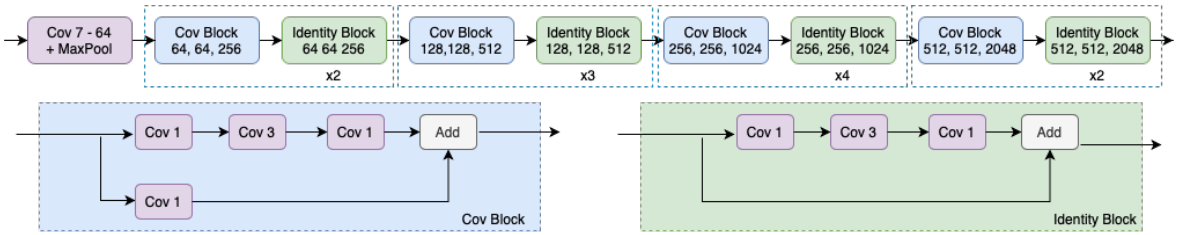

Figure 8: Demonstration of the ResNet-50 architecture use in the experiment

This experiment explores whether HM-DNN can further improve the hierarchical inconsistency when the baseline model can achieve fairly good accuracy. The base DNN model used in both baseline and HM-DNN models is ResNet-50 (He et al., 2015). Figure 8 illustrates the configuration of ResNet-50 implemented in the experiment. Since the original input sizes vary from image to image, all images are sized to 256 x 256 before the training step. Image augmentation is also used as a preprocessing step to improve the overall accuracy further. The image augmentation includes random rotations within 20 degrees, random horizontal flips, random shifts of the original images by 10% in all directions. The image preprocessing is performed on AMD Threadripper 2970WX CPU (@4.0 GHz x 48 threads) in parallel with training on GPU, which has minimal impact on overall runtime. Due to the VRAM size limit on the GPU, the batch size is set to 32. For the same reason, the baseline model is split into three separate identical models for each class. The overall

Algorithms	Ultraclass Err	Superclass Err	Subclass Err	HI	$ARTPE$
Baseline	7.09%	9.30%	14.77%	9.30%	281s 772ms
HM-DNN	**6.59%**	**7.25%**	**12.53%**	**2.97%**	**154s 429ms**

Table 3: Performance comparison between baseline algorithm and HM-DNN on Car-196 dataset

RTPE of the baseline model is the total RTPE of three individual models. Similar to the previous experiments, the optimizer is Adam with an initial learning rate of 1e−4 and decay of 1e−6. For HM-DNN, the loss weights W_i for Car Type, Brand, and Model branches are 0.2, 0.35, and 0.45, respectively.

The results in Table 3 are promising. The hierarchical inconsistency improves more than three times compared to the baseline model. The error rates of each class label also dropped slightly in the HM-DNN model, which indicates that the model learns better when the learned features are shared across different branches. The observations from the results highlight the potential to use HM-DNN to further improve the hierarchical classification models in real-world applications.

6. Conclusion and Future Work

In this paper, I propose a novel approach to solve hierarchical image classification problems via multi-output DNN. Through concatenating the FC layer from the superclass branch to the subclass branch, it enables the proposed model to discover the hierarchical relationships among different class labels, which significantly improves the hierarchical consistency. As evidenced by my experiments, HM-DNN can maintain decent hierarchical consistency regardless of the overall accuracy of each class branch. Furthermore, the experiment results also illustrate that HM-DNN has a slight improvement in overall accuracy than the baseline model. This further proves the benefit of sharing the embedding space in the FC layer from an upper-level branch with a lower-level branch.

This research also reveals two new areas for future researches. HM-DNN works very well in hierarchical image classification problems. Due to time constraints, the same architecture is experimented on in other areas, such as hierarchical text classification. It would be worth experimenting whether it can also achieve similar improvements in those areas.

Lastly, HM-DNN maintains good hierarchical consistency by sharing embedding spaces between parent and child branches. In future research, I would also like to explore the architecture on the output layer where the output nodes can inhibit the other ones that are not in the same hierarchical tree to improve the hierarchical inconsistency further.

Appendix A. Sample Images

Figure 9: Example inputs and corresponding hierarchical labels from CIFAR-10 dataset

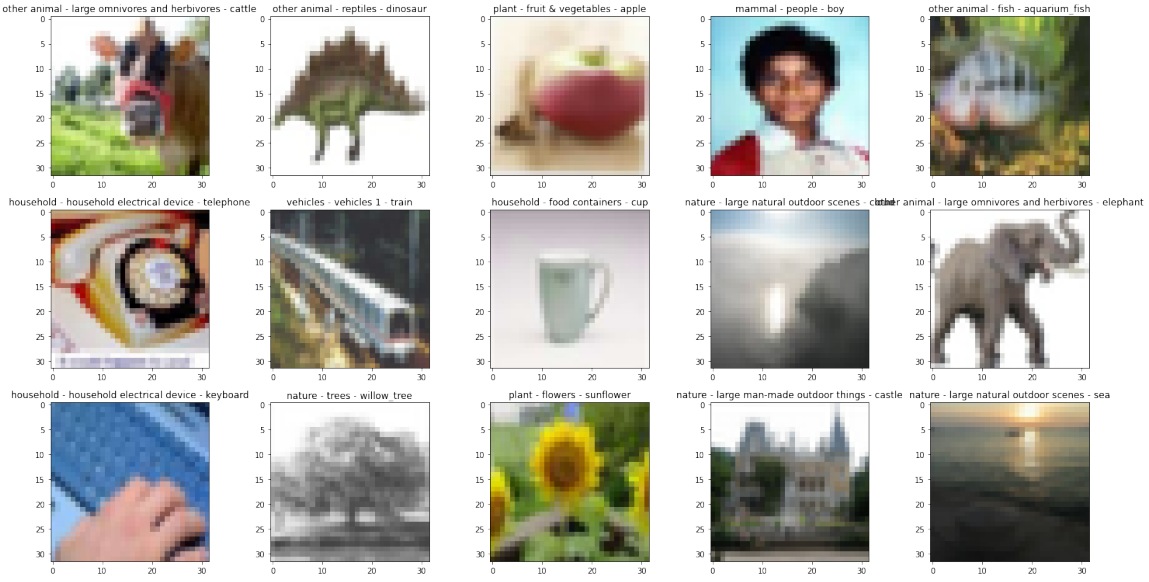

Figure 10: Example inputs and corresponding hierarchical labels from CIFAR-100 dataset

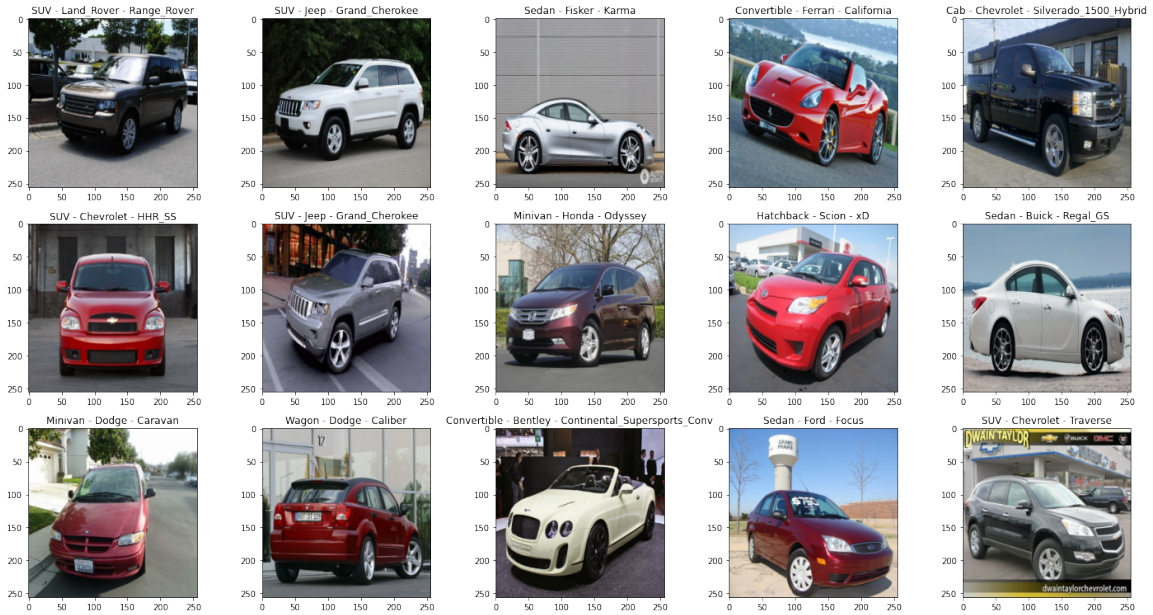

Figure 11: Example inputs and corresponding hierarchical labels from CAR-196 dataset

References

Juan José Burred and Alexander Lerch. A hierarchical approach to automatic musical genre classification. In *in Proc. Of the 6 th Int. Conf. on Digital Audio Effects (DAFx*, pages 8–11, 2003.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. Hierarchical versus flat classification of emotions in text. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, page 140–146, USA, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*, ICCVW '13, page 554–561, USA, 2013.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

F. K. Nakano, W. J. Pinto, G. L. Pappa, and R. Cerri. Top-down strategies for hierarchical classification of transposable elements with neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2539–2546, 2017.

Bruno C. Paes, A. Plastino, and A. Freitas. Improving local per level hierarchical classification. *J. Inf. Data Manag.*, 3:394–409, 2012.

Rodolfo M. Pereira, Diego Bertolini, Lucas O. Teixeira, Carlos N. Silla, and Yandre M.G. Costa. Covid-19 identification in chest x-ray images on flat and hierarchical classification scenarios. *Computer Methods and Programs in Biomedicine*, 194:105532, 2020.

Yian Seo and Kyung shik Shin. Hierarchical convolutional neural networks for fashion image classification. *Expert Systems with Applications*, 116:328–339, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

Z. Wu and Sean Saito. Hinet: Hierarchical classification with neural network. *ArXiv*, abs/1705.11105, 05 2017.

Xinqi Zhu and Michael Bain. B-cnn: Branch convolutional neural network for hierarchical classification. *ArXiv*, abs/1709.09890, 09 2017.

Arthur Zimek, Fabian Buchwald, Eibe Frank, and Stefan Kramer. A study of hierarchical and flat classification of proteins. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 7:563–71, 07 2010.