Combining Class Taxonomies and Multi Task Learning To Regularize Fine-grained Recognition

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS in Computer Science and Engineering by Research

by

Riddiman Dasgupta
201307558
riddhiman.dasgupta@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Combining Class Taxonomies and Multi Task Learning To Regularize Fine-grained Recognition" by Riddhiman Dasgupta, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. Anoop Namboodiri

To my parents.

Acknowledgments

First of all, I would like to thank my advisor Dr. Anoop Namboodiri immensely for his unwavering support and guidance throughout the duration of my research experience at IIIT Hyderabad. Research is not always easy, and marred with unknown obstacles, but even in the toughest of times, Dr. Namboodiri was always there to lend a ear, to encourage, to reassure. More than academic guidance and technical discussions, of which we definitely had more than our fill, I feel it is this other form of support that is much more critical and much more rarer, and hence, to be much more cherished. The faculty members, senior researchers, fellow batchmates, and juniors alike, at the Center for Visual Information Technology (CVIT) were instrumental in providing me the drive to work hard by setting very motivating examples. I have much gratitude to all my friends in the lab, and for brevity's sake I am not naming everyone who has helped make the journey memorable. However, extra acknowledgements go out to Aniket Singh, Koustav Ghosal, and Ameya Prabhu for all the fruitful technical discussions, that have helped me stand on the shoulders of giants. I would like to specially thank Satarupa Guha for being my unwavering lighthouse in the turmoil as well as the calm, Anirban Ghose for helping me to find my bearings, and Tanay Dutta for pushing me to go beyond my comfort zone. I cannot forget to thank my group of friends at IIIT – Koustav Mullick, Debarshi Dutta, Ayushi Dalmia, Sudipto Banerjee – for making graduate life so much easier, for making me miss home so much less, for giving me so many memories to cherish. I also must mention the role my parents have had to play - almost all of what I am is because of the hard word and efforts and sacrifices they have invested in me.

> The door is more than it appears. It separates who you are from who you can be. You do not have to walk through it... You can run!

Abstract

Fine-grained classification is an extremely challenging problem in computer vision, impaired by subtle differences in shape, pose, illumination and appearance, and further compounded by subtle intra-class differences and striking inter-class similarities. While convolutional neural networks have become versatile jack-of-all-trades tool in modern computer vision, approaches for fine-grained recognition still rely on localization of keypoints and parts to learn discriminative features for recognition. In order to achieve this, most approaches necessitate copious amounts of expensive manual annotations for bounding boxes and keypoints. As a result, most of the current methods inevitably end up becoming complex, multi-stage pipelines, with a deluge of tunable knobs, which makes it infeasible to reproduce them or deploy them for any practical scenario. Since image level annotation is prohibitively expensive for most fine-grained problems, we look at the problem from a rather different perspective, and try to reason about what might be the minimum amount of additional annotation that might be required to obtain an improvement in performance on the challenging task of fine-grained recognition.

In order to tackle this problem, we aim to leverage the (taxonomic and/or semantic) relationships present among fine-grained classes. The crux of our proposed approach lies in the notion that fine-grained recognition effectively deals with subordinate-level classification, and as such, subordinated classes imply the presence of inter-class and intra-class relationships. These relationships may be taxonomical, such as super-classes, and/or semantic, such as attributes or factors, and are easily obtainable in the sense that domain expertise is needed for each fine-grained label, not for each image separately. We propose to exploit the rich latent knowledge embedded in these inter-class relationships for visual recognition. We posit the problem as a multi-task learning problem where each different label obtained from inter-class relationships can be treated as a related yet different task for a comprehensive multi-task model. Additional tasks/labels, which might be super-classes or attributes, or factor-classes can act as regularizers, and increase the generalization capabilities of the network. Class relationships are almost always a free source of labels that can be used as auxiliary tasks to train a multi-task loss which is usually a weighted sum of the different individual losses. Multiple tasks will try to take the network in diverging directions, and the network must reach a common minimum by adapting and learning features common to all tasks in its shared layers.

Our main contribution is to utilize the taxonomic/semantic hierarchies among classes, where each level in the hierarchy is posed as a classification problem, and solved jointly using multi-task learning. We employ a cascaded multi-task network architecture, where the output of one task feeds into the next, thus

enabling transfer of knowledge from the easier tasks to the more difficult ones. To gauge the relative importance of tasks, and apply appropriate learning rates for each task to ensure that the related tasks aid and unrelated tasks does not hamper performance on the primary task, we propose a novel task-wise dynamic coefficient which controls its contribution to the global objective function.

We validate our proposed methods for improving fine-grained recognition via multi-task learning using class taxonomies on two datasets, viz. CIFAR 100, which has a simple 2 level hierarchy, albeit a bit noisy, which we use to estimate how robust our proposed approach is to hyperparameter sensitivities, and CUB-200-2011, which has a 4 level hierarchy, and is a more challenging real-world dataset in terms of image size, which we use to see how transferable our proposed approach is to pre-trained networks and fine-tuning. We perform ablation studies on CIFAR 100 to establish the usefulness of multi-task learning using hierarchical labels, and measure the sensitivity of our proposed architectures to different hyperparameters and design choices in an imperfect 2 level hierarchy. Further experiments on the popular, real-world, large-scale, fine-grained CUB-200-2011 dataset with a 4 level hierarchy re-affirm our claim that employing super-classes in an end-to-end model improves performance, compared to methods employing additional expensive annotations such as keypoints and bounding boxes and/or using multi-stage pipelines. We also prove the improved generalization capabilities of our multi-task models, by showing how multiple connected tasks act as regularizers, reducing the gap between training and testing errors. Additionally, we demonstrate how dynamically estimating auxiliary task relatedness and updating auxiliary task coefficients is more optimal than manual hyperparameter tuning for the same purpose.

Contents

Chapter Page

1 Introduction . . . 1
1.1 Fine Grained Recognition . . . 1
1.2 Hierarchy/Taxonomy/Ontology Based Recognition . . . 2
1.3 Neural Networks . . . 4
1.4 Convolutional Neural Networks . . . 5
1.5 Multi-Task Learning . . . 7
1.6 MTL as Regularization for Convnets . . . 8
1.7 Major contributions . . . 8
1.8 Outline of thesis . . . 9

2 Related Work . . . 10
2.1 Fine Grained Recognition . . . 10
2.1.1 Part Based Methods . . . 11
2.1.2 Ensemble Based Methods . . . 12
2.1.3 Attention Based Methods . . . 15
2.2 Deep Multi-task Learning . . . 16
2.3 Taxonomy Based Classification . . . 18
2.4 Dynamic Multi Task Coefficients . . . 19
2.5 Chapter Summary . . . 20

3 Proposed Approach . . . 21
3.1 Multiple Related Tasks . . . 21
3.2 Hierarchy as a Related Task . . . 23
3.3 Task-specific Coefficients . . . 23
3.4 Chapter Summary . . . 25

4 Experimental Results on CIFAR 100 . . . 27
4.1 Dataset details . . . 27
4.2 Architecture details . . . 28
4.3 Hyperparameter details . . . 30
4.4 Experimental results . . . 31
4.4.1 State-of-the-art Models . . . 31
4.4.2 Baseline Models . . . 32
4.4.3 Multi-Task Models . . . 32
4.4.4 Effect of Hierarchy . . . 34

4.5 Overall Analysis and Chapter Summary 38

5 Experimental Results on Caltech UCSD Birds 39
5.1 Dataset details 39
5.2 Architecture details 40
5.3 Hyperparameter details 41
5.4 Experimental results 43
5.4.1 State-of-the-art Models 43
5.4.2 Baseline Models 44
5.4.3 Multi-Task Models 45
5.4.4 Combining with State-of-the-art Methods 47
5.4.5 Multi-Task as Regularization 48
5.5 Overall Analysis and Chapter Summary 49

6 Conclusions 51

Appendix A: Full Parsed Taxonomy for Caltech-UCSD Birds-200-2011 Dataset 53

Bibliography 61

List of Figures

Figure Page

1.1 An example of the generic recognition problem in computer vision. Note that for many object categories, visually distinguishing them becomes simple because of a multitude of factors such as background, illumination, shape, etc. 1
1.2 Different examples for fine-grained recognition. It is to be noted how many of the challenging aspects of fine-grained recognition can be observed from even a random sampling of fine-grained images. Best viewed enlarged, in color. 2
1.3 Different examples for how relationships can be found among fine-grained classes. Figures taken from Xie *et al*. [71]. Best viewed in colour. 3
1.4 Leveraging the taxonomic ontology of birds for fine grained recognition. From top to bottom, we have family, order and species for five classes of kingfishers in the CUB-200-2011 dataset [68]. Observe how identifying the family or order can help identifying the class, e.g. in case of ringed kingfisher and green kingfisher. Best viewed enlarged, in color. 4
1.5 On the left, we have a model of a single neuron, while on the right, a simple multilayer perceptron is shown. 5
1.6 A simple convolutional neural network, with convolutional, pooling and fully-connected layers leading to a softmax based classification output. Dashed boxes show the filter size at each layer, i.e. the patch on which that layer operates in a sliding window fashion. . 6
1.7 Block diagram representation of two standard deep convolutional neural networks for large-scale image classification, viz. Alexnet [39] and VGGNet [59]. Deeper models such as GoogleNet [64] and ResNets [26] are not shown here due to lack of space. Interested readers can visit this link. 6
1.8 An example of a multi-task network. Figure taken from [23]. Best viewed enlarged, in color. 7

2.1 Illustration of pipeline for Part based R-CNN [77]. Best viewed digitally in color. Figure taken from original paper. 10
2.2 Illustration of pipeline for pose normalized deep convnet [5]. Best viewed digitally in color. Figure taken from original paper. 11
2.3 Illustration of pipeline for part-stacked CNN from Huang *et al*. [30]. Best viewed digitally in color. Figure taken from original paper. 12
2.4 Illustration of pipeline for end-to-end model from Zhang *et al*. [78]. Best viewed digitally in color. Figure taken from original paper. 13
2.5 Illustration of pipeline for Mixture of DCNN from Ge *et al*. [19]. Best viewed digitally in color. Figure taken from original paper. 13

2.6 Illustration of pipeline for multiple granularity descriptor convnet [69]. Best viewed digitally in color. Figure taken from original paper. 14
2.7 Illustration of pipeline for bilinear convnets for fine-grained recognition [44]. Best viewed digitally in color. Figure taken from original paper. 14
2.8 Illustration of pipeline for Sermanet *et al.*'s attention for fine-grained recognition [57]. Best viewed digitally in color. Figure taken from original paper. 15
2.9 Illustration of pipeline for Jaderberg *et al.*'s spatial transformer networks [32]. Best viewed digitally in color. Figure taken from original paper. 16
2.10 Illustration of multi-task neural networks frameworks in natural language processing. Best viewed digitally in color. Figures taken from original papers. 17
2.11 Illustration of pipeline for Li *et al.*'s multiview face detection [41]. Best viewed digitally in color. Figure taken from original paper. 17
2.12 Illustration of pipeline for Dai *et al.*'s instance aware semantic segmentation method with multiple loss functions [14]. Best viewed digitally in color. Figure taken from original paper. 18
2.13 Illustration of pipeline for Zamir *et al.*'s feedback networks [74]. Best viewed digitally in color. Figure taken from original paper. 19
2.14 Illustration of pipeline for Yan *et al.*'s HD-CNN [72]. Best viewed digitally in color. Figure taken from original paper. 20

3.1 A few examples of typical multi-task neural network architectures. Note that the first few layers are shared by all the tasks, and there is no constraint on what type of task can be added. Shared layers are represented in grey, while layers in task-specific branches are shown in green. Hi means the hidden layer and Ki denotes the final loss layer for task i. In the top row, from left to right, we have (a) a single task network, followed by (b) a plain multi-task network, folowed by (c) a multi-task network with task-specific hidden layers. In the bottom row, from left to right, we have (d) a network where features from some tasks are concatenated before being passed to another task, and (e) a cascading multi-task network where each task feeds into the next one. 22
3.2 Examples of relationships among classes that can be exploited for multi-task learning. *a* and *b* show relationships inherent among breeds of dogs, and types of aircrafts respectively. Best viewed digitally, in color. 24

4.1 An illustration of a single residual layer/block from He *et al.* [26]. One of the branches uses one or more convolutional blocks, while the other branch is simply an identity function, and both are combined using element-wise addition. 29
4.2 Plots showing how the fine-grained accuracy is affected by the auxiliary task coefficient at different values of the bifurcation point. Each plot shows the same trends, but for different base models. From left to right, we have ResNet20 and ResNet32 in the top row and ResNet44 and ResNet56 in the bottom row. 35

5.1 Some of the classes from the Caltech-UCSD Birds-200-2011 dataset [68]. For more examples, please visit `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`. 40

5.2 The task specific layers of the model for the CUB dataset. Shared layers are shown in grey, and are usually taken from a model pre-trained on Imagenet. Each task specific branch is color-coded, viz. orders is in green, families is in blue, genera is in red, and species is in purple. D here represents the dimensionality of the feature vector from the pre-final layer of the pre-trained model. For VGG16 and VGG19, $D = 4096$, while for all Resnets except Resnet-34, $D = 2048$. 41

List of Tables

Table Page

4.1 The taxonomy of coarse super-classes and fine-grained sub-classes for the CIFAR 100 dataset. 28

4.2 Generalized overview of the residual network architecture used for our experiments. The architectures are taken from the ones specified by He *et al*. in [26]. The following shows the base model, that is suitable only for a single task with K classes. It does not reflect the bifurcation point and task-specific private branches described above. 29

4.3 Summary of common hyperparameter options used during training different models. . 31

4.4 State-of-the-art results for the latest methods on the CIFAR 100 dataset. 32

4.5 Baseline results for the residual networks with varying depths on the CIFAR 100 dataset for both fine-grained and coarse-grained classification tasks. 32

4.6 Fine-grained accuracies obtained after running multi-task residual networks on the CIFAR 100 dataset. The primary task is fine-grained classification, with an auxiliary task of coarse-grained classification. We show here the results for a hyperparameter sweep over bifurcation points of $3, 4, 5$ and auxiliary task coefficients of $0.00, 0.25, 0.50, 0.75, 1.00$ respectively. 33

4.7 Coarse-grained accuracies obtained after running multi-task residual networks on the CIFAR 100 dataset. The primary task is fine-grained classification, with an auxiliary task of coarse-grained classification. We show here the results for a hyperparameter sweep over bifurcation points of $3, 4, 5$ and auxiliary task coefficients of $0.00, 0.25, 0.50, 0.75, 1.00$ respectively. 34

4.8 Impact of global label transformation/shuffling for both fine-grained and coarse-grained classification tasks. 37

4.9 Impact of epoch-wise label transformation/shuffling for both fine-grained and coarse-grained classification tasks. 37

5.1 Summary of statistics for Caltech-UCSD Birds-200-2011 dataset. 40

5.2 Summary of common hyperparameter options used during training different models. . 43

5.3 State-of-the-art results for the latest methods for fine-grained recognition on the Caltech-UCSD Birds-200-2011 dataset. 44

5.4 Accuracies for each possible task by fine-tuning each pre-trained model independently with and without additional fine-tuning specific hidden layers. The tasks are order, family, genus, and species classification on the Caltech-UCSD Birds 200 dataset. . . . 45

5.5 Accuracy for fine-grained recognition with the proposed variants. Please note that fine-grained species classification is given a constant weight $\alpha_{class} = 1$. MTL refers to simple multi-task learning, while $MTL - H$ implies that each task has its own private layer. Concatenating and cascading are applied as shown in figure 3.1. Dynamic task re-weighting refers to equation 3.2. Coefficients for auxiliary tasks, i.e. $\alpha_{order}, \alpha_{family}, \alpha_{genus}$ are set to 0.10. 46

5.6 Accuracy on fine-grained classification with a single-task bilinear CNN and a multi-task bilinear CNN. We use the $[M, M]$ version of the model, since fitting the larger $[D, D]$ bilinear CNN variant into a single GPU requires hyperparameters that are different from the rest of our experiments. The MTL variant is a simple multi-task model, with no additional task-specific hidden layers. 47

5.7 Table showing the improvement in terms of testing accuracy for MTL models over STL models for different tasks. The STL models were trained on each task separately, while the MTL model was trained on each task jointly. 48

A.1 The order, family, genus and species for each class name in the Caltech-UCSD Birds-200-2011 dataset [68], obtained by parsing the American Ornithologists' Union Checklist of North American Birds [9]. 59

Chapter 1

Introduction

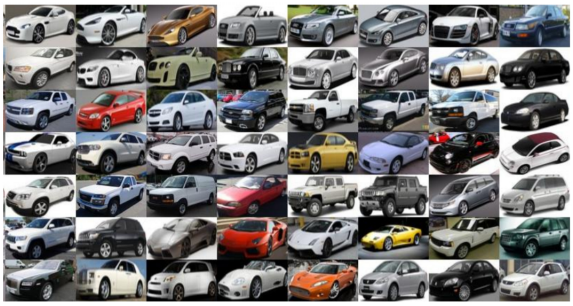
1.1 Fine Grained Recognition

Figure 1.1: An example of the generic recognition problem in computer vision. Note that for many object categories, visually distinguishing them becomes simple because of a multitude of factors such as background, illumination, shape, etc.

According to the Oxford English dictionary, the term $fine - grained$ means marginally/subtly different, involving great attention to detail. Fine-grained computer vision deals with distinguishing subordinate categories within entry-level categories, and much like the dictionary definition, the devil lies in the details for any form of fine-grained recognition. While traditional classification has been about distinguishing between different object classes, such as cars versus birds as depicted in figure 1.1, fine-grained recognition is about distinguishing subordinate categories within an entry-level category, i.e. it is more about identifying exact car models, or exact bird species. This stark difference is showcased in figure 1.2, where many of the challenges associated with fine-grained recognition can be observed as well.

The primary challenges of fine grained recognition can be summarized as the following:

- Data collection and annotation is extremely expensive, requiring domain expertise, time and manual effort for proper annotation and collection



(a) Fine-grained cars

(b) Fine-grained birds

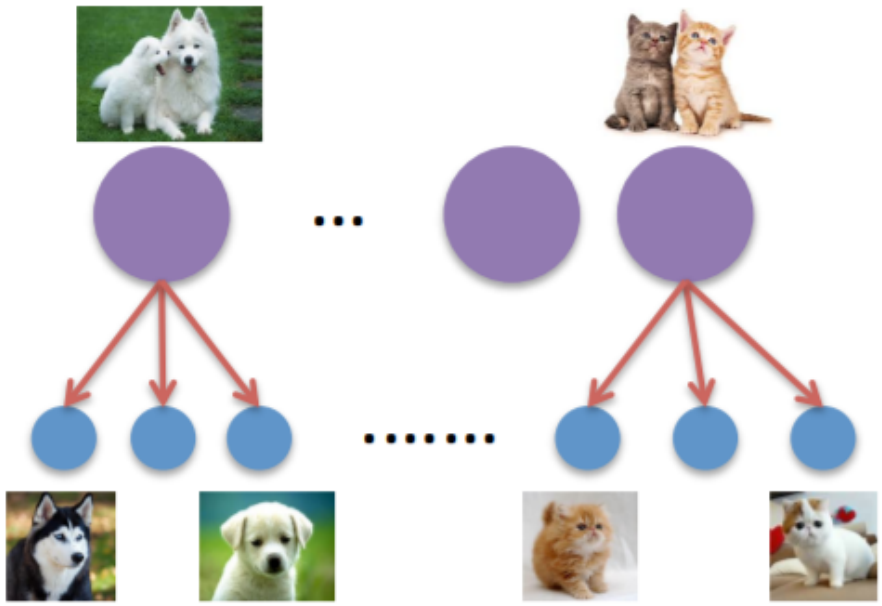
Figure 1.2: Different examples for fine-grained recognition. It is to be noted how many of the challenging aspects of fine-grained recognition can be observed from even a random sampling of fine-grained images. Best viewed enlarged, in color.

- As a result of the above, large scale datasets are difficult to obtain, and most datasets are plagued by a dearth of samples per class
- Large variations in pose and illumination across images make things more difficult
- Subtle intra-class differences and striking inter-class similarities further render the problem even more challenging

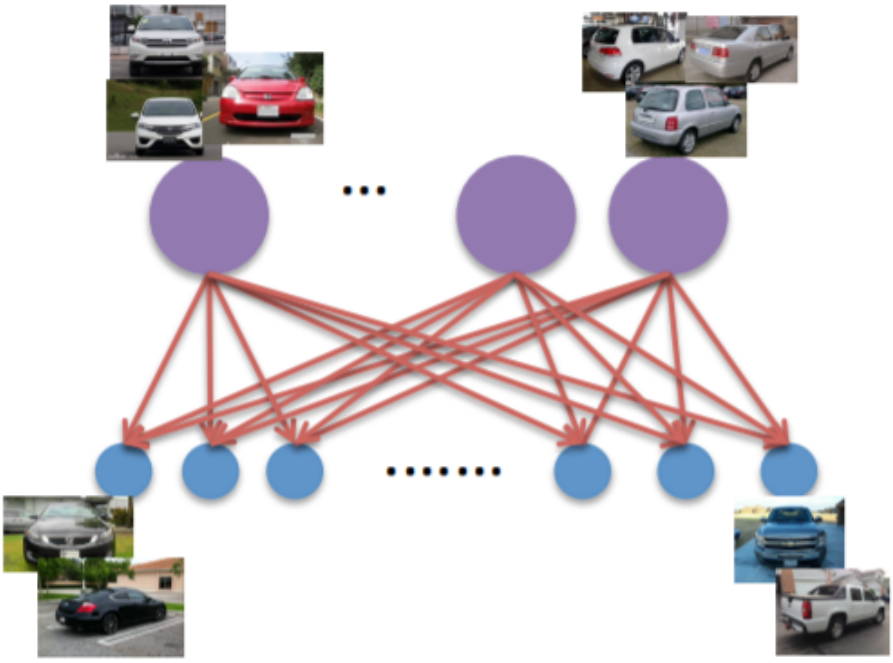
1.2 Hierarchy/Taxonomy/Ontology Based Recognition

Most modern methods for fine grained recognition rely on a combination of localizing discriminative regions and learning corresponding discriminative features. This in turn requires strong supervision, usually in the form of part, keypoint or attribute annotations, which are expensive and difficult to obtain at scale. Part or keypoint annotations are essential for localization of discriminative regions, which forms the cornerstone of most modern approaches to solving fine-grained recognition. However, unlike in generic classification where little domain expertise is required to label data and tools such as crowd-sourcing can be leveraged, these annotations are exceedingly difficult to obtain for fine-grained subordinate classes, requiring domain expertise, and extremely expensive both in terms of manual labour and time. On the other hand, since fine grained recognition deals with subordinate-level classification, there exists an implied relationship among the labels. These relationships may be taxonomical (such as super classes) or semantic (such as attributes or factors) in nature, as shown in figure 1.3 and usually exists in the form of free labels. The ontology obtained in this manner contains rich latent knowledge about finer differences between classes that can be exploited for visual classification. The embedded latent knowledge in these class hierarchies is what we propose to exploit.

If we consider a hierarchical taxonomy of super-classes and subordinate-classes, then an object at the subordinate-level carries with it all the parent labels along the path of its taxonomy tree. As an example,



(a) Fine-grained classes grouped into hierarchical super-classes, e.g. dog breeds into breed groups.



(b) Fine-grained classes grouped semantically on attributes or factors, e.g. cars into types and manufacturers.

Figure 1.3: Different examples for how relationships can be found among fine-grained classes. Figures taken from Xie *et al*. [71]. Best viewed in colour.

the downy woodpecker is Picidae, Dryobates and Dryobates Pubescens at the family, genus and species level, respectively. A further example with kingfishers is highlighted in figure 1.4. The question now becomes how one can leverage these labels to boost classification performance at the subordinate-level, which is usually the most difficult, and of the most interest. In this thesis, we propose the use of multi-task convolutional neural networks to classify images at different levels of a taxonomical hierarchy, from coarse-grained to fine-grained. Before moving onto the intricacies of our proposed method, we delve into some of the terminology, concepts and topics required throughout the rest of the thesis.

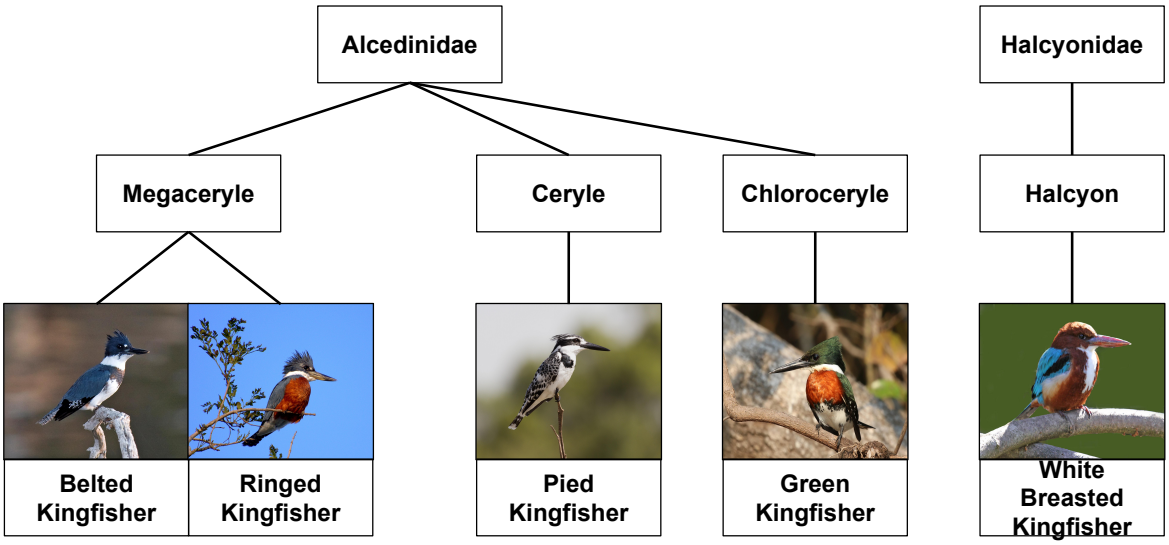


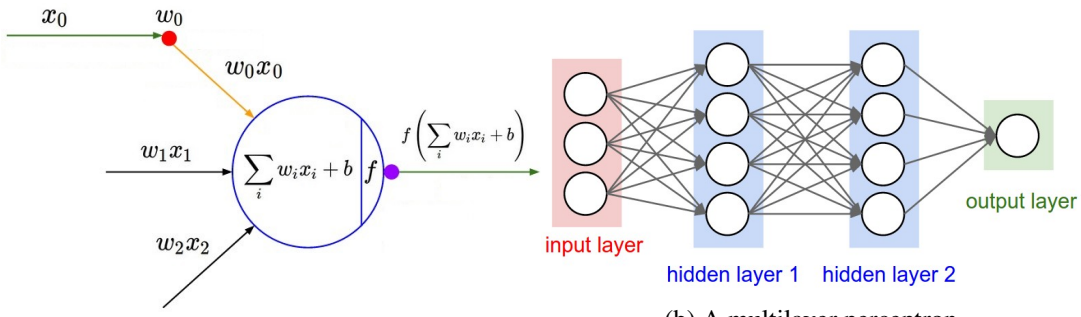
Figure 1.4: Leveraging the taxonomic ontology of birds for fine grained recognition. From top to bottom, we have family, order and species for five classes of kingfishers in the CUB-200-2011 dataset [68]. Observe how identifying the family or order can help identifying the class, e.g. in case of ringed kingfisher and green kingfisher. Best viewed enlarged, in color.

1.3 Neural Networks

Without resorting to analogies to neuroscience and biological models of the brain, the simplest way to describe a neural network succinctly is to consider it to be a sequence of non-linear transformations on input data. In mathematical terms, if x is our input vector, and W is a weight matrix, then we get a simple linear classifier by computing Wx. Extending this example, and ignoring the bias terms, a simple neural network would be computing something like $F_2(W_2(F_1(W_1(x))))$, where W_is are weight matrices, and F_is are non-linear transformations [1]. The concept of neural networks was primarily inspired by modelling biological neural systems, and each unit in a neural network is known as a neuron, a computational unit shown in the left half of figure 1.5. A single neuron, or perceptron as it is also

[1] For a comprehensive list of activation functions, visit https://nn.readthedocs.io/en/rtd/transfer/

known, can be stacked vertically into layers, and these layers can then be stacked one after the other to form a multilayer perceptron, which is the name given to simple neural networks with at least 2 layers from input to output. An example of a multilayer perceptron is shown in the right half of figure 1.5. Neural networks are extermely powerful in terms of approximating functions that model the transformation from the input space to the output space. In a supervised learning scenario, neural networks are mostly trained via backpropagation [55], which is an application of the chain rule of derivatives to the weights and biases of the layers in a neural network.



(a) A single neuron, computing $F(W \times x)$

(b) A multilayer perceptron

Figure 1.5: On the left, we have a model of a single neuron, while on the right, a simple multilayer perceptron is shown.

1.4 Convolutional Neural Networks

Traditional feed-forward neural networks are not the only variety of neural networks available. In fact, they are quite inflexible in several settings, such as when dealing with inherent structure, e.g. spatial structure in images, temporal structure in time series, etc. For such cases, there exist a plethora of different neural network families, such as convolutional neural networks for spatial data, recurrent neural networks for temporal data, etc. Since images have a spatial structure, convolutional neural networks are designed to take into account certain properties regarding images or image-like spatially or volumetrically connected data which makes the forward and backward passes of the neural network more efficient and vastly reduces the number of parameters. Traditionally, a convnet will consist of multiple convolutional, pooling and fully-connected layers, along with non-linear activations and regularisers. In some modern applications, the pooling and fully-connected layers have become optional, such as in [26, 43, 60]. An example of a convnet is provided in figure 1.6. Convolutional layers operate on volumetric inputs, e.g. images represented in some $height \times width \times depth$ format, producing another volumetric output. These are usually followed by non linear activation functions, along with regularisation layers like batch normalization. These are usually followed by pooling layers that perform downsampling on the spatial dimensions, usually keeping the depth fixed. At the end, we usually have some linear or fully-connected layers, usually ending with a softmax over the number of classes.

Convolutional neural networks(CNNs) first tasted mainstream success with their impressive perfor-

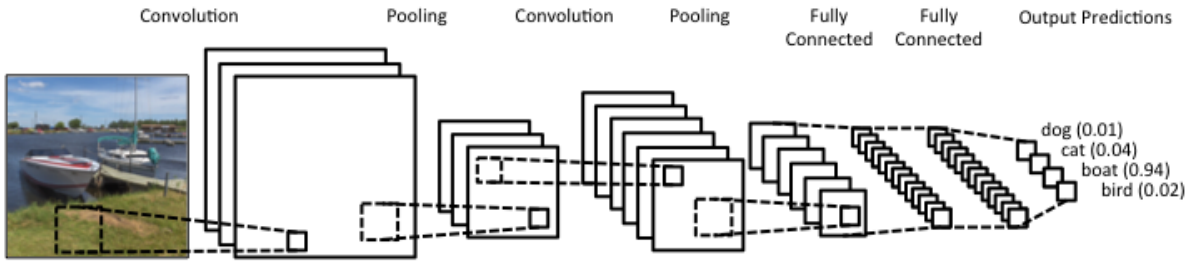


Figure 1.6: A simple convolutional neural network, with convolutional, pooling and fully-connected layers leading to a softmax based classification output. Dashed boxes show the filter size at each layer, i.e. the patch on which that layer operates in a sliding window fashion.

mance on large scale image recognition challenges, starting with Krizhevsky *et al*. [39], which brought them into the limelight. Modern convnets for large-scale datasets like Imagenet have kept increasing in depth, complexity and performance since the days of AlexNet. Since AlexNet arrived on the scene with 60 million learnable parameters and 7 convolutional/fully-connected layers, we have had in subsequent years challenge-winning architectures spanning from VGG-16 [59] with 16 convolutional/fully-connected layers and 138 parameters, GoogleNet [64] with 22 convolutional/fully-connected layers and 5 million learnable parameters, and finally residual networks [26], which have between 18-200 convolutional/fully-connected layers! Figure 1.7 shows an overview of some of these deep convolutional neural networks.

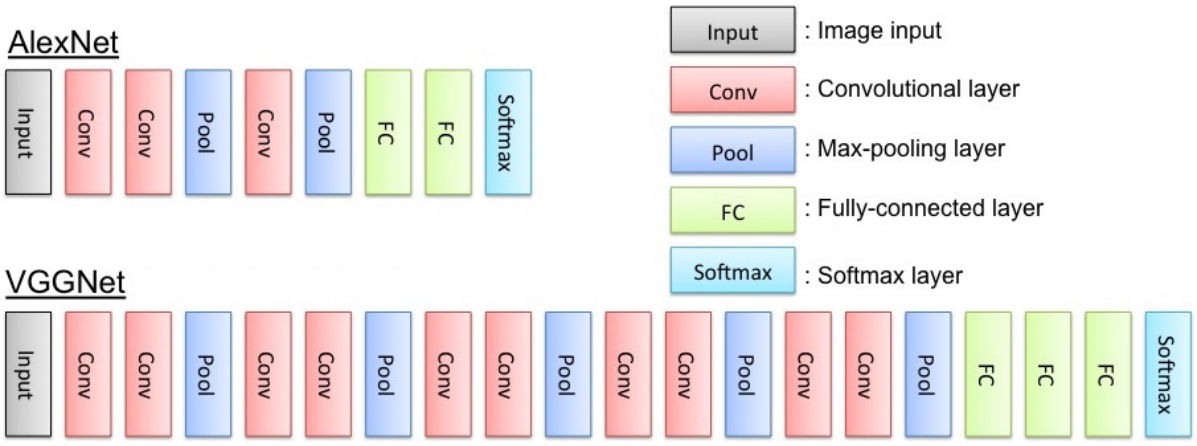


Figure 1.7: Block diagram representation of two standard deep convolutional neural networks for large-scale image classification, viz. Alexnet [39] and VGGNet [59]. Deeper models such as GoogleNet [64] and ResNets [26] are not shown here due to lack of space. Interested readers can visit this link.

1.5 Multi-Task Learning

Multi-task learning, or MTL, first explored by Caruana in [6] and by Mostafa in [2], is a machine learning technique dealing with optimising multiple loss functions jointly, usually to achieve improved performance on one or more of the loss functions being optimised. Caruana summarizes the goal of MTL succinctly as:

> MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks.

Figure 1.8 shows an example of a MTL neural network, where the input x is processed via multiple shared hidden layers, and finally branches off into N separate task-specific branches, giving rise to N outputs $y_1, ..., y_N$. MTL is effectively a form of inductive transfer, which can help improve a model by introducing an inductive bias provided in the form of auxiliary tasks, causing the model to prefer hypotheses that explain more than one task.

The model we propose consists of a single deep convolutional neural network, with each level of the

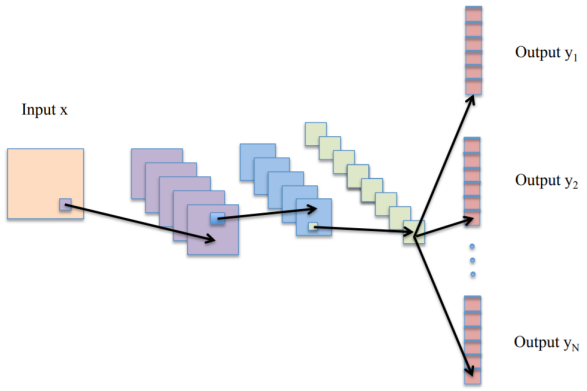


Figure 1.8: An example of a multi-task network. Figure taken from [23]. Best viewed enlarged, in color.

taxonomy tree giving rise to an additional set of labels for the input images. These additional labels are used as auxiliary tasks for a multi-task neural network, which can be trained end-to-end using a simple weighted objective function. The convolutional neural network is trained in an end-to-end fashion for multiple tasks by having a number of shared convolutional layers, followed by task-specific branches ending in task-specific layers. For fine-grained recognition by leveraging the taxonomy tree, the auxiliary tasks are basically classifying the input image at different levels of the taxonomy tree, and thus all the auxiliary tasks are in effect classification tasks. A primary point of concern in multi-task learning remains how to most efficiently combine the contributions from all the auxiliary tasks. Typically, for neural networks, the global objective function to be optimized becomes a weighted sum of the individual objective function for each task, and the weights or coefficients for each auxiliary task becomes a hyperparameter to be tuned. In order to alleviate this hyperparameter tuning, we also propose a novel

method to dynamically update the learning rates (hereforth referred to as the task coefficients) for each task in the multi-task network, based on its relatedness to the primary task.

1.6 MTL as Regularization for Convnets

Training a convnet from scratch is usually too expensive and will not result in the same discriminative power of one that is trained on a large dataset like Imagenet. A far more effective strategy is to fine-tune a convnet pre-trained on Imagenet to new datasets and/or tasks. Consequently, researchers have adapted convnets that were pre-trained on Imagenet for a vast plethora of tasks, ranging from object detection and semantic segmentation to pose estimation, depth estimation, attribute prediction, part localization, and many more. The works by Donahue *et al*. [16], Ravazian *et al*. [52], Chatfield *et al*. [8], and Oquab *et al*. [50] have shown beyond any reasonable doubt that convnets are ripe for transfer learning via fine-tuning.

Convnets are prone to over-fitting, and need heavy regularization as the parameters significantly outnumber the amount of training data. Hence it is imperative to find a way to adapt a pre-trained convnet to a new dataset and a new task with the least hassle. Regularizers like weight decay, dropout [61], batch normalization [31] play a crucial role in reducing a convnet's tendency to over-fit. We posit that simply adding multiple levels of the hierarchy as additional tasks should be enough to regularize the primary task of classification. In this work, we analyze the utility of jointly learning multiple related/auxiliary tasks that could regularize each other to prevent over-fitting, while ensuring that the network retains its discriminative capability. Much like dropout is bagging taken to the extreme, multi-task learning is analogous to boosting, if each task is considered a weak learner. We note that our model can be plugged into or used in conjunction with more complex multi-stage pipeline methods such as [30, 32, 44, 78] to further improve performance for fine grained recognition. Furthermore, this enables us to learn a single model that can be used for multiple tasks, effectively reducing the training time by a factor of T for T tasks. Our experimental results show that adding additional tasks are effective as regularizers, especially for convnets where there is not enough training data available. This is often the case in fine grained datasets, where labelled data is scarce and expensive to obtain.

1.7 Major contributions

The major contributions of this thesis are as follows:

- We show how performance on fine-grained classification can be improved by leveraging inter-class and intra-class relationships, which may be taxonomical/hierarchical (such as super classes) or semantic (such as attributes or factors) in nature.

- We show how labels at multiple levels of a taxonomy/hierarchy can be treated as auxiliary tasks in a multi-task neural network, thereby classifying input images at multiple levels of the taxonomy/hierarchy using only a single network.
- We show how taxonomical classification in a multi-task neural network has a regularizing effect, which ends up improving generalization for the multi-task neural network and overall classification accuracy on the task of interest.
- We also propose a method to use private task-specific layers to identify task relatedness, which in turn is used to dynamically update the coefficient of each task in the overall weighted objective function, alleviating the problem of picking coefficients.
- We showcase results on both toy and real-world fine-grained recognition datasets, where our models perform comparatively with state-of-the-art models, without requiring all the expensive bounding box, keypoint, part annotations that current state-of-the-art methods require.

1.8 Outline of thesis

The rest of this thesis is outlined as follows:

- Chapter 2 deals with the relevant background work which this thesis builds upon, by briefly providing an overview of convolutional neural networks, as well as summarizing the recent work done in fine-grained recognition, multi-task learning, and taxonomy based classification.
- Chapter 3 discusses the details of the proposed methods, including how to obtain the taxonomy, how to set up a multi-task network, and what architectural variants can be used for this task.
- Chapter 4 highlights our initial experiments on the CIFAR-100 dataset [37], which act as a proof of concept of how using a taxonomy as auxiliary tasks in a multi-task setting can aid fine-grained classification.
- Chapter 5 showcases the results of our method on a real-world fine-grained dataset, the Caltech-UCSD Birds-200-2011 dataset [68], and analyzes our results, demonstrating the efficacy of our proposed method compared to other state-of-the-art methods.
- Chapter 6 summarizes our work and concludes with possible future directions, some of which are being pursued.

Chapter 2

Related Work

2.1 Fine Grained Recognition

Fine-grained recognition has received significant attention in recent times, courtesy recent advancements in deep learning based computer vision solutions. The current state-of-the-art approaches to fine-grained recognition can be grouped into a few key categories, depending on use of additional information leveraged. Here, we discuss each of these different types of solutions for fine-grained classification in detail.

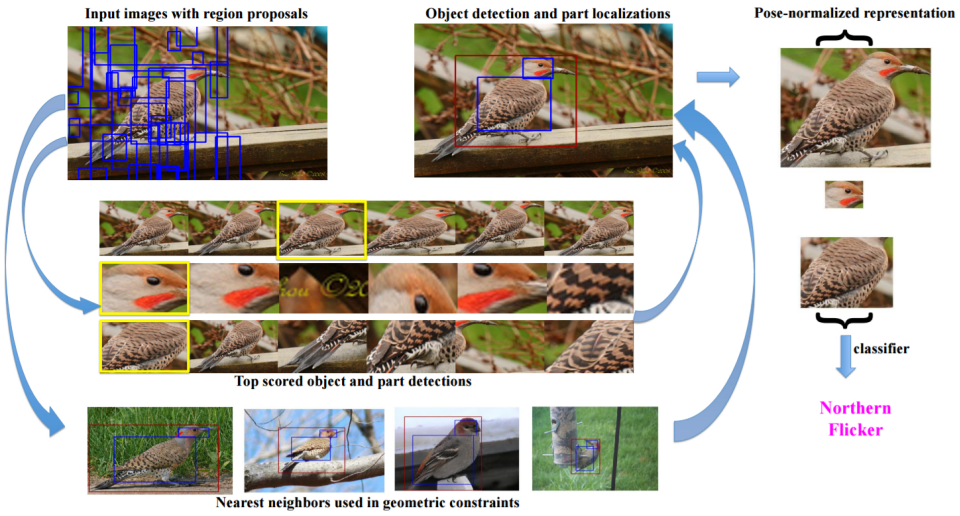


Figure 2.1: Illustration of pipeline for Part based R-CNN [77]. Best viewed digitally in color. Figure taken from original paper.

2.1.1 Part Based Methods

There exists a plethora of literature tackling fine grained recognition with the help of convnets, with many approaches relying on alignment and localization of keypoints and parts, viz. [5,30,36,42,77,78]. Zhang *et al*. [77] extend R-CNNs [22] by combining region proposals with geometric constraints to train part based networks. Region proposals from selective search are used to train object and part detectors on convnet features. For testing, all the proposals are scores by the detectors, and subsequently, geometric constraints are applied to rank all the detections and select the best proposals. These localized detections from the proposals result in a pose-normalised representation, on which a final classifier is trained for recognition, as shown in figure 2.1.

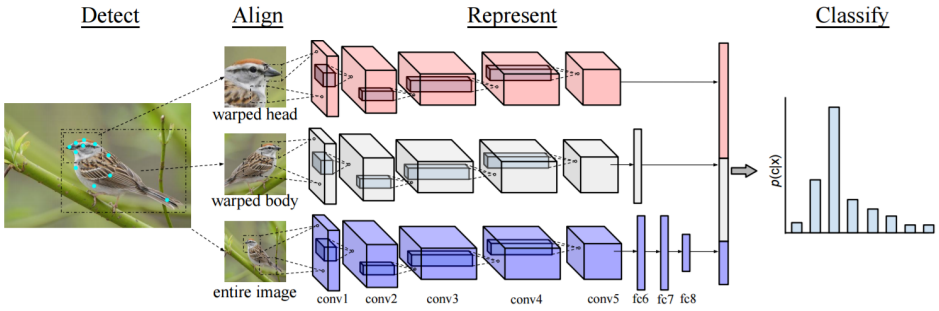


Figure 2.2: Illustration of pipeline for pose normalized deep convnet [5]. Best viewed digitally in color. Figure taken from original paper.

Branson *et al*. [5] use keypoint based templates to align bird parts and learn separate part based networks, which are then combined. A template based aligment approach inspired from deformable part models [4] is used to predict coordinate locations of keypoints, and the keypoints and corresponding bounding boxes are employed to learn pose prototypes. A final classifier is trained by concatenating the convnet features for each part, thereby combining low level and high level features, as depicted in figure 2.2. At test time, detected keypoints and bounding boxes are used to crop out warped image regions, whose features are subsequently concatenated and sent to the final classifier. Both of these methods rely on keypoints and bounding boxes requiring expensive labor-intensive annotations, which is not a strict requirement for our proposed method.

Lin *et al*. [42] formulate a complex non-parametric valve linkage function to connect localization and classification networks by aligning predicted parts and keypoints. Their alignment sub-network is formulated to chain back-propagation between the localization and classification sub-networks. It receives the localization predictions, generates a warped, pose-aligned crop, and sends it over for classification. In the backward pass, it receives classification errors and adjusts the gradients accordingly to update the localization sub-network.

More recent methods such as Huang *et al*. [30] and Zhang *et al*. [78] aim to combine attention based models and part based models by cropping parts corresponding to predicted keypoints and feeding them to discriminative feature extractors for fine grained recognition. In Huang *et al*.'s part stacked CNN,

a fully convolutional network is used for localization, and crops from localization predictions are used to train a two-stream classification network, which takes both the original image features, as well as cropped image features, as shown in figure 2.3.

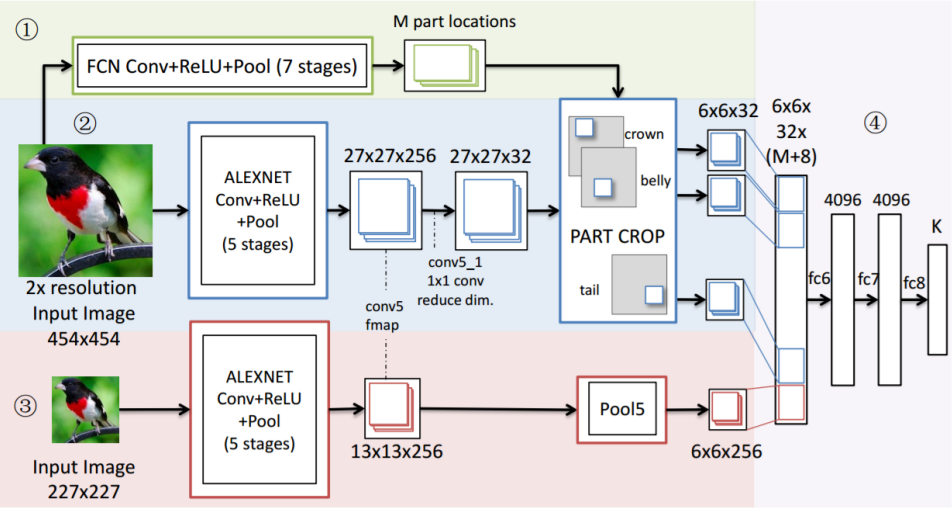


Figure 2.3: Illustration of pipeline for part-stacked CNN from Huang *et al*. [30]. Best viewed digitally in color. Figure taken from original paper.

Zhang *et al*. [78] also propose a similar model, differing only in how the localization sub-network's outputs are combined and propagated to the classification sub-network, as shown in figure 2.4. Although both these methods claim to be end-to-end trainable, both papers end up employing a stage-wise training methodology of first training the localization sub-network, then freezing it and finetuning the classification sub-network. In contrast, our proposed methods do not require such stage-wise training, but are truly end-to-end, and furthermore, also do not require the expensive localization annotations.

2.1.2 Ensemble Based Methods

An alternative way of tackling the problem of fine-grained recognition is to divide the load, either by splitting the data into similar clusters, or by using multiple models, or a combination of both, such as in [19, 20, 44, 69].

In [20], Ge *et al*. introduce subset feature learning for fine-grained recognition, where they partition the dataset into visually similar clusters, and learn a separate convnet specific to each cluster separately. Scores are obtained from each convnet corresponding to each cluster, as well as from a global convnet, and a subset selector convnet is used to compute the final classification probabilities. There are two primary drawbacks to this approach, viz. the clustering and the subset/cluster selection. Clustering is done on $fc6$ features with dimensionality reduction, which introduces significant bias into the overall model, since the $fc6$ features come from either some pre-trained or some fine-tuned network, which will be having its own set of mistakes. At test time, the subset selection also becomes tricky, relying

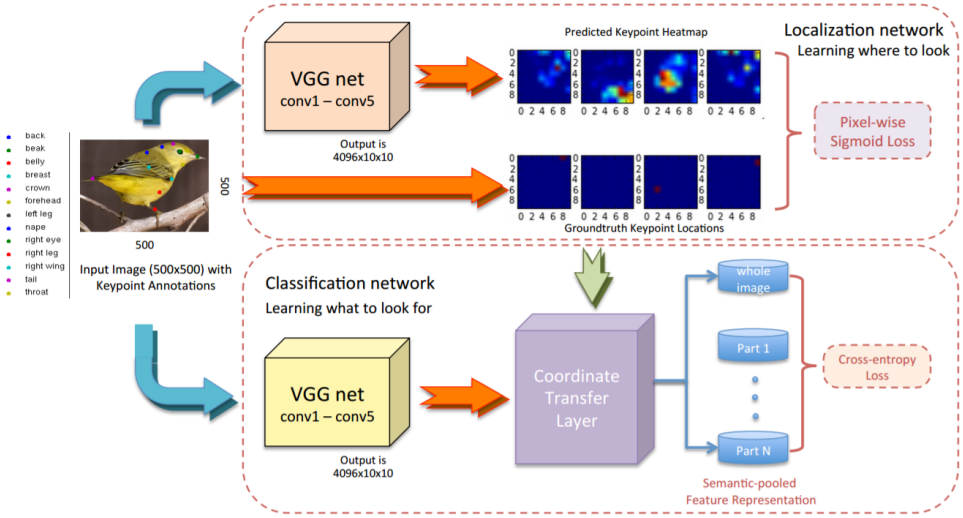


Figure 2.4: Illustration of pipeline for end-to-end model from Zhang *et al*. [78]. Best viewed digitally in color. Figure taken from original paper.

on scores from all the convnets for all the subsets/clusters to select the most likely subset and hence the most likely convnet for a given test image.

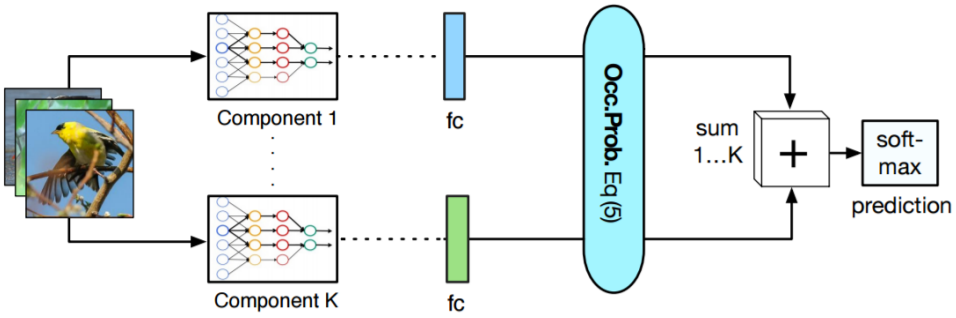


Figure 2.5: Illustration of pipeline for Mixture of DCNN from Ge *et al*. [19]. Best viewed digitally in color. Figure taken from original paper.

Quite similar to this work is the model specified by Ge *et al*. [19]. In mixture of deep convnets, as depicted in figure 2.5, there are similarly K convnets trained on K subsets of the data. However, at test time, unlike the subset feature learning method described previously, mixture of deep convnets adopts occupation probabilities to get weights for each convnet, which are then used to combine the predictions into a single final prediction for a given test image. The occupation probabilities are a function of the classification scores for each convnet, and the whole model can be trained end-to-end. The major drawback here remains partitioning the data into subsets efficiently.

Another ensemble based method is the multiple granularity descriptor [69] proposed by Wang *et al*. In this paper, the authors make the key observation that fine-grained classification implies a hierarchy of

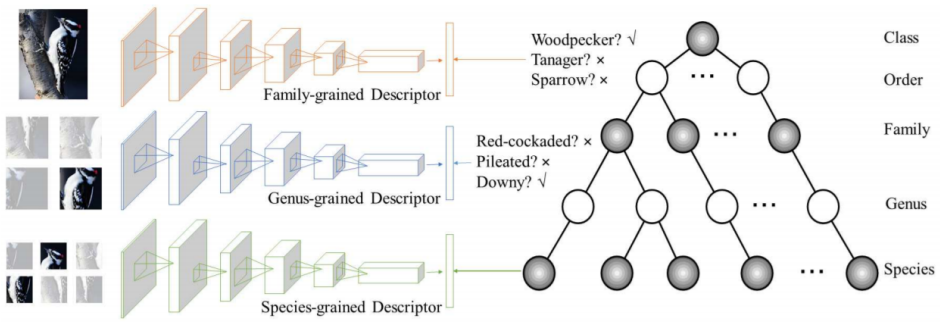


Figure 2.6: Illustration of pipeline for multiple granularity descriptor convnet [69]. Best viewed digitally in color. Figure taken from original paper.

labels, each corresponding to a level in the ontology. These free labels can be extracted cheaply, and can be used to train multiple convnets at differing levels of the hierarchy. Scores from the classifiers at different levels can then be combined to train a final classifier at the finest level as required. The primary difference of this work with our proposed method is two-fold:

- A separate convnet is trained for each level of the hierarchy, while we propose a single end-to-end model that can predict at all levels of the hierarchy
- Each level specific convnet takes as input a salient ROI instead of the entire image, thus requiring region specific annotations which are not required by our method

The framework proposed by the multiple granularity descriptor is show in figure 2.6.

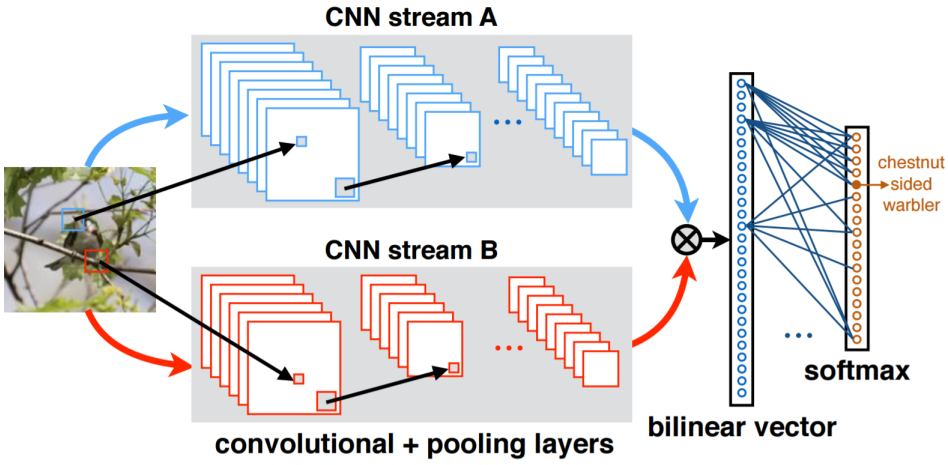


Figure 2.7: Illustration of pipeline for bilinear convnets for fine-grained recognition [44]. Best viewed digitally in color. Figure taken from original paper.

Lin *et al*. [44] use two separate convnets to extract deep features and subsequently combine them using

a bilinear layer. Two separate convnets are used as feature extractors, and their outputs are combined using an outer product at every spatial location, to effectively model rich pairwise feature interactions. Their architecture is shown in figure 2.7, and while not evident from the figure, the two feature extractors can, in practice, be the same convnet, thereby greatly reducing the complexity of the model.

2.1.3 Attention Based Methods

Attention is a rather novel approach to the problem of fine-grained recognition, which works by allowing salient regions to come under the spotlight as required, instead of focussing on the whole image. Some of the works that leverge attention mechanisms for fine-grained recognition include [32, 36, 46, 47, 57, 70, 82].

Krause *et al*. [36] use cosegmentation followed by alignment, while Xiao *et al*. [70] combines bottom-up part extraction that proposes candidate patches with top-down part-based attention that selects relevant patches to a certain object. Spectral clustering is used to group filters into clusters, where each cluster acts as a part detector. While both methods forego the requirement of annotated regions, they end up being complex multi-stage pipelines in contrast to our simple end-to-end trainable models.

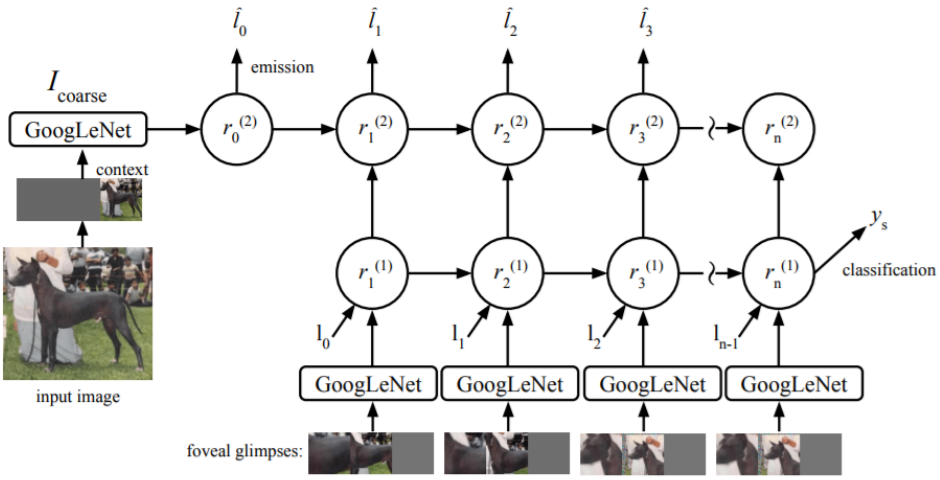


Figure 2.8: Illustration of pipeline for Sermanet *et al*.'s attention for fine-grained recognition [57]. Best viewed digitally in color. Figure taken from original paper.

Sermanet *et al*. [57] take inspiration from how humans perform visual recognition of fine-grained objects, and propose a deep recurrent neural network that processes the entire image in terms of multi-resolution crops known as glimpses at each timestep. Figure 2.8 shows the attention module, which uses a recurrent neural network in conjunction with a convnet which takes a glimpse of the original image as input at every timestep, and predicts the next glimpse location along with classification scores.

Jaderberg *et al*. [32] employ spatial transformer networks for the task. Spatial transformers are a dif-

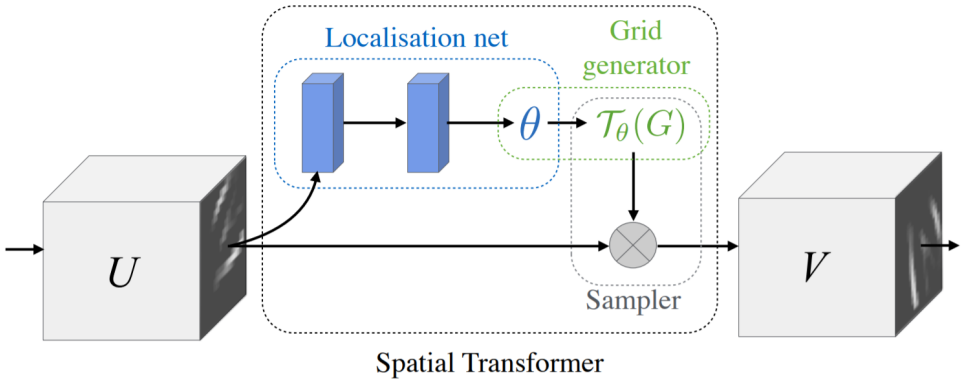


Figure 2.9: Illustration of pipeline for Jaderberg *et al.*'s spatial transformer networks [32]. Best viewed digitally in color. Figure taken from original paper.

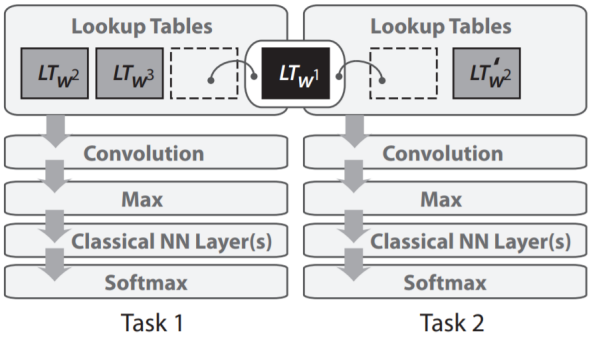
ferentiable layer that has the capacity to learn to apply a spatial warp on input feature maps to produce affine transformed output feature maps. As shown in figure 2.9, the extra spatial transformer layer performs an explicit geometric transformation on the input via means of a localisation sub-network, a grid generator, and a sampler. While these models are end-to-end trainable, they are large, slow and difficult to interpret. Our proposed method is at a clear advantage here because it can be plugged into any of these end-to-end methods to potentially achieve even higher efficacy.

While [32] and [57] are soft attention based methods, further hard attention based methods for fine-grained recognition have also cropped up in recent times. These include the methods proposed in [46, 47, 82], which rely on reinforcement learning techniques to take advantage of the hard attention mechanism. A thorough review of these methods can be found in Zhao *et al.*'s survey paper on fine-grained recognition [81].

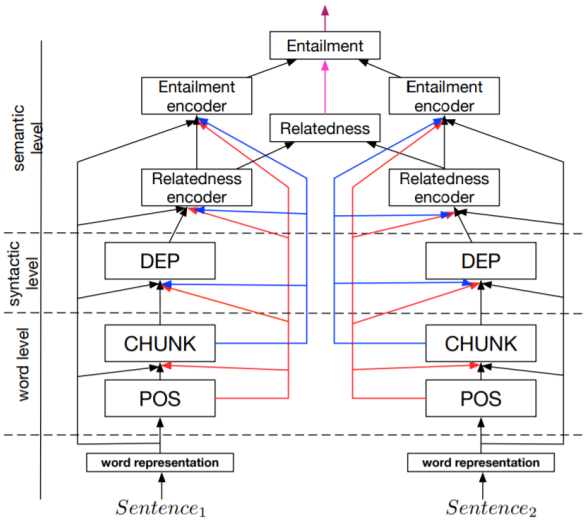
2.2 Deep Multi-task Learning

Multi-task learning in the context of deep neural networks was first extensively explored by Rich Caruana in [6]. A rigorous analysis of what works and what doesn't work for multi-task learning in neural networks can be found in Caruana's survey paper [7]. In terms of modern neural networks, the first noted work on multi-task learning is by Collobert *et al.* [13] who trained a deep neural network on three related tasks in the domain of natural language processing. This seminal work has recently been improved on by Hashimoto *et al.* [25], and the two architectures are shown in figure 2.10. A survey of recent techniques combining multi-task learning with deep neural networks can be found in [53].

Collobert *et al.*'s seminal work paved the way for deep multi-task networks, ranging from joint prediction of depth, surface normals and semantic labels [17] and joint learning of facial landmark localization, pose estimation and gender recognition [51], to instance segmentation [14] and immediacy prediction [11]. More recently, Misra *et al.* [49] presented a detailed study of how and where to share



(a) Multi-task pipeline from Collobert *et al.*'s paper [13]



(b) Multi-task pipeline from Hashimoto *et al.*'s paper [25]

Figure 2.10: Illustration of multi-task neural networks frameworks in natural language processing. Best viewed digitally in color. Figures taken from original papers.

tasks across layers in convnets.

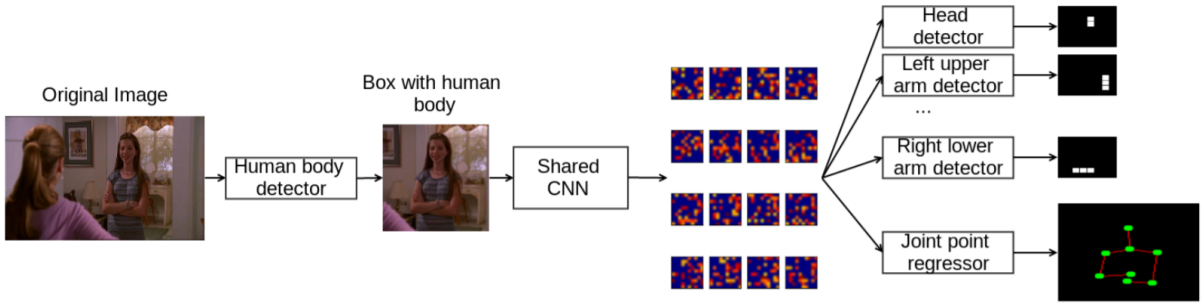


Figure 2.11: Illustration of pipeline for Li *et al.*'s multiview face detection [41]. Best viewed digitally in color. Figure taken from original paper.

Li *et al.* [41] exploit localization as an additional task to find human pose keypoints, and an illustration of their model is in figure 2.11 Zhang *et al.* [76] use pose and keypoints as additional tasks for multiview face detection. However, both methods consider all tasks to have fixed learning rates. Girshick etal [21] in their fast R-CNN paper also resort to multi-task learning, with one task being the object classification task, while the auxiliary task is a bounding box regression task for localization. The auxiliary bounding box regression, though, uses a fixed learning rate, unlike our proposed method. Zhang *et al.* [79] use auxiliary tasks such as attribute prediction to make facial keypoint detection more robust. Tian *et al.* [66] use a task-assistant CNN to jointly learn attributes to detect pedestrians. Both methods resort

to multi-step alternating gradient descent methods to tweak task weights, resulting in increased training time and complexity.

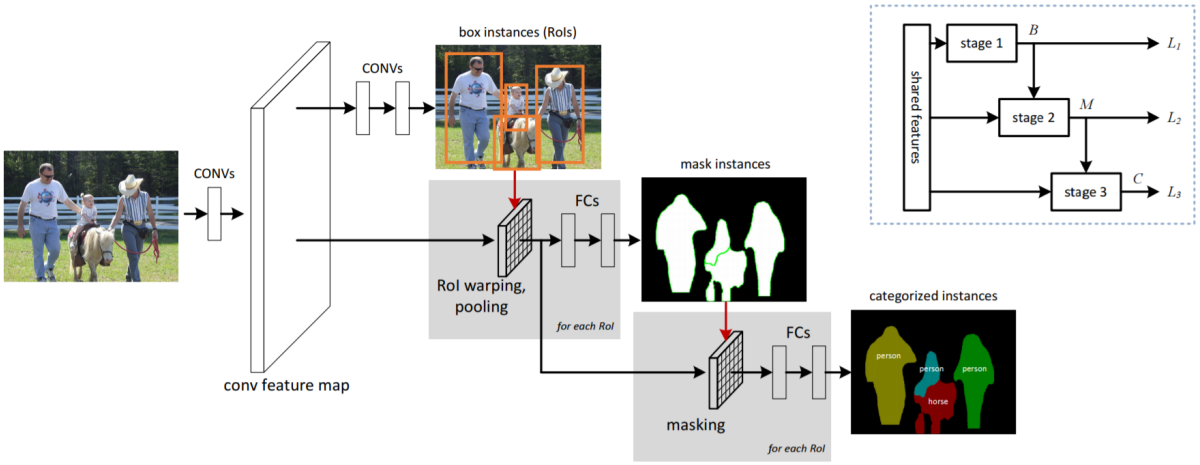


Figure 2.12: Illustration of pipeline for Dai *et al.*'s instance aware semantic segmentation method with multiple loss functions [14]. Best viewed digitally in color. Figure taken from original paper.

In more recent times, researchers have been marrying multi-task learning with deep neural network in a myriad of different ways. This has led to diverse applications such as, but not limited to, joint face detection, landmark localization, pose estimation and gender recognition for faces using multi-level feature fusion by Ranjan *et al.* [51] and cascaded multi-task networks for instance aware semantic segmentation with RoI pooling and masking by Dai *et al.* [14], as shown in figure 2.12. Kendall *et al.* [33] propose a novel and principled way to simultaneously learn multiple losses using homoscedastic task uncertainties. Gkioxari *et al.* [23] study the effect of multi-task learning using R-CNNs fine-tuned for pose estimation and action classification. Teterwak [65] provides a succinct attempt at regularizing deep neural networks using multi-task learning, and perform several experiments to show that bifurcating a convnet to learn hierarchies of features can lead to improved recognition performance.

2.3 Taxonomy Based Classification

Perhaps closest to our approach lies the work done by Wang *et al.* [69], Deng *et al.* [15] and Srivastava *et al.* [62]. Wang *et al.* [69] claim that a set of classification labels at the subordinate level implies a hierarchy of labels. Their work involves separate models being learnt for each level of the hierarchy and fused for global recognition. We instead aim to employ multi-task learning to regularize the subordinate level classification using the other levels of the ontology tree. This makes our proposed method work with a single model, which can be much smaller as well as end-to-end trainable. [15] uses Hierarchy and Exclusion or HEX graphs to model the inherent relationships between labels, while [62] leverage tree-based priors over classification labels to organise them into a tree hierarchy. Both of these meth-

ods have the advantage of not relying on pre-defined class hierarchies or relationships. However, [62] requires an initial tree which they initialise via Wordnet, as well as a non-parametric Chinese restaurant process which checks whether every class to see if it should be assigned to an existing parent in the tree or to a new branch. On a related note, the HEX graphs of [15] are more suited for a zero-shot classification setting, where existing attributes can be used to infer where in the class hierarchy a new class might lie.

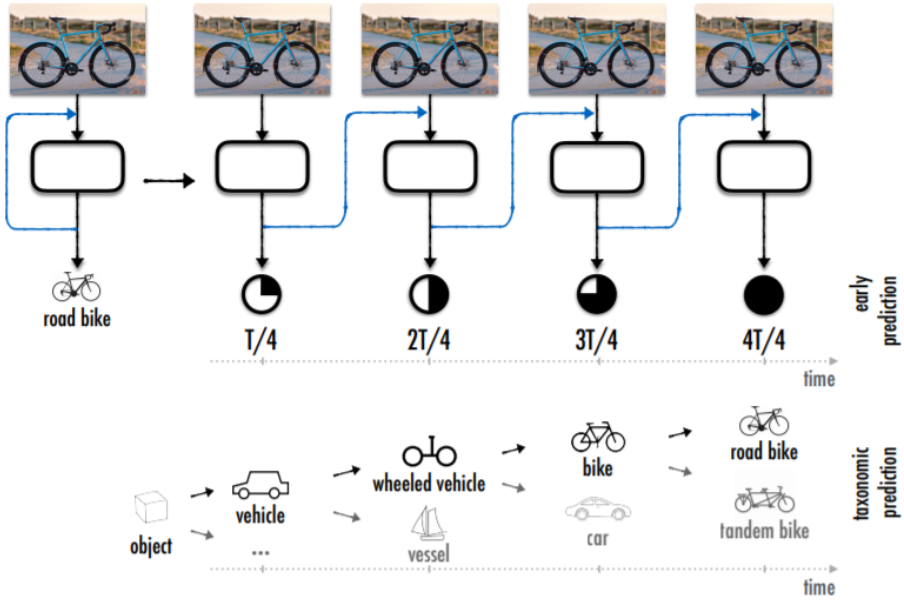


Figure 2.13: Illustration of pipeline for Zamir *et al.*'s feedback networks [74]. Best viewed digitally in color. Figure taken from original paper.

Zamir *et al.* [74] establish an iterative, feedback based model, combining convolutional networks and recurrent networks, which leverages class hierarchies and curriculum learning to classify an image at multiple levels of granularity. Their system is highlighted in figure 2.13. Yan *et al.* [72] also attempy to learn a hierarchical model, where an initial coarse-grained model identifies which fine-grained model to be used, and each coarse-grained class has its own fine-grained model, trained only on its children classes. Naturally, as can be seen from their pipeline in figure 2.14, this leads to a huge blowup in terms of parameters and memory, making it infeasible to scale up to too many fine-grained classes.

2.4 Dynamic Multi Task Coefficients

Zhang *et al.* [79] apply task-wise early stopping, but do not tune the task-wise rates throughout training. Zhang *et al.* [80] employ both dynamic task coefficients and task correlations, but end up requiring multiple alternating gradient updates for each mini-batch. Abdulnabi *et al.* [1] propose a latent task matrix to capture the relationship among tasks, which can only be trained via a combination

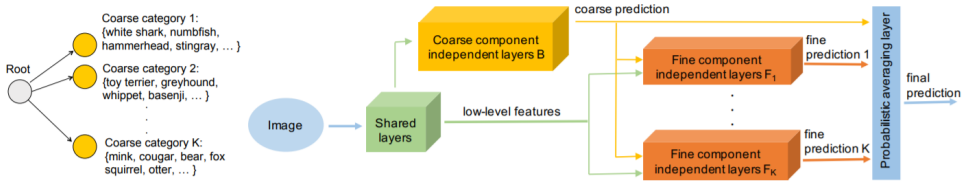


Figure 2.14: Illustration of pipeline for Yan *et al.*'s HD-CNN [72]. Best viewed digitally in color. Figure taken from original paper.

of multiple separate gradient descent steps for each mini-batch, rendering training highly impractical. We overcome many of the aforementioned restrictions in our model. We consider no constraints such as binary tasks or equal task coefficients. Our work aims to present multi-task learning as not just an easy way of training a single network for multiple tasks, but as an effective regularizer.

2.5 Chapter Summary

In this chapter, we aim to get the lay of the land with respect to the different topics that encompass our work. We observe that our work sits at the intersection of fine-grained recognition and taxonomy-based learning, with inspiration from multi-task neural networks. We survey the state of the art models for fine-grained recognition and taxonomy-based classification. In terms of recent developments combining deep neural networks with multi-task networks, we focus on a few select examples from computer vision, since a comprehensive overview is beyond our scope here. A survey of recent techniques combining multi-task learning with deep neural networks can be found in [53].

Equipped with this rich body of literature, we conceive our work which we will describe in detail in the successive chapters. In the next chapter, we will discuss how multi-task learning might work with a bifurcated convolutional neural network, with each branch tackling a different level of a class hierarchy. We propose a few architectural modifications to standard convnets, as well as a new way of adapting the task-specific coefficients in such a multi-task scenario automatically after some iterations based on the performance of each auxiliary task with respect to a primary task. Afterwards, to validate our conjecture, we test the performance on two fine-grained datasets. Initially, we will perform experiments on the CIFAR 100 dataset [38], whose small size allows us to perform a number of ablation studies to test the robustness of our proposed architectural modifications. Later, we peruse the Caltech-UCSD Birds-200-2011 dataset [68], where we try to finetune Imagenet pre-trained models to performa fine-grained classification at multiple levels of a class taxonomy.

Chapter 3

Proposed Approach

In this chapter, we strive to formulate how we wish to harness an existing class hierarchy/taxonomy to act as auxiliary tasks in a multi-task setting, so as to aid the primary task of fine-grained classification. We first define how multiple tasks can be learnt in the context of deep neural networks (deep convolutional networks in this case), followed by how different levels of a hierarchy can act as auxiliary tasks and regularize the overall multi-task model. We end the chapter by trying to fix a major point of concern for multi-task neural network, i.e. each task needs to be weighed accordingly in the joint objective function so as to ensure that no on single task overshadows all others. We propose a way of adapting the task-specific loss coefficient dynamically after every epoch based on the performance of each auxiliary task with respect to the primary task of interest.

3.1 Multiple Related Tasks

Learning multiple tasks jointly is a natural way of regularization [24] for deep neural networks that typically have a large number of parameters or weights. The crux of the idea stems from the notion that if tasks are related, then features representing the task should be related as well.
Multi-task learning requires a way to share features across tasks. In neural networks, including deep convolutional ones that are predominantly used in computer vision, regularization via multi-task learning can be accomplished by branching or bifurcation. The neural network architecture is such that all tasks share the same network weights until a bifurcation or branching point in the network, say at layer L. All layers before the branching or bifurcation are shared, while the subsequent layers are all task dependent. The features, or rather the outputs, at the point of bifurcation or branching at layer L are used as inputs for several task-specific sub-networks with different target objectives or loss functions and different task-specific architectures. One of these tasks is usually denoted as the primary task of interest, depending on the problem at hand being solved, while the other tasks are different but related tasks, known hereforth as auxiliary tasks. Figure 3.1 shows a few examples of typical convolutional neural networks designed for learning multiple tasks. Here, we aim to optimize the performance of a main or primary task T_0 with the aid of additional related/auxiliary tasks $\tau = \{1, ..., T\}$. The general

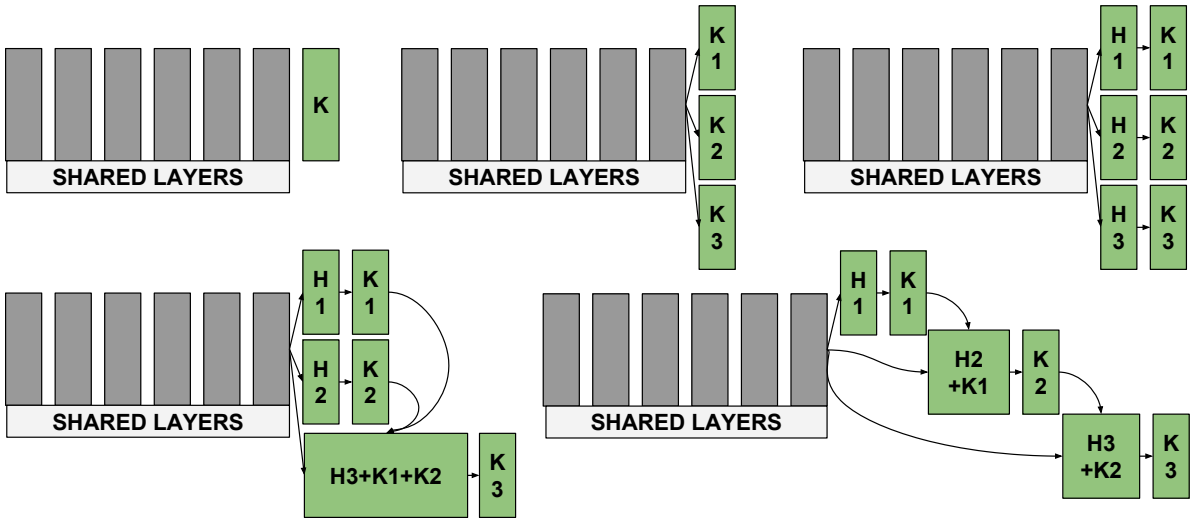


Figure 3.1: A few examples of typical multi-task neural network architectures. Note that the first few layers are shared by all the tasks, and there is no constraint on what type of task can be added. Shared layers are represented in grey, while layers in task-specific branches are shown in green. Hi means the hidden layer and Ki denotes the final loss layer for task i. In the top row, from left to right, we have (a) a single task network, followed by (b) a plain multi-task network, folowed by (c) a multi-task network with task-specific hidden layers. In the bottom row, from left to right, we have (d) a network where features from some tasks are concatenated before being passed to another task, and (e) a cascading multi-task network where each task feeds into the next one.

form of the objective function that we aim to minimize here is:

$$\underset{W_0, W_t \forall t \in \tau}{argmin} \sum_{i=1}^{N} \left[\alpha_0 l_0(y_i^0, f_0(x_i, W_0)) + \sum_{t=1}^{T} \alpha_t l_t(y_i^t, f_t(x_i, W_t)) \right] \tag{3.1}$$

The 0^{th} index refers to the primary task. For N input samples, W_t denotes the weights of the network with respect to task t, while y_i^t denotes the ground truth for the input representation x_i. f_t represents the feature transformation of the input x_i with respect to the task t and the corresponding weights W_t, and l_t is the corresponding loss function for the task. It is to be noted that $W_t = (W_s, W_{tt})$, where W_s is the shared representation, *i.e.*, the weights of the shared layers, while W_{tt} is the set of weights from the task specific layers.

Compared to traditional multi-task learning, in this formulation we can employ different loss functions for each task as appropriate. This is slightly different from traditional multi-task learning where all the tasks might be considered to be equally important. For this purpose, we associate each task t with its loss function l_t and a coefficient α_t, which acts as a coefficient determining the relative importance of the corresponding task. Note that this task-specific coefficient now becomes an additional hyperparameter that needs to be tuned, adding to the ever increasing list of knobs that need to be turned to optimize performance in deep neural network. In a later subsection, we delve into relatively old work done on

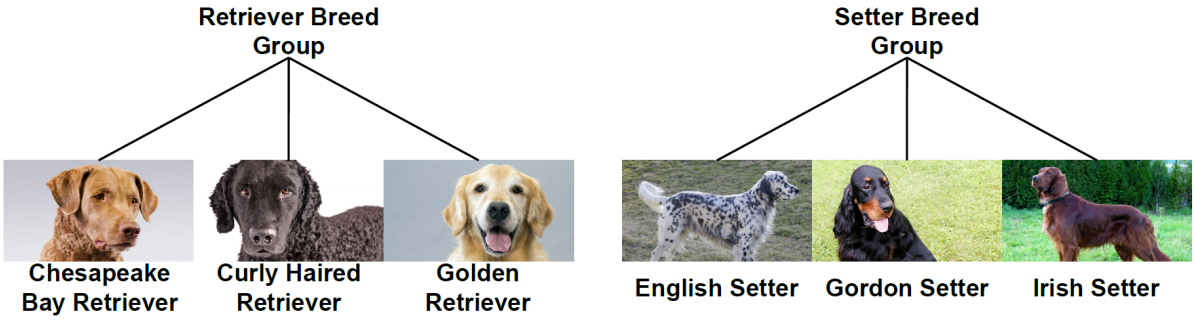
multi-task learning in neural networks, and re-formulate a decades-old technique to automate the process of selecting these task-specific coefficients such that they are dynamically updated.
Even with disparate loss functions, the entire convnet can still be trained in end-to-end fashion using vanilla backpropagation. Each loss function l_t will compute the error E_t^i with respect to an input x_i and a target y_t^i, along with a set of error gradients ∇_t^i. These gradients for each task t are then back-propagated all the way to the point of branching and bifurcation, where they are combined and propagated backwards through the shared layers. The task-specific coefficients α_t are applied to the errors E_t^i and gradients ∇_t^i to ensure that each task contributes accordingly to the global loss or objective function.
This formulation of multi-task learning faces two challenges, namely that of finding related tasks, and that of setting the proper task-specific coefficients. We now discuss our proposed way of dealing with these two hurdles.

3.2 Hierarchy as a Related Task

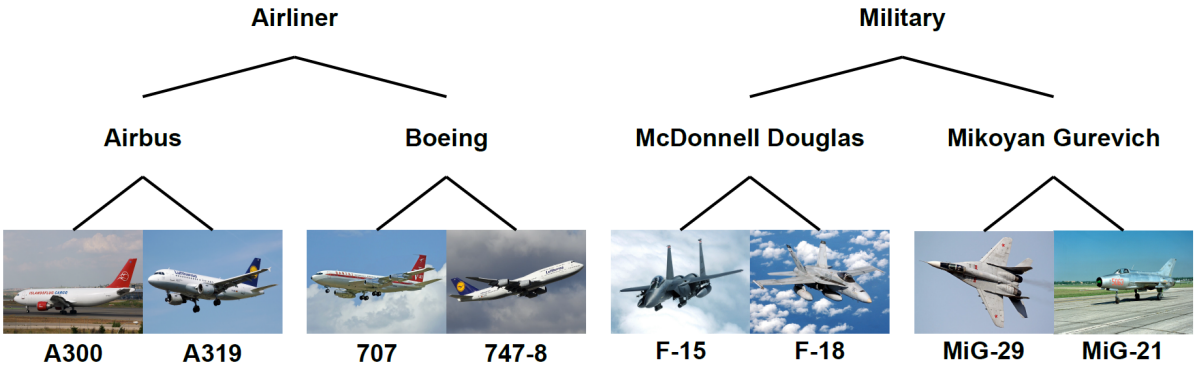
Multi-task learning works only with the correct set of tasks to learn jointly. Trying to learn unrelated tasks leads to negative transfer, resulting in poor generalization. We opine that inherent relationships present among classes can be effectively used as related tasks to regularize the primary classification task. For example, the semantic relationships in case of automobiles, *i.e*. the type of car (sedan *vs*. hatchback), manufacturer (Ford *vs*. BMW), model (Tesla Model S *vs*. Tesla Roadster), form a three level hierarchy that can be exploited for fine grained recognition. From fig. 3.2, we can observe that distinguishing commercial airliners from military fighters is easier than distinguishing a Boeing airliner from an Airbus one. One can also obtain a multiple tasks from attributes, such as ingredients of food items, or from superclasses, as shown in fig. 3.2 for breeds of dogs. Even when relationships cannot be obtained automatically, and require domain knowledge, it is far cheaper to have experts extract a ontology among classes, than to have them manually annotate each image in a dataset for keypoints, attributes, etc. We use this to our advantage, and use a hierarchy based on the scientific names according to the Linnaean taxonomy [45] which is in effect a taxonomical hierarchy. Thus, for any organism, traversing the hierarchy tree results in multiple labels depending on the level in the tree, where each label is a classification task that needs to be learnt by our model. As an example, human beings belong to $Homosapiens$ at the species level, $Homo$ at the genus level, $Primates$ at the order level, $Mammals$ at the class level, and so on. Naturally, classification can benefit from such a hierarchy of classes, since going to a higher level enables one to leverage inter-class differences to distinguish classes, while intra-class variations can be tackled by traversing to a lower level in the hierarchy.

3.3 Task-specific Coefficients

We are now left with the daunting problem of figuring out how to specify task importance. Tasks need to be initialised with a proper task-specific coefficient, which effectively weighs its contribution



(a) Hierarchy among dogs



(b) Hierarchy among aircraft

Figure 3.2: Examples of relationships among classes that can be exploited for multi-task learning. *a* and *b* show relationships inherent among breeds of dogs, and types of aircrafts respectively. Best viewed digitally, in color.

to the total loss and gradients. Furthermore, these coefficients need to be monitored and tuned during training based on whether a task is helping or hurting the performance of the primary task. We adapt the work by Silver *et al*. [58] where a separate dynamic task coefficient is introduced for each task based on a measure of relatedness of each auxiliary task with the primary task.

We extend Silver *et al*.'s measure of relatedness to work for any task with any loss function and with any number of hidden layers in its own task-specific branch, as long as each task specific branch has a linear layer with the same number of units. Like Silver *et al*., we consider that the primary task has a coefficient of α_0, and each task has a coefficient of α_t for $t = \{1, ..., T\}$, where we now consider α_t to be a measure of relatedness between the t^{th} task and the primary task, *i.e*.,

$$\alpha_t = tanh\Big(\frac{accuracy_t}{distance_t + \epsilon} \times \frac{1}{RELMIN}\Big) \tag{3.2}$$

where $accuracy_t$ is the performance measure of the t^{th} task and $distance_t$ is the Euclidean distance between the weights of the t^{th} task and the primary task. $RELMIN$ is a hyperparameter, and ϵ is a small constant $(1e-6)$ to prevent division by zero. The hyperbolic tangent clamps the task specific coefficient between 0 and 1 (as the operand is always positive), while the primary task has a coefficient

of $\alpha_0 = 1$ in our experiments. This ensures that the auxiliary tasks always contribute less than the primary task to the weighted loss.
In our model, each task specific branch has a final hidden layer with the same number of units. Thus, mathematically, $distance_t$ is computed as the distance between the weights of the final hidden layers in task t's branch and primary task's branch. We further introduce task competition by applying a softmax on the task coefficients of the auxiliary tasks. As a result, the primary task has $\alpha_0 = 1$ and for the auxiliary tasks, $\sum_{t=1}^{T} \alpha_t = 1$. Since each auxiliary task now has its contribution clamped further, this inter-task competition acts as an additional regularizer. Even though our model has task specific branches and multi-output tasks unlike [1, 66, 79, 80], simply by making the hidden layer of each task have the same size we can dynamically update task specific coefficients, and by smoothening the aforementioned coefficients using a softmax function, we obtain task wise feature competition.

3.4 Chapter Summary

We discuss how multi-task learning might work in conjunction with convolutional neural networks via multiple loss functions at the end of multiple task specific branches, with each branch tackling a different level of a series of hierarchical classes. We propose a few architectural modifications to standard convnets, as well as a new way of adapting the task-specific coefficients in such a multi-task scenario automatically after some iterations based on the performance of each auxiliary task with respect to a primary task. Now that we have defined the different architectural changes we wish to apply to standard convolutional networks, it is time for us to experiment on some dataset to see whether multiple hierarchical classification tasks can provide regularization via multi-task learning. In order to verify the plausibility of our claims, we look at fine-grained classification datasets, where the fine-grained classes can be organised into higher-level super-classes. While there are a number of fine-grained datasets available, we choose to peruse the CIFAR 100 dataset [38], which has smaller images and a 2 level hierarchy, as well as the Caltech-UCSD Birds-200-2011 dataset [68], which has larger images and a 4 level hierarchy for our experiments.

- In the following chapter, we discuss the results of running ablation experiments using multi-task residual networks on the CIFAR 100 dataset, to identify how robust our proposed multi-task network variants are to architectural design choices, such as bifurcation point and task-specific coefficients. CIFAR 100 involves 32×32 images, and has a 2-level hierarchy only, and as such we can iterate fast, trying out a multitude of different design choices to understand whether what works for the goose also works for the gander.

- Further, in the chapter following the next, we move on to a more real world problem, where we try to classify birds into different levels of the Linnaean taxonomy [45] which form a hierarchical taxonomy. This involves the Caltech-UCSD Birds-200-2011 dataset, which has larger images

than CIFAR 100, along with a 4 level hierarchy, and here we highlight how we can leverage the benefits arising from fine-tuning Imagenet pretrained models.

It should be noted that there are additional fine-grained datasets available, which might be grouped into higher-level super-classes, but we did not peruse them for certain logistical reasons, viz.:

- ImageNet Large Scale Visual Recognition Challenge [56] dataset for image classification: While classes in the ILSVRC dataset can be grouped together based on Wordnet hierarchies, the computational resources required to train modern state-of-the-art convnets on this dataset are quite exorbitant given our computation budget. Notably, a single residual network of depth 32 requires over 4 days to reach convergence using 4 TitanX GPUs and a fast SSD, with more time required for larger residual networks. Given that most pretrained models are trained on the ILSVRC dataset, the option to start from a pretrained model was also not available for this dataset, and as such we decided not to pursue this direction.

- Stanford Dogs [34] fine-grained dataset: While the 120 fine-grained dog breeds can be grouped into breed groups [1], the entire dataset is a subset of the ILSVRC dataset, and as such, papers using this dataset such as Sermanet *et al*. [57] pretrain their own models on the ILSVRC dataset after removing the common images. This results in the same issues as noted in the previous point. Further, CIFAR 100 already serves as an example of a smaller 2 level hierarchy, and so we do not believe we could have gleaned any new insight from another dataset with a 2 level hierarchy.

[1] https://en.wikipedia.org/wiki/Breed_group_(dog)

Chapter 4

Experimental Results on CIFAR 100

Before we move on to real-world, fine-grained datasets, and try to train deep convolutional networks with multiple branches and loss functions simultaneously, we thought it prudent to first test our ideas on a small-scale dataset. Since modern deep neural networks are resource hungry, we thought it would be sagacious to have some results on a small dataset that could serve as a proof of concept for our ideas, before we move on to more challenging settings.

4.1 Dataset details

To highlight the efficacy of our proposed methods, we select the CIFAR 100 dataset [38], which was collected by Krizhevsky *et al*. [37]. This dataset, along with its sister data, CIFAR 10 [37, 38], are labelled subsets of the 80 Million Tiny Images dataset from Torralba *et al*. [67]. The CIFAR 100 dataset consists of 60000 colour images, each of resolution $32x32$. There are 50000 training images and 10000 test images. There are 100 fine-grained classes, each with 600 images, and these fine-grained classes are grouped in 20 super-classes. This entails that each image is annotated with a $fine$ label and a $coarse$ label, denoting the fine-grained class and the super-class to which it belongs, respectively.
Table 4.1 contains the detailed list of coarse super-classes and fine-grained sub-classes in the dataset, as provided by Alex Krizhevsky. Notice how the grouping of fine-grained sub-classes into coarse super-classes is semantic, and not always an exact hierarchy. For example, the super-class $Vehicles1$ contains vehicles that are regularly used for transportation, while $Vehicles2$ contains objects that might be considered man-made objects with motors or engines capable of locomotion. Some super-classes are oddly specific, whereas others are quite generic. This is symptomatic of how real-world hierarchies and taxonomies might be, where different areas might have different levels of detailed annotation available. As such, we believe this makes the CIFAR 100 a potentially good dataset to try out initial experiments on, to observe the effect of fine-tuning with a hierarchy of classes that is not too perfect. Classifying each image into the corresponding fine-grained sub-class and coarse-grained super-class constitute the two tasks in our multi-task model, with fine-grained classification being the primary task and coarse-grained classification being the auxiliary task.

Table 4.1: The taxonomy of coarse super-classes and fine-grained sub-classes for the CIFAR 100 dataset.

Coarse-Grained Super-Class	**Fine-grained Sub-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

Note that we are not the first to suggest using the CIFAR datasets as a testbed for ideas before moving on to more challenging large-scale dataset. The same was, in fact, done by He etal [26] where they first prove that residual learning can indeed work on the CIFAR 10 dataset, before scaling up to the ImageNet Large Scale Visual Recognition Challenge [56] dataset, on which they obtained state-of-the-art results enabling them to win the challenge in 2015. The same was done by Zagoruyko *et al*. [73] for their proposed wide residual networks, and by Huang *et al*. [28] for their proposed densely connected networks, where they showcase results on CIFAR 10 and 100 datasets, before scaling up to larger datasets.

4.2 Architecture details

For our ablation based experiments, we pick the currently popular residual networks [26]. Residual networks are made up of stacks of residual blocks, where each block is a combination of convolutional layers along with skip or residual connections. Each residual block contains two branches, one with usually two convolutional blocks, while the other is usually an identity function, combined via an element-wise addition. Figure 4.1 shows an example of one such residual block. Each convolutional block is basically a convolutional layer with filters of kernel size $3x3$, followed by batch normalization and ReLU non-linearity. Each stack of residual blocks is made up of n residual blocks, where n

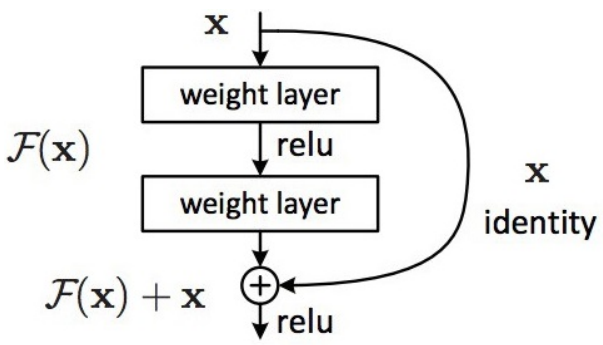


Figure 4.1: An illustration of a single residual layer/block from He *et al*. [26]. One of the branches uses one or more convolutional blocks, while the other branch is simply an identity function, and both are combined using element-wise addition.

is usually a hyperparameter controlling the overall depth of the network. The first residual block in a stack may optionally have a stride of 2 in its first convolutional layer in its first convolutional block, and this stride of 2 results in a downsampling of the input height and width by a factor of 2. For this reason, there are no separate pooling layers in residual networks. For the CIFAR datasets, where the input image size is 32×32 pixels, the residual network architecture starts with a single convolutional block, followed by 3 stacks of residual blocks. The 2^{nd} and 3^{rd} residual stack have a stride of 2 in the 1^{st} convolutional block in the 1^{st} convolutional layer, reducing the spatial dimensions first to $16x16$, then $8x8$. After this, there is a spatial average pooling layer and a non-linear mapping to the number of classes with a softmax at the end. The details of the architecture are summarized in table 4.2. We

Table 4.2: Generalized overview of the residual network architecture used for our experiments. The architectures are taken from the ones specified by He *et al*. in [26]. The following shows the base model, that is suitable only for a single task with K classes. It does not reflect the bifurcation point and task-specific private branches described above.

Layer/Group	**Kernel size, No. of filters**	**Number of Blocks**	**Output Size**
ConvBlock	[3x3, 16]	-	16x32x32
ResidualStack 1	[3x3, 16] [3x3, 16]	n	16x32x32
ResidualStack 2	[3x3, 32] [3x3, 32]	n	32x16x16
ResidualStack 3	[3x3, 64] [3x3, 64]	n	64x8x8
Average Pooling	8x8	-	64x1
Linear	K	-	Kx1
Softmax	K	-	Kx1

experiment with different values of n, just like in the original paper [26], and have used the following models for our experiments, viz. ResNet-20, ResNet-32, ResNet-44, and ResNet-56. Due to resource

constraints, we did not use the larger variants, notably Resnet-110 and Resnet-1202 for our experiments, but the results should transfer over to those models as well. For each of these models, we have a new choice we need to make to convert the plain model to one suitable for learning multiple tasks. Since we have two tasks, viz. fine-grained and coarse-grained classification, in the CIFAR 100 dataset, whichever residual network architecture is chosen requires to bifurcate into two task-specific private branches at some point. Based on table 4.2, we can see that the potential bifurcation points are as follows:

1. After the first convolutional block
2. After the 1^{st} residual stack
3. After the 2^{nd} residual stack
4. After the 3^{rd} residual stack
5. After the average pooling layer

Note that whenever a bifurcation point is chosen, it implies that after that point, the model splits into two task-specific private branches, both of which have the same architecture till the final linear layer, which maps the coarse-grained branch to 20 output units, and the fine-grained branch to 100 output units. This means that if an earlier bifurcation point is chosen, only a few layers in the overall model are shared among tasks, whereas if a later bifurcation point is chosen, most layers are shared among tasks with only a few layers remaining task-specific. The task of choosing which layer to bifurcate at is a challenging one and is compounded as the number of tasks and the complexity of tasks increases. Recently, Misra *et al*. [49] propose Cross-Stitch Networks, Ruder *et al*. [54] propose sluice networks and Lu *et al*. [48] propose fully-adaptive feature sharing to tackel this problem of sharing features in neural networks among related tasks based on task relationships and task-wise performance. However, these papers deal with tasks that are loosely related, in the sense that the relationships cannot always be clearly described, while we deal with tasks where the inter-class relationship is quite explicit in the form of a taxonomy/hierarchy/ontology. We perform ablation studies to pick the optimal bifurcation point and auxiliary task coefficients in our experiments.

4.3 Hyperparameter details

Most of the hyperparameter choices are summarized in table 5.2. Since the images are quite small, only 32×32 pixels, we can afford to use a larger batch-size of 128 on 1 GPU, which is similar to the batch-size of 32 across each of 4 GPUs used in the official code [1] released from Facebook. We use the same learning rate schedule as specified in the codebase, which reduces the learning rate by a factor of 10 after the 81^{st} epoch and again after the 122^{nd} epoch. The training goes till a total of 164 epochs, and since the turnaround time per experiment is quite low, we do not bother with any fancy early stopping mechanism. We perform ablation studies on two of the hyperparameters, viz.:

[1] https://github.com/facebook/fb.resnet.torch

Bifurcation point $3/4/5$ (2 is skipped for efficiency reasons explained below)

Auxiliary task coefficient $0.00/0.25/0.50/0.75/1.00$ (0.00 is similar to baseline single-task model)

For bifurcation point, we do not consider 2 for our experiments, since that would entail a multi-task model with only one convolutional layer shared among two tasks. Such a model not only has too little feature sharing to analyse the effect of multi-task learning, it also becomes almost as large as two separate single-task models, and thus is computationally ineffective as a multi-task model. For these reasons, we opt not to use this. For our baselines, we train the various residual networks architectures for each of the two tasks of fine-grained classification and coarse-grained classification separately. Note that in our ablation experiments, setting the auxiliary task coefficient to be 0.00 would result in a single-task baseline model.

Table 4.3: Summary of common hyperparameter options used during training different models.

Hyperparameter	**Value**
Batch size	128
Learning rate	0.1
Weight decay	0.0001
Momentum coefficient	0.9
Optimization method	SGD
Maximum epochs	164
Learning rate decay factor	0.1
Learning rate decay patience	80 and 120 epochs
Auxiliary task coefficients	0.00/0.25/0.50/0.75/1.00
Primary task coefficient	1.0
Bifurcation point	3, 4, 5

4.4 Experimental results

4.4.1 State-of-the-art Models

We do not compare with state-of-the-art results for the CIFAR-100 directly, since we use plain residual networks, and most of the current state-of-the-art results on CIFAR-100, as shown in table 5.3 are obtained by methods that are modifications using residual blocks as building blocks. Most of these methods such as Stochastic Residual Networks [29], Wide Residual Networks [73] and DenseNets [28] can be adapted to be used as building blocks in our proposed models, and we believe the results shown here translate naturally to such residual network variants.

Table 4.4: State-of-the-art results for the latest methods on the CIFAR 100 dataset.

Method	Accuracy
Network in Network [43]	64.3
Deeply Supervised Net [40]	65.4
Highway Network [63]	67.6
Residual Network [26]	72.2
Pre-activation ResNet [27]	77.3
Stochastic Depth ResNet [29]	75.4
Wide ResNet [73]	81.1
Densely Connected Networks [28]	82.8

4.4.2 Baseline Models

In table 5.4, we highlight the baseline accuracies we obtain on the test set of CIFAR 100 dataset for both fine-grained and coarse-grained classification tasks trained separately, with residual network architectures of varying depths, viz. $20, 32, 44, 56$. Note that while we train baseline single-task networks for each task separately, the same could be done by training a multi-task model with two task-specific private branches, with the task-specific coefficients being set to $(0.0, 1.0)$ and $(1.0, 0.0)$ respectively. However, the multi-task models contain an additional hyperparameter of bifurcation point, and so for our baselines, we forego additional hyperparameter choices and use single-task models. Note that in the original paper on residual networks [26], and its follow-up work on pre-activation residual networks [27], the best results obtained with a 1202 layer network were at 72.18 and 77.29 respectively, which corresponds well with our observed trends in increase in accuracy as depth is increased.

Table 4.5: Baseline results for the residual networks with varying depths on the CIFAR 100 dataset for both fine-grained and coarse-grained classification tasks.

ResNet Depth	Coarse Accuracy	Fine Accuracy
20	78.08	68.32
32	79.71	69.51
44	80.13	70.76
56	80.94	71.16

4.4.3 Multi-Task Models

Tables 4.6 and 4.7 show the results of a hyperparameter sweep over bifurcation points of $3, 4, 5$ and auxiliary task coefficients of $0.00, 0.25, 0.50, 0.75, 1.00$ respectively for a 20-layer multi-task residual network on the CIFAR 100 dataset for joint classification of fine-grained classes and coarse-grained classes. There are a number of observations and conclusions that can be gleaned from the values present in the table. First of all, we can confirm that at each level of bifurcation and for each model, having

an auxiliary task coefficient of 0.00 is almost similar to the corresponding fine-grained classification baseline from table 5.4, with the coarse-grained accuracy hovering at between $4.00 - 6.00\%$. Since the number of coarse-grained classes in the CIFAR 100 dataset is 20, this coincides with the accuracy that we would obtain by random choice, which is what a coefficient of 0.00 implies.

Table 4.6: Fine-grained accuracies obtained after running multi-task residual networks on the CIFAR 100 dataset. The primary task is fine-grained classification, with an auxiliary task of coarse-grained classification. We show here the results for a hyperparameter sweep over bifurcation points of $3, 4, 5$ and auxiliary task coefficients of $0.00, 0.25, 0.50, 0.75, 1.00$ respectively.

		Fine Grained Accuracies			
Bifurcation	**Coefficient**	**ResNet20**	**ResNet32**	**ResNet44**	**Resnet56**
3	0	68.44	69.89	70.69	70.91
	0.25	69.22	70.23	71.19	71.36
	0.5	68.85	70.04	71.23	71.41
	0.75	68.68	70.07	70.81	71.29
	1	68.12	69.58	70.32	71.05
4	0	68.29	69.65	70.51	71.00
	0.25	68.58	69.93	70.72	71.52
	0.5	68.99	70.42	70.97	71.65
	0.75	68.18	69.8	70.62	71.42
	1	67.99	69.51	70.03	71.01
5	0	68.35	69.49	70.39	71.06
	0.25	68.49	69.97	70.65	71.33
	0.5	68.56	70.11	71.11	71.92
	0.75	68.37	69.79	70.15	71.47
	1	68.07	69.25	69.71	70.89

Next, we can analyse the effect of bifurcation or branching points, and observe that different bifurcation points do not have any noticeable or quantifiable impact on the accuracy. Different bifurcation points do perform better than others for individual models, but the differences are marginal. As a result, we can say that for hierarchical classification in a multi-task setting, the neural network can be branches wherever convenient. This implies that for large models and large datasets, a later bifurcation point can be chosen, while when computational power is more readily available, an earlier branching point can be opted for. This can be seen more clearly in figure 4.2.

The subplots in figure 4.2 actually show a more noticeable overall trend regarding the auxiliary task coefficients. We can observe that an increase in the auxiliary task coefficient,i.e. the coefficient for coarse-grained classification, results in an increase in the fine-grained classification accuracy till around 0.50 in all cases, and even till 0.75 in some cases. At higher values of the auxiliary task coefficient, i.e. in some cases for 0.75, and in all cases for 1.00, the higher value for auxiliary task coefficient results in a drop in fine-grained classification accuracy. What this entails is that with higher auxiliary task coefficients, the model is dragged in different diverging directions by the different tasks or loss functions,

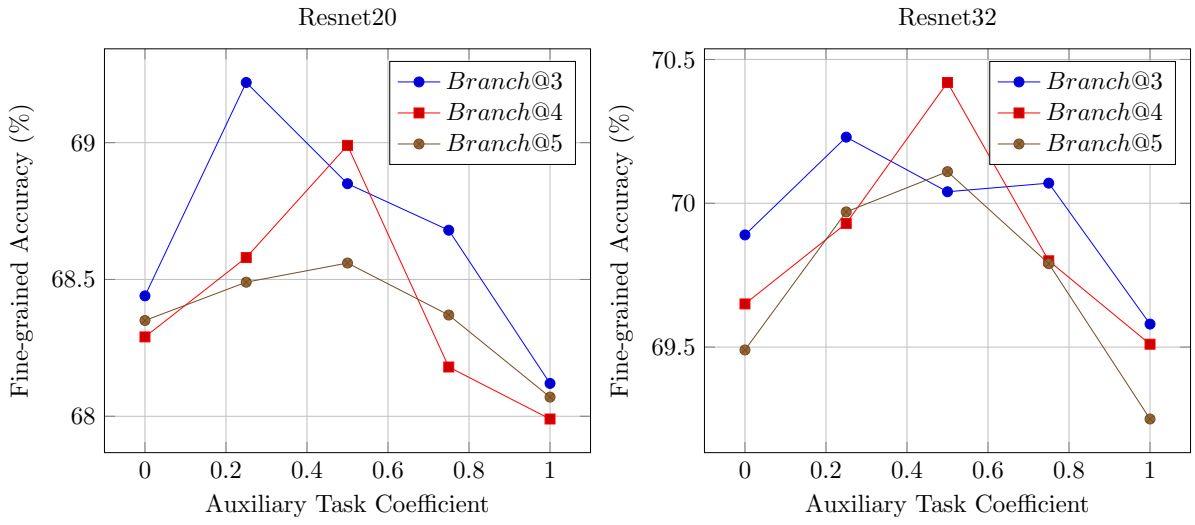
Table 4.7: Coarse-grained accuracies obtained after running multi-task residual networks on the CIFAR 100 dataset. The primary task is fine-grained classification, with an auxiliary task of coarse-grained classification. We show here the results for a hyperparameter sweep over bifurcation points of $3, 4, 5$ and auxiliary task coefficients of $0.00, 0.25, 0.50, 0.75, 1.00$ respectively.

		Coarse Grained Accuracies			
Bifurcation	**Coefficient**	**ResNet20**	**ResNet32**	**ResNet44**	**Resnet56**
3	0	4.47	4.25	5.32	5.59
	0.25	78.09	79.22	80.11	80.72
	0.5	78.36	79.52	80.45	80.97
	0.75	78.96	79.87	80.79	81.17
	1	79.21	79.7	81.01	81.09
4	0	5.7	4.11	6.03	5.24
	0.25	78.16	80.04	80.42	80.83
	0.5	78.43	80.37	80.67	81.17
	0.75	78.31	80.25	80.91	81.32
	1	78.42	80.06	80.27	81.32
5	0	4.82	6.58	5.34	4.55
	0.25	77.99	79.61	80.27	80.71
	0.5	78.13	80.34	80.91	81.16
	0.75	78.85	79.94	81.09	81.24
	1	78.14	79.45	80.73	81.44

resulting in negative task transfer. Alternatively, with lower values for auxiliary task coefficients, we observe an improvement in fine-grained accuracy. This entails that at lower values, the coarse-grained classification task actually helps the fine-grained classification task, thereby improving its classification accuracy via multi-task learning. Task relationships are notoriously difficult to model, even in case of simple taxonomical or ontological relationships as present in class hierarchies. In the previous chapter 3.3, we propose a method to dynamically update the auxiliary task coefficients such that they adapt to the task relationships without need for manual fine-tuning of coefficients as hyperparameters.
Note that for this CIFAR 100 dataset, we do not experiment with auxiliary task feature concatenation and cascading, since we do not believe additional complexities on top of very deep residual networks will give us much insights considering that the dataset is quite simple. We instead defer such architectural variants along with dynamic task coefficient updation and task-wise early stopping to the next chapter, where we deal with real-world, large-scale, fine-grained datasets.

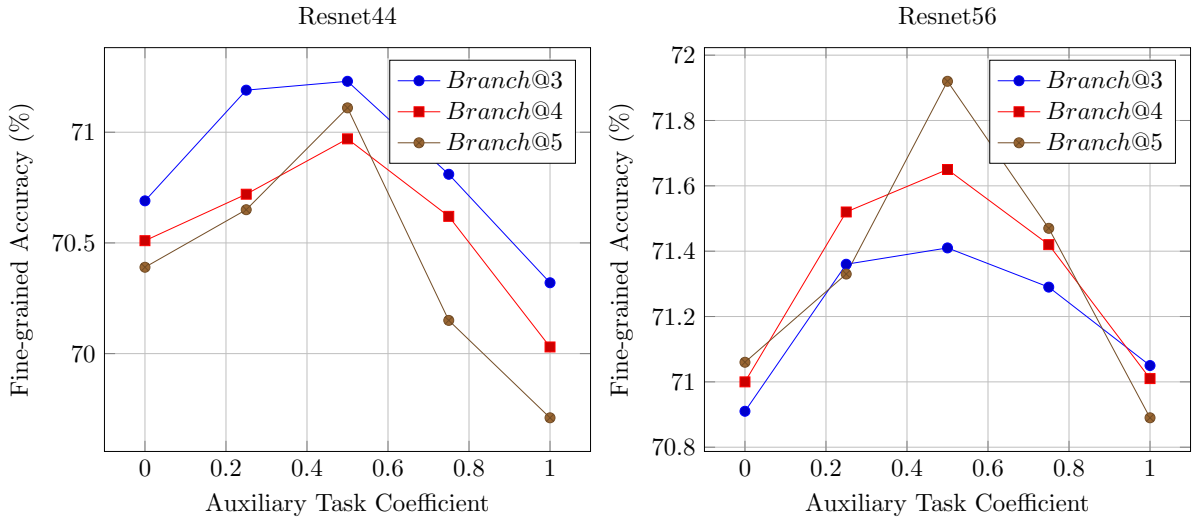
4.4.4 Effect of Hierarchy

Till now, we have been able to establish that with suitable values of auxiliary task coefficient, learning hierarchically related classification tasks as auxiliary tasks in a multi-task setting improves classification accuracy on the fine-grained classification task of primary interest. However, the question remains



(a) Resnet20

(b) Resnet32



(c) Resnet44

(d) Resnet56

Figure 4.2: Plots showing how the fine-grained accuracy is affected by the auxiliary task coefficient at different values of the bifurcation point. Each plot shows the same trends, but for different base models. From left to right, we have ResNet20 and ResNet32 in the top row and ResNet44 and ResNet56 in the bottom row.

whether the improvement obtained is because of the nature of the auxiliary task, or otherwise. In other words, we wish to verify whether it is the presence of super-classes as an auxiliary task that aids the mult-task optimization procedure, or is merely the presence of an auxiliary task in the guise of super-classes enough to push the joint loss function towards a better optimum. We posit the notion that to confirm whether the hierarchy is essential or not, we can run experiments similar to the above, but with randomly transformed labels for the auxiliary task. In order to confirm our hypothesis that the presence of a hierarchy is essential, and simply having random super-classes is not enough, we resort to the following strategies to randomly transform the coarse-grained super-class labels on both the training and testing set:

- Apply a random permutation on the label array, i.e. for $60,000$ samples, we have a corresponding label array of $60,000$ elements, varying between $1-20$ for the 20 super-classes. We create a random permutation array of length $60,000$, where every element is a unique number between $1-60,000$ and use it to randomly shuffle the labels.

- Apply a random translation to the label array, i.e. since we have a label array of length $60,000$ elements, varying between $1-20$, we randomly pick a number r between $1-19$. Now, each label is transformed, or rather, translated, by adding the selected random number to it, and taking the modulo with respect to 20, i.e. $l = (l+r)\%20$.

Note that in both of the above cases, viz. permutation or translation, we apply the random shuffling to the coarse-grained labels of both the training and testing set. We keep the fine-grained labels of the training and testing sets as is, to measure performance. Furthermore, note that we can apply the random transformation once at the beginning of the entire experiment, or alternatively, at the beginning of every epoch. Decidedly, it is expected that transforming labels randomly at the beginning of every epoch will lead to poorer performance, but we still wish to make the comparison quantitatively. Moreover, it is to be noted that for both permutation and translation based shuffling of labels, the number of samples belonging to each coarse-grained super-class remains the same, so the label distribution is not affected. For our experiments on label shuffling, we use resnets at different depths, but keep the bifurcation point fixed at 3, and the auxiliary task coefficient fixed at 0.25. We believe that the results from the previous sub-section show that results with these settings are optimal enough to transfer to other settings of these hyperparameter. Table 4.8 contains the results for shuffling via permutation and translation globally, i.e. at the beginning of an entire training run. As we can see, in case of random permutation of the labels, the coarse-grained classification task fails to converge, hovering around 5.00% classification accuracy on both training and testing sets, which is almost the same as random choice on 20 classes. In the case of translation or shifting of labels, we see that the coarse-grained classification task has reached moderate accuracy values for the training set, but still hovers at around 5.00% for the testing set, which is emblematic of severe over-fitting. Both of these results are in line with what Zhang *et al*. [75] have reported in their recent study on the robustness of deep neural networks to label noise, where they also show that deep neural networks can fit random labels, but such models usually face severe over-fitting,

Table 4.8: Impact of global label transformation/shuffling for both fine-grained and coarse-grained classification tasks.

Depth	Fine Train	Fine Test	Coarse Train	Coarse Test
Global Label Translation				
20	87.04	67.66	48.56	04.26
32	93.80	69.47	42.04	03.84
44	90.06	70.02	65.21	05.70
56	91.52	71.12	66.03	09.50
Global Label Permutation				
20	83.16	67.63	05.01	04.72
32	85.50	68.95	04.76	04.67
44	89.34	70.20	05.07	04.95
56	91.45	70.57	04.85	04.77

similar to what we observe here. Another point to note from table 4.8 is that both types of shuffling not only fail to achieve respectable accuracies on the coarse-grained classification task, but also degrade performance slightly on the fine-grained classification task, instead of improving it. This proves our conjecture that proper task relationships in the form of super-classes and sub-classes are essential to learn a hierarchical multi-task network jointly. We also show the results of applying the label shuffling strategies of permutation and translation at the beginning of every epoch in table 4.9. Similar trends prevail in this setting, compared to the one above, with respect to failure to generalize and severe overfitting for the coarse-grained classification task. Note that these results are in sync with what is expected from such a setting based on both the results obtained above as well as the results from Zhang *et al*. [75].

Table 4.9: Impact of epoch-wise label transformation/shuffling for both fine-grained and coarse-grained classification tasks.

Depth	Fine Train	Fine Test	Coarse Train	Coarse Test
Epoch-wise Label Translation				
20	81.08	67.49	50.12	08.16
32	88.19	69.10	47.66	03.25
44	91.41	70.21	50.72	03.27
56	93.30	70.63	48.52	09.31
Epoch-wise Label Permutation				
20	83.32	67.61	06.30	05.06
32	89.59	68.74	06.26	04.91
44	90.51	70.11	06.38	04.77
56	93.85	71.00	06.54	05.19

4.5 Overall Analysis and Chapter Summary

In this chapter, we have tried to apply our proposed architectural modifications to the 2 level class hierarchy of CIFAR 100. While CIFAR 100 might seem like a simple dataset owing to the nature of the relatively tiny image size of 32×32, it must be remembered that it is still quite challenging because of the sheer number of classes at play, i.e. 100 fine-grained classes at such a low resolution. Furthermore, the class hierarchy is not quite perfect, and has its own quirks in grouping some classes together to form super-classes. We maintain that the results showcased in this section vindicate our claim that a hierarchy/taxonomy/ontology of classes, even in slightly noisy, in conjunction with a multi-task neural network with multiple classification tasks seems to improve performance on the primary fine-grained task of interest. Now that we have demonstrated the veracity of our proposed ideas with the help of coarse-grained classification as an auxiliary task on the CIFAR 100 dataset, in the next chapter, we aim to demonstrate the efficacy of our method on larger, real-world, fine-grained classification datasets, with larger models and deeper class hierarchies. Our objective will be to obtain improved performance on fine-grained classification along with improved generalization.

Chapter 5

Experimental Results on Caltech UCSD Birds

In the previous chapter, we have conclusively shown that using the taxonomy of classes in a multi-task neural network can help fine-grained classification performance on the CIFAR-100 dataset. Now, we proceed to try out our proposed model on a larger, real-world, fine-grained dataset, with larger models, to see if we can achieve comparable results with state-of-the-art methods. We showcase quantitative results on the Caltech-UCSD Birds-200-2011 dataset. We use the Torch7 [12] machine learning library, on an NVIDIA Titan X GPU with 12GB of GPU memory. Each of the proposed architectural variants takes around 90 seconds per epoch on average for training, and 30 seconds per epoch on average for inference, with CUDA 8 and CUDNN 5.1 [10].

5.1 Dataset details

The Caltech-UCSD Birds-200-2011 dataset, first described by Wah *et al.* in [68] contains $11,788$ images of birds belonging to 200 species. Each image contains additional annotations consisting of 15 pairs of keypoint coordinates, 312 binary attributes and one bounding box for localization per image. Unlike most existing methods that operate on this dataset, we do not use these labels at all. The training and test splits are roughly equal in size (5994 vs. 5794), and so this dataset has limited training data, to the tune of roughly 30 training images per class. This puts it firmly in the fine-grained recognition domain, with few samples available for training per subordinate category. Regularization via multi-task learning should therefore boost the performance of the primary task. We programmatically parse the American Ornithologists' Union Checklist of North American Birds [9] to obtain the taxonomic hierarchy for avian species. For each of the 200 species present in the dataset, we extract the order, family and genus, for a total of 15 orders, 42 families, 120 genera and 200 species in the dataset. Classifying each bird image into the corresponding genus, family, order and species make up the four tasks in our multi-task model, with species level classification being the primary task. The remaining three auxiliary tasks contribute to the main goal of identifying the species of the bird present in the image. A summary of the details of the dataset are present in the table 5.1. The exact mapping between

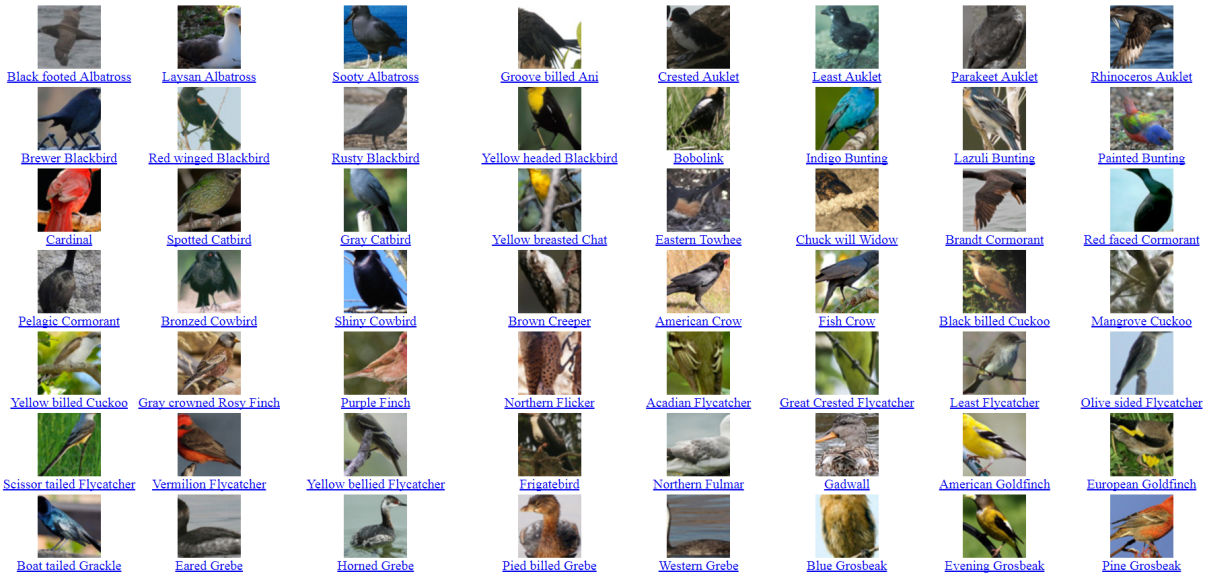


Figure 5.1: Some of the classes from the Caltech-UCSD Birds-200-2011 dataset [68]. For more examples, please visit `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`.

the original class names, and the extracted and prepared label set of order, family, genus and species names is provided in the appendix A.

Table 5.1: Summary of statistics for Caltech-UCSD Birds-200-2011 dataset.

Orders	**Families**	**Genera**	**Species**	**Training images**	**Testing images**
15	42	120	200	5994	5794

5.2 Architecture details

We use an Imagenet pre-trained model to initialise our network. We take all the layers till the last fully connected layer. For VGG-16 and VGG-19, this gives us a 4096 dimensional feature vector, whereas for ResNet-34, it gives us a 512 dimensional feature vector, and for ResNet-50, ResNet-101, ResNet-152, we are left with a 2048 dimensional feature vector. For generality's sake, let us denote the size of this feature vector with D, irrespective of the pre-trained model being used. Right after this layer, we create four branches, whose details are provided in figure 5.2 below. Each branch i goes from the D dimensional feature vector, to a final output of C_i, where C_i denotes the number of classes in the $i-th$ task. Optionally, each task specific branch might have its own private hidden layer of 512 units, which lies in between the mapping from D to C_i units, as shown in figure 5.2. This private task-specific hidden layer draws inspiration from the domain of NLP, where word embeddings are sometimes passed

through an additional non-linear mapping to translate them to the domain specific subspace more easliy. Similarly, the private task-specific hidden layers help in translating the common shared D dimensional feature vector to a more suitable subspace specific to each task. Additionally, for computing the task relatedness to dynamically update the task coefficients, the weights of the hidden layer are essential, since they give us a measure of how similar the task specific weights are. All task specific layers except the final ones have ReLU non-linearities and Dropout with a probability of 0.5. The 512 dimensional linear layer in each branch is shown in orange because it is used to compute the relatedness of the auxiliary tasks with the primary task. Figure 5.2 shows the base model, which we also adapt to form concatenated and cascaded models, similar to the ones shown in figure 3.1.

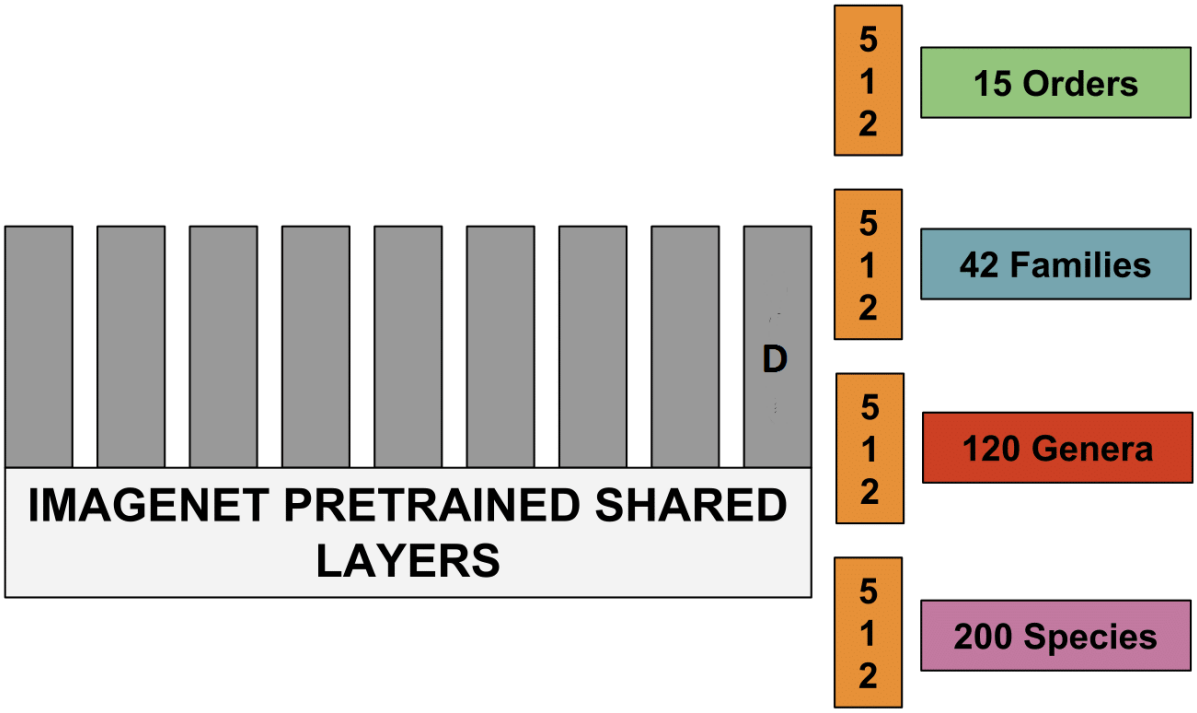


Figure 5.2: The task specific layers of the model for the CUB dataset. Shared layers are shown in grey, and are usually taken from a model pre-trained on Imagenet. Each task specific branch is color-coded, viz. orders is in green, families is in blue, genera is in red, and species is in purple. D here represents the dimensionality of the feature vector from the pre-final layer of the pre-trained model. For VGG16 and VGG19, $D = 4096$, while for all Resnets except Resnet-34, $D = 2048$.

5.3 Hyperparameter details

The joint objective function is thus a weighted sum of four terms, with the weights for species, genus, family and order classification being denoted by $\alpha_{class}, \alpha_{genus}, \alpha_{family}$ & α_{order} respectively. α_{class} is set to 1.0, while all other task coefficients are set to 0.1 initially, and updated at the end of every epoch.

We also employ task wise early stopping, where we set the coefficient for a task to 0 if its performance saturates in a fixed number of epochs. Mini-batch gradient descent is used for training, with a batch-size of 32, an initial learning rate of 0.001, Nesterov momentum of 0.9, and weight decay of $5e-4$. For the task-specific branches, the weights have a learning rate and weight decay that is 10 times the learning rate and weight decay for the pre-trained layers. The biases in the task-specific branches similarly have a learning rate and weight decay that is 20 times the learning rate and weight decay for the pre-trained layers. We set the value of $RELMIN$ to 0.05 following Silver *et al.* [58], who show that their method is robust to changes in the value of $RELMIN$. We experiment with a number of models pre-trained on the Imagenet Large Scale Visual Recognition Challenge dataset [56], viz.:

- Alexnet [39]
- VGG-16 [59]
- ResNet-34 [26]
- ResNet-50 [26]
- ResNet-101 [26]

Each of our experiments are run for a maximum of 50 epochs. Early stopping is used to stop training whenever performance has not improved for a specified number of epochs. We toy with two types of early stopping, viz.:

- Global early stopping, when the training is stopped if performance on the primary task does not improve for $patience$ consecutive epochs after the first $patience$ epochs have passed
- Task wise early stopping, where each auxiliary task is stopped after $task-wise-patience$ epochs till only the primary task remains

For our experiments, we set $patience = 10$ and $task-wise-patience = 10$ respectively. In global early stopping, the first $patience$ epochs are not considered, and early stopping only kicks in once they have passed, i.e. the model is trained for at least $patience$ epochs before we start monitoring its performance for improvements.
In addition to early stopping, we also use a specific learning rate schedule for fine-tuning. Specifically, we start with an initial base learning rate of 0.001 for the pre-trained layers, and 10 times the learning rate for the newly initialised task-specific layers. We drop the learning rate by a factor of 10 every 15 epochs. Since we train for a maximum of 50 epochs, this means that during training, our models encounter at most 3 learning rate drops, at the $15^{\text{th}}, 30^{\text{th}} and 45^{\text{th}}$ epochs, with the base learning rate dropping down to $1e-4, 1e-5, and 1e-6$ respectively. The task-specific layers train at learning rates that are 10 times these values throughout. A summary of the various hyperparameter options is present in table 5.2. Unlike in the CIFAR 100 experiments, we do not try different bifurcation points here, since for Imagenet pretrained models, such experimentation is prohibitively expensive, especially for larger models like ResNets and VGGNets. Instead, we keep the pretrained layers till the final classifier

layer shared among all tasks, and simply add different tasks using that as a point of bifurcation or branching. As baselines, we have both single task models and plain multi-task models without and with

Table 5.2: Summary of common hyperparameter options used during training different models.

Hyperparameter	**Value**
Batch size	32
Base learning rate for pre-trained layers	0.001
Base weight decay for pre-trained layers	0.0005
Learning rate multiplier for new layers	10 for weights, 20 for biases
Weight decay multiplier for new layers	10 for weights, 20 for biases
Momentum coefficient	0.9
Optimization method	SGD
Maximum epochs	50
Global early stopping patience	10 epochs
Task wise early stopping patience	10 epochs
Learning rate decay factor	0.1
Learning rate decay patience	10 epochs
RELMIN	0.05
Auxiliary task coefficients	0.1
Primary task coefficient	1.0

task-specific private hidden layers of 512 units. The specific choice of 512 is random, and any other value could have been chosen, but 512 remains large enough to retain information, yet small enough not to blow up memory and computational requirements.

5.4 Experimental results

5.4.1 State-of-the-art Models

State-of-the-art results on the Caltech-UCSD Birds-200-2011 dataset are shown in table 5.3. Current state-of-the-art methods can be easily divided into a few sub-categories, viz. part based methods, that usually rely on part detection and localisation, ensemble based methods, that usually combine multiple features from multiple models to boost performance, and attention based models, that try to find and focus on salient regions in the image to aid the classification process. Most of these methods are quite complex, and many involve multi-stage pipelines, with many moving parts and tunable knobs. As such, it is difficult to adapt most of these methods to new datasets, since many of them depend on expensive annotations such as keypoints and bounding boxes. In a later section in this very chapter, we use one of the more easily accessible state-of-the-art models, and use it as a building block for our proposed architectural changes to show how our proposed methods can be combined with state-of-the-art methods.

Table 5.3: State-of-the-art results for the latest methods for fine-grained recognition on the Caltech-UCSD Birds-200-2011 dataset.

Method	Model	Additional Requirements	Accuracy
Part Based Models			
Pose Normalized CNN [5]	Alexnet	Parts+Keypoints+BBox	75.7
Part Based R-CNN [77]	Alexnet	Parts+Keypoints+Bbox	76.4
Part Stacked CNN [30]	Alexnet	Parts+Keypoints+Bbox	76.2
Deep LAC [42]	Alexnet	Parts+Bbox	80.2
Co-segmentation [36]	Alexnet	Parts+Segmentation+Bbox	74.9
Zhang et al [78]	VGG-16	Parts+Keypoints+Compact Bilinear	85.9
Ensemble Models			
Subset Feature Learning [20]	Alexnet	Clustering+Ensemble Voting	77.5
Mixture of DCNN [19]	Alexnet	Clustering+Ensemble Voting	74.1
Multiple Granularity CNN [69]	VGG-16	Bbox+Part Detection+Ensemble	83.0
Bilinear CNN [44]	VGG-16	Ensemble	84.0
Attention Based Models			
Two Level Attention [70]	Alexnet	Parts+Two Level Attention	69.7
Spatial Transformer Network [32]	GoogleNet	Attention+Ensemble	84.1
Attribute-Guided Attention [46]	ResNet-50	Parts+Attributes+Attention	85.5
Fully Convolutional Attention [47]	GoogleNet	Attention+Reinforcement Learning	84.3
Diversified Visual Attention [82]	VGG-16	Attention+Reinforcement Learning	79.0

5.4.2 Baseline Models

To establish some baselines, we fine-tune each of the selected Imagenet pre-trained models mentioned earlier on each of the 4 tasks of order, family, genus and species classification individually, i.e. we take each pre-trained model, and finetune it 4 times separately for the 4 separate tasks, in order to establish baseline single-task accuracies for each task. While fine-tuning, we have an option of whether to add an additional hidden layer to the model or not. We denote the model without the hidden layer by STL, and the model with the extra hidden layer by $STL - H$, where STL is an abbreviation for single-task learning. Table 5.4 highlights the accuracies for each classification task for fine-tuning with and without an additional hidden layer. Expectedly, the accuracies improve as the models become deeper, from Alexnet to ResNets. The benefits from having an extra hidden layer also diminish as the model depth increases. This is explained by the fact that the larger models already have increased capacity to learn, and have reached a better optimum in the corresponding feature space, so an extra layer of parameters does not aid them with respect to this sort of fine-tuning to a new domain/application.

Table 5.4: Accuracies for each possible task by fine-tuning each pre-trained model independently with and without additional fine-tuning specific hidden layers. The tasks are order, family, genus, and species classification on the Caltech-UCSD Birds 200 dataset.

Task	Model / Hidden?	Alexnet	VGG16	ResNet34	ResNet50	ResNet101
Order	**STL**	90.87	95.15	96.35	96.77	97.44
	STL-H	91.68	96.04	96.60	96.98	97.91
Family	**STL**	72.49	82.71	86.74	90.81	91.71
	STL-H	77.09	88.31	89.85	91.73	92.19
Genus	**STL**	64.55	81.36	84.03	85.89	87.45
	STL-H	68.53	83.39	85.37	86.53	87.81
Species	**STL**	55.74	73.52	77.85	79.78	81.37
	STL-H	61.49	76.72	79.23	80.68	82.47

5.4.3 Multi-Task Models

Now that we have established some baseline numbers, we wish to observe the effect of converting these single-task models to multi-task ones. In this regard, we have quite a few variations that we need to compare, viz.

- Adding multiple auxiliary tasks as parallel branches
- Concatenating auxiliary tasks' features and passing them to final task
- Cascading each task's feature to each subsequent task till final task

Note that for all our experiments, we try architectures both with and without task-specific hidden layers. Additonally, we study the impact of dynamically updating the auxiliary task coefficients, and of task-wise early stopping, both independently and jointly. We use the aforementioned list of Imagenet pretrained models as the base shared layers of our multi-task models. Table 5.5 summarizes our results for the current set of experiments, in terms of fine-grained species recognition accuracies on the test set of the Caltech-UCSD Birds 200 dataset. As can be seen, almost all the multi-task models outperform their single-task variants from table 5.4. Dynamic task re-weighting, i.e. dynamically updating the auxiliary task coefficients leads to improvements over having fixed auxiliary task coefficients, as can be seen from the seconds row in each subsection of the table. On the other hand, simply using task-wise early stopping does not give any improvement in terms of performance. This can be explained by the fact that these tasks are all hierarchical or sequential in nature, and are arranged in increasing order of difficulty, i.e. order classification saturates much before family classification, while family classification reaches an optimum before genus classification, which in turn is still slightly simpler than species classification. Hence, even though technically stopping each task early enough should stop it form over-fitting, we are not gaining any additional test-time generalization capability in the final primary task of interest. Note that this sort of sequential early stopping for each task is quite similar to curriculum learning [3], and

remains a potential direction to explore further. From the rest of the table, we can see that task-wise weight updation along with task-wise early stopping performs even better. The explanation behind this is that once the earlier tasks start to overfit, they could be assigned higher coefficients. By enabling task-wise early stopping in conjunction, any task that starts to saturate or overfit is potentially stopped , and the coefficients are recomputed among the remaining tasks, such that the sum of the coefficients for the remaining auxiliary tasks sums upto 1. This effectively ensures that by the time a task is overfitting and has nothing left to contribute positively to the multi-task objective function that is being minimised, its contribution is reduced drastically and redistributed among the other auxiliary tasks, so that there is no negative transfer. Further inspection of the results shows us that concatenating features and cascading features also results in further improved performance for fine-grained recognition. We do not observe any significant differences in terms of performance between concatenation and cascade of auxiliary task features. One reason for the same might be the limited number of auxiliary tasks, and perhaps in a problem with more auxiliary tasks, there might be some differences among the two. This also remains a potential future research direction, especially for multi-label attribute prediction settings, where attributes can also be grouped.

Table 5.5: Accuracy for fine-grained recognition with the proposed variants. Please note that fine-grained species classification is given a constant weight $\alpha_{class} = 1$. MTL refers to simple multi-task learning, while $MTL - H$ implies that each task has its own private layer. Concatenating and cascading are applied as shown in figure 3.1. Dynamic task re-weighting refers to equation 3.2. Coefficients for auxiliary tasks, i.e. $\alpha_{order}, \alpha_{family}, \alpha_{genus}$ are set to 0.10.

Type	Additional Details	Alexnet	VGG16	ResNet34	ResNet50	ResNet101
Without Task-Specific Hidden Layer						
MTL		56.43	74.18	78.01	79.96	81.59
MTL	Dynamic Task Re-Weighting	56.92	74.57	78.19	80.15	81.67
MTL	Task-wise Early Stopping	55.31	73.61	77.73	79.81	81.29
MTL	Dynamic Task Re-Weighting + Task-wise Early Stopping	57.07	75.02	78.79	80.42	81.84
MTL-Concat	Dynamic Task Re-Weighting + Task-wise Early Stopping	57.38	75.26	78.94	80.79	81.97
With Task-Specific Hidden Layer						
MTL-H		61.98	77.23	79.41	80.97	82.71
MTL-H	Dynamic Task Re-Weighting	62.25	77.62	79.59	81.19	82.84
MTL-H	Task-wise Early Stopping	61.54	76.91	79.21	80.59	82.35
MTL-H	Dynamic Task Re-Weighting + Task-wise Early Stopping	62.47	77.85	78.07	81.53	82.97
MTL-H-Concat	Dynamic Task Re-Weighting + Task-wise Early Stopping	62.87	78.02	78.36	81.86	83.21
MTL-H-Cascade	Dynamic Task Re-Weighting + Task-wise Early Stopping	62.89	77.93	78.43	81.89	83.13

5.4.4 Combining with State-of-the-art Methods

Now that we have established that adding multiple tasks from a hierarchy of labels aids the process of fine-grained classification, it remains to see whether our proposed methods can play nice with other state-of-the-art methods. Most of the methods mentioned in 5.3 are quite complex in terms of implementation, and many are in fact multi-stage pipelines. While theoretically, we can combine our multi-task classifier on top of any model, for ease of implementation, we decided to pick one of the end-to-end models to verify whether we can combine our proposed variants with the said state-of-the-art model to achieve any further gains in fine-grained recognition. For the experiments in this section, we decide to use the bilinear CNN by Lin *et al*. [44]. A block diagram of their architecture is given in figure 2.7 in section 2. We describe how we obtain a small boost in accuracy by adapting bilinear CNN to a multi-task setting naively. We implement the $[M, M]$ variant of the bilinear CNN model, because it fits onto

Table 5.6: Accuracy on fine-grained classification with a single-task bilinear CNN and a multi-task bilinear CNN. We use the $[M, M]$ version of the model, since fitting the larger $[D, D]$ bilinear CNN variant into a single GPU requires hyperparameters that are different from the rest of our experiments. The MTL variant is a simple multi-task model, with no additional task-specific hidden layers.

Model	Additional Details	Accuracy
BCNN [M,M]	STL Variant	78.10
BCNN [M,M]	MTL Variant	79.04

a single GPU with the same batch-size as the rest of our experiments, i.e. 32. Using the larger $[D, D]$ model into a single GPU requires a smaller batch-size of 8, which in turn means many hyperparameters need to be tuned again for the multi-task variant [1] We convert the $[M, M]$ bilinear CNN to a multi-task variant by replacing the single classifier with our baseline MTL classifier, *i.e*. with 4 branches corresponding to the 4 tasks we intend to learn jointly. Note that the bilinear feature vector is quite large, almost $250,000$ dimensional, which is much larger than the 4096 dimensional feature vector obtained at the last shared layer for models such as Alexnet, VGG16 and the ResNets. As a consequence, we now have weight matrices of size $250,000 \times 15$, $250,000 \times 42$, $250,000 \times 120$ and $250,000 \times 200$ in the linear layers for order, family, genus and species classification. For this reason, not only is the STL variant of bilinear CNN quite memory intensive, but the MTL variant almost saturates the entire GPU memory. This remains an additional reason to prefer the $[M, M]$ variant over the $[D, D]$ variant for our experiments. Factorization of bilinear layers is an active area of research [18, 35], and reducing the feature dimensionality while capturing multilinear combinations from multiple tasks can be another potential research direction.

[1] Personal correspondence with the authors of [44] have revealed that in order to fit the $[D, D]$ model into a single GPU, hyperparameters such as the batch size, momentum, etc. needed to reduced drastically from the standard values used elsewhere. These reduced values would be completely different from the standard hyperparameter values we have been using for our other experiments. As a result, we opt not to re-implement the $[D, D]$ model since we cannot directly compare the results with our other experiments due to different hyperparameter settings.

5.4.5 Multi-Task as Regularization

Table 5.7: Table showing the improvement in terms of testing accuracy for MTL models over STL models for different tasks. The STL models were trained on each task separately, while the MTL model was trained on each task jointly.

Model	Type	Order	Family	Genus	Species
Alexnet	STL	8.946	24.533	34.332	43.119
	MTL	**7.956**	**21.027**	**31.157**	**42.574**
	STL-H	7.518	20.584	26.259	29.363
	MTL-H	**4.617**	**13.501**	**15.045**	**27.956**
VGG-16	STL	5.007	12.965	18.475	26.019
	MTL	**3.461**	**7.885**	**12.202**	**18.98**
	STL-H	3.971	11.483	15.729	22.482
	MTL-H	**2.947**	**9.364**	**13.797**	**21.48**
Resnet-34	STL	3.592	10.998	15.898	22.111
	MTL	**3.186**	**10.618**	**15.135**	**20.887**
	STL-H	3.317	9.698	14.563	18.695
	MTL-H	**2.109**	**6.446**	**9.033**	**17.917**
Resnet-50	STL	3.21	9.066	14.001	20.094
	MTL	**2.521**	**8.699**	**13.203**	**19.61**
	STL-H	2.97	8.313	12.94	18.297
	MTL-H	**2.116**	**7.12**	**11.382**	**18.086**
Resnet-101	STL	2.504	8.184	12.53	18.623
	MTL	**2.224**	**8.11**	**12.504**	**18.122**
	STL-H	2.272	7.755	12.769	16.786
	MTL-H	**1.751**	**6.367**	**10.255**	**15.936**

We try to substantiate the regularization effect of multi-task learning with proof that the multi-task models have indeed generalized better than their single-task counterparts. We address the claim of multi-task learning providing better regularization by analyzing the gap between the final training and testing accuracies for different models, as shown in table 5.7. In table 5.7, each row refers to a particular pretrained model, e.g. Alexnet or VGG-16 or Resnet-50, as well as a specific type, e.g. either a STL model, or a MTL-H model, i.e. a MTL model with task-specific hidden layers. For all the tasks, viz. the four levels of the class hierarchy, order, family, genus, and species, we show the difference between the final training and testing accuracy. This gap is representative of the generalization capabilities of the model. In other words, the higher this difference, the higher the probability that the model is overfitting, while a lower difference is proportional to lower overfitting. The entries in the STL and $STL-H$ rows denote the training accuracies obtained when training a single-task model for that task, while the entries in the MTL and $MTL-H$ rows denote the training accuracies obtained for each task in a multi-task model. Although we only display select variants, i.e. plain MTL and $MTL-H$, similar trends hold for other architectural variants, e.g. for concatenated and cascaded models, and for models with task-wise

dynamic coefficient updates and task-wise early stopping. Two distinct trends can be observed across all models:

- Adding task specific hidden layers improves generalization, in the form of lowered gap between final training and testing accuracies
- Multi-task learning significantly improves generalization as well, in the form of lowered gap between final training and testing accuracies

We posit that task specific hidden layers add more capacity to a pretrained network, letting it learn better if the new dataset on which it is being fine-tuned is different from the original dataset on which it was pre-trained. Furthermore, it is clear that auxiliary classification tasks improves the robustness of the model, improving the performance in terms of accuracy on not only the primary task, but on all auxiliary tasks as well. This clear improvement for all auxiliary tasks is possible only because of the hierarchically related nature of the classes, which ensures that there is not negative transfer between tangentially related auxiliary tasks. This is clear evidence that auxiliary tasks in the form of hierarchical super-classes aids in resolving confusing samples, pushing models towards better performance. As a side-effect of this type of training, our models can be used to predict super-classes when confidence in the fine-grained classes is low, since it is performing quite well on all levels of the hierarchy. This notion can possibly be extended to few-shot learning as well.

5.5 Overall Analysis and Chapter Summary

From table 5.5, it is evident that forcing a deep network to learn multiple tasks simultaneously results in increased testing accuracy on the primary task(s). The added regularization increases the generalization of the model. A closer look at the learning procedure reveals some more benefits of having multiple tasks. While performance from our multi-task models does not reach state of the art, it surpasses or provides close competition to [5,69,77,78] which rely on expensive additional annotations of keypoints and bounding boxes, and/or are multi-stage methods. In contrast, our model is an order of magnitude simpler, and can easily be utilised in other end-to-end fine grained recognition pipelines, such as [30,32,44]. Additionally, our results are purely in the weakest regime of evaluation, without any bounding boxes provided during either training or testing, unlike methods such as [36]. Supervision in the form of bounding boxes is bound to increase accuracy, and remains in the scope of future work.
From our results, it can be seen that recognition is aided by not only adding multiple related tasks, but by task coefficient re-weighting and smoothing as well. Further, it seems that concatenating and cascading tasks improves performance even further, courtesy the knowledge embedded in the manifolds of the auxiliary task branches. We further implement the $BCNN[M, M]$ method of [44], and replace the single classifier with our baseline MTL classifier, *i.e.* with 4 branches corresponding to the 4 tasks we intend to learn jointly. We show that combining a hierarchy based MTL classifier with the $BCNN$ gives a slight boost in accuracy. Compared to [44] that runs at 8 frames/sec, our model runs at 32

frames/sec while training.

We also successfully corroborate the claim of multi-task learning providing better regularization leading to models that have generalized better, by highlighting how the gap between training and testing accuracies has decreased, as a result of rising testing accuracies and dropping training accuracies. For fine-grained recognition, decreasing overfitting is extremely important, and multi-task learning via hierarchical classes helps combat overfitting.

Chapter 6

Conclusions

In this thesis, we have tried to tackle the challenging computer vision problem of fine-grained recognition from a novel perspective. Because of immense variations in illumination, pose, background, etc., coupled with subtle intra-class differences and striking inter-class similarities, fine-grained recognition is plagued by challenges, whence one man's meat might become another man's poison. Instead of relying on additional expensive and prohibitive annotations such as keypoints, parts, bounding boxes, segmentation maps, attributes, etc. which require domain expertise at the image level, we attempted to harness the relationships among different fine-grained labels. The implied relationships among subordinate-level fine-grained classes are captured quite well in the form of taxonomies (such as super-classes) and/or ontologies(such as attributes and/or factors). These free labels can be cheaply extracted almost automatically using minimal domain knowledge and human effort, i.e. domain expertise is required only at the level of classes, not at the level of individual images. The embedded latent knowledge in such inter-class relationships is what we set about to exploit in this thesis, by leveraging multi-task neural networks in order to classify an image at multiple levels of the hierarchy. The usage of a label hierarchy to generate auxiliary tasks for a multi-task model is an effective strategy for regularization of a learning algorithm.

We exploit the inter-dependency and relatedness among tasks to train deep convolutional networks that are more accurate and robust. We perform full-fledged ablation studies on CIFAR 100 and its 2 level hierarchy to analyze the effect of various hyperparameters specific to our problem, such as bifurcation level and auxiliary task coefficient. Furthermore, we experiment on large-scale, real-world, fine-grained datasets such as CUB-200-2011 with a 4 level hierarchy of classes to highlight the efficacy of our proposed methods even when used for finetuning and transfer learning from Imagenet pretrained models. Experiments with multiple models as the base network for shared layers have shown how our proposed variants are robust to the choice of architecture. Ablation studies on auxiliary task coefficients show how our proposed dynamic task coefficient update technique yields superior performance to manual tuning of hyperparameters. Our experiments show that even with limited training data, the presence of hints in the form of additional tasks and a joint objective function causes the convnet to learn meaningful features that generalize well on testing data. The regularization effect of multi-task learning is

unequivocally demonstrated in the form of a comprehensive reduction in the gap between the training and testing accuracies which implies lessened overfitting. Additionally, a multi-task neural network can learn multiple tasks faster than a single-task network (theoretically, N times faster for N tasks), and requires much less memory than multiple single-task networks as well. Multi-task networks should clearly be preferred from a computational cost point of view, as they can provide interpretability as well for their predictions.

Potential future directions for research along this thread can be along multiple orthogonal dimensions. One aspect involves scaling to many auxiliary tasks, such as in attribute prediction, with attribute grouping to form a hierarchy of attributes. Another angle revolves around combining features from multiple tasks in a smart yet efficient manner. Concatenation and cascading are just the tip of the iceberg, and improved methods could be built by extending bilinear layers for the multilinear scenario, along with compression of the feature vector for efficiency purposes. Instead of the simple heuristics used to dynamically update the task coefficients, more sophisticated methods to understand task relatedness and task performance are required. Attention based models could be leveraged to attend on each task and learn appropriate coefficients for each auxiliary task, combining them in such a manner so as to minimize the error on the primary task of interest. A further future topic of study could be flexible inference, where a multi-task model makes predictions at various levels of a class taxonomy depending on its confidence, and the goal is to balance exploration versus exploitation. We hope the work done in this thesis enables and aids future research in the promising topic of exploiting class hierarchies to aid fine-grained recognition.

Appendix A

Full Parsed Taxonomy for Caltech-UCSD Birds-200-2011 Dataset

Species / Class	Order	Family	Genus
001.Black_footed_Albatross	Procellariiformes	Diomedeidae	Phoebastria
002.Laysan_Albatross	Procellariiformes	Diomedeidae	Phoebastria
003.Sooty_Albatross	Procellariiformes	Diomedeidae	Phoebetria
004.Groove_billed_Ani	Cuculiformes	Cuculidae	Crotophaga
005.Crested_Auklet	Charadriiformes	Alcidae	Aethia
006.Least_Auklet	Charadriiformes	Alcidae	Aethia
007.Parakeet_Auklet	Charadriiformes	Alcidae	Aethia
008.Rhinoceros_Auklet	Charadriiformes	Alcidae	Cerorhinca
009.Brewer_Blackbird	Passeriformes	Icteridae	Euphagus
010.Red_winged_Blackbird	Passeriformes	Icteridae	Agelaius
011.Rusty_Blackbird	Passeriformes	Icteridae	Euphagus
012.Yellow_headed_Blackbird	Passeriformes	Icteridae	Xanthocephalus
013.Bobolink	Passeriformes	Icteridae	Dolichonyx
014.Indigo_Bunting	Passeriformes	Cardinalidae	Passerina
015.Lazuli_Bunting	Passeriformes	Cardinalidae	Passerina
016.Painted_Bunting	Passeriformes	Cardinalidae	Passerina
017.Cardinal	Passeriformes	Cardinalidae	Cardinalis
018.Spotted_Catbird	Passeriformes	Ptilonorhynchidae	Ailuroedus
019.Gray_Catbird	Passeriformes	Mimidae	Dumetella
020.Yellow_breasted_Chat	Passeriformes	Parulidae	Icteria
021.Eastern_Towhee	Passeriformes	Emberizidae	Pipilo
022.Chuck_will_Widow	Caprimulgiformes	Caprimulgidae	Antrostomus

023.Brandt_Cormorant	Suliformes	Phalacrocoracidae	Phalacrocorax
024.Red_faced_Cormorant	Suliformes	Phalacrocoracidae	Phalacrocorax
025.Pelagic_Cormorant	Suliformes	Phalacrocoracidae	Phalacrocorax
026.Bronzed_Cowbird	Passeriformes	Icteridae	Molothrus
027.Shiny_Cowbird	Passeriformes	Icteridae	Molothrus
028.Brown_Creeper	Passeriformes	Certhiidae	Certhia
029.American_Crow	Passeriformes	Corvidae	Corvus
030.Fish_Crow	Passeriformes	Corvidae	Corvus
031.Black_billed_Cuckoo	Cuculiformes	Cuculidae	Coccyzus
032.Mangrove_Cuckoo	Cuculiformes	Cuculidae	Coccyzus
033.Yellow_billed_Cuckoo	Cuculiformes	Cuculidae	Coccyzus
034.Gray_crowned_Rosy_Finch	Passeriformes	Fringillidae	Leucosticte
035.Purple_Finch	Passeriformes	Fringillidae	Haemorhous
036.Northern_Flicker	Piciformes	Picidae	Colaptes
037.Acadian_Flycatcher	Passeriformes	Tyrannidae	Empidonax
038.Great_Crested_Flycatcher	Passeriformes	Tyrannidae	Myiarchus
039.Least_Flycatcher	Passeriformes	Tyrannidae	Empidonax
040.Olive_sided_Flycatcher	Passeriformes	Tyrannidae	Contopus
041.Scissor_tailed_Flycatcher	Passeriformes	Tyrannidae	Tyrannus
042.Vermilion_Flycatcher	Passeriformes	Tyrannidae	Pyrocephalus
043.Yellow_bellied_Flycatcher	Passeriformes	Tyrannidae	Empidonax
044.Frigatebird	Suliformes	Fregatidae	Fregata
045.Northern_Fulmar	Procellariiformes	Procellariidae	Fulmarus
046.Gadwall	Anseriformes	Anatidae	Anas
047.American_Goldfinch	Passeriformes	Fringillidae	Spinus
048.European_Goldfinch	Passeriformes	Fringillidae	Carduelis
049.Boat_tailed_Grackle	Passeriformes	Icteridae	Quiscalus
050.Eared_Grebe	Podicipediformes	Podicipedidae	Podiceps
051.Horned_Grebe	Podicipediformes	Podicipedidae	Podiceps
052.Pied_billed_Grebe	Podicipediformes	Podicipedidae	Podilymbus
053.Western_Grebe	Podicipediformes	Podicipedidae	Aechmophorus
054.Blue_Grosbeak	Passeriformes	Cardinalidae	Passerina
055.Evening_Grosbeak	Passeriformes	Fringillidae	Coccothraustes

056.Pine_Grosbeak	Passeriformes	Fringillidae	Pinicola
057.Rose_breasted_Grosbeak	Passeriformes	Cardinalidae	Pheucticus
058.Pigeon_Guillemot	Charadriiformes	Alcidae	Cepphus
059.California_Gull	Charadriiformes	Laridae	Larus
060.Glaucous_winged_Gull	Charadriiformes	Laridae	Larus
061.Heermann_Gull	Charadriiformes	Laridae	Larus
062.Herring_Gull	Charadriiformes	Laridae	Larus
063.Ivory_Gull	Charadriiformes	Laridae	Pagophila
064.Ring_billed_Gull	Charadriiformes	Laridae	Larus
065.Slaty_backed_Gull	Charadriiformes	Laridae	Larus
066.Western_Gull	Charadriiformes	Laridae	Larus
067.Anna_Hummingbird	Apodiformes	Trochilidae	Calypte
068.Ruby_throated_Hummingbird	Apodiformes	Trochilidae	Archilochus
069.Rufous_Hummingbird	Apodiformes	Trochilidae	Selasphorus
070.Green_Violetear	Apodiformes	Trochilidae	Colibri
071.Long_tailed_Jaeger	Charadriiformes	Stercorariidae	Stercorarius
072.Pomarine_Jaeger	Charadriiformes	Stercorariidae	Stercorarius
073.Blue_Jay	Passeriformes	Corvidae	Cyanocitta
074.Florida_Jay	Passeriformes	Corvidae	Aphelocoma
075.Green_Jay	Passeriformes	Corvidae	Cyanocorax
076.Dark_eyed_Junco	Passeriformes	Emberizidae	Junco
077.Tropical_Kingbird	Passeriformes	Tyrannidae	Tyrannus
078.Gray_Kingbird	Passeriformes	Tyrannidae	Tyrannus
079.Belted_Kingfisher	Coraciiformes	Alcedinidae	Megaceryle
080.Green_Kingfisher	Coraciiformes	Alcedinidae	Chloroceryle
081.Pied_Kingfisher	Coraciiformes	Alcedinidae	Ceryle
082.Ringed_Kingfisher	Coraciiformes	Alcedinidae	Megaceryle
083.White_breasted_Kingfisher	Coraciiformes	Halcyonidae	Halcyon
084.Red_legged_Kittiwake	Charadriiformes	Laridae	Rissa
085.Horned_Lark	Passeriformes	Alaudidae	Eremophila
086.Pacific_Loon	Gaviiformes	Gaviidae	Gavia
087.Mallard	Anseriformes	Anatidae	Anas
088.Western_Meadowlark	Passeriformes	Icteridae	Sturnella

089.Hooded_Merganser	Anseriformes	Anatidae	Lophodytes
090.Red_breasted_Merganser	Anseriformes	Anatidae	Mergus
091.Mockingbird	Passeriformes	Mimidae	Mimus
092.Nighthawk	Caprimulgiformes	Caprimulgidae	Chordeiles
093.Clark_Nutcracker	Passeriformes	Corvidae	Nucifraga
094.White_breasted_Nuthatch	Passeriformes	Sittidae	Sitta
095.Baltimore_Oriole	Passeriformes	Icteridae	Icterus
096.Hooded_Oriole	Passeriformes	Icteridae	Icterus
097.Orchard_Oriole	Passeriformes	Icteridae	Icterus
098.Scott_Oriole	Passeriformes	Icteridae	Icterus
099.Ovenbird	Passeriformes	Parulidae	Seiurus
100.Brown_Pelican	Pelecaniformes	Pelecanidae	Pelecanus
101.White_Pelican	Pelecaniformes	Pelecanidae	Pelecanus
102.Western_Wood_Pewee	Passeriformes	Tyrannidae	Contopus
103.Sayornis	Passeriformes	Tyrannidae	Sayornis
104.American_Pipit	Passeriformes	Motacillidae	Anthus
105.Whip_poor_Will	Caprimulgiformes	Caprimulgidae	Antrostomus
106.Horned_Puffin	Charadriiformes	Alcidae	Fratercula
107.Common_Raven	Passeriformes	Corvidae	Corvus
108.White_necked_Raven	Passeriformes	Corvidae	Corvus
109.American_Redstart	Passeriformes	Parulidae	Setophaga
110.Geococcyx	Cuculiformes	Cuculidae	Geococcyx
111.Loggerhead_Shrike	Passeriformes	Laniidae	Lanius
112.Great_Grey_Shrike	Passeriformes	Laniidae	Lanius
113.Baird_Sparrow	Passeriformes	Emberizidae	Ammodramus
114.Black_throated_Sparrow	Passeriformes	Emberizidae	Amphispiza
115.Brewer_Sparrow	Passeriformes	Corvidae	Corvus
116.Chipping_Sparrow	Passeriformes	Emberizidae	Spizella
117.Clay_colored_Sparrow	Passeriformes	Emberizidae	Spizella
118.House_Sparrow	Passeriformes	Passeridae	Passer
119.Field_Sparrow	Passeriformes	Emberizidae	Spizella
120.Fox_Sparrow	Passeriformes	Emberizidae	Passerella
121.Grasshopper_Sparrow	Passeriformes	Emberizidae	Ammodramus

122.Harris_Sparrow	Passeriformes	Emberizidae	Zonotrichia
123.Henslow_Sparrow	Passeriformes	Emberizidae	Ammodramus
124.Le_Conte_Sparrow	Passeriformes	Emberizidae	Ammodramus
125.Lincoln_Sparrow	Passeriformes	Emberizidae	Melospiza
126.Nelson_Sharp_tailed_Sparrow	Passeriformes	Emberizidae	Ammodramus
127.Savannah_Sparrow	Passeriformes	Emberizidae	Passerculus
128.Seaside_Sparrow	Passeriformes	Emberizidae	Ammodramus
129.Song_Sparrow	Passeriformes	Emberizidae	Melospiza
130.Tree_Sparrow	Passeriformes	Emberizidae	Spizelloides
131.Vesper_Sparrow	Passeriformes	Emberizidae	Pooecetes
132.White_crowned_Sparrow	Passeriformes	Emberizidae	Zonotrichia
133.White_throated_Sparrow	Passeriformes	Emberizidae	Zonotrichia
134.Cape_Glossy_Starling	Passeriformes	Sturnidae	Lamprotornis
135.Bank_Swallow	Passeriformes	Hirundinidae	Riparia
136.Barn_Swallow	Passeriformes	Hirundinidae	Hirundo
137.Cliff_Swallow	Passeriformes	Hirundinidae	Petrochelidon
138.Tree_Swallow	Passeriformes	Hirundinidae	Tachycineta
139.Scarlet_Tanager	Passeriformes	Cardinalidae	Piranga
140.Summer_Tanager	Passeriformes	Cardinalidae	Piranga
141.Arctic_Tern	Charadriiformes	Laridae	Sterna
142.Black_Tern	Charadriiformes	Laridae	Chlidonias
143.Caspian_Tern	Charadriiformes	Laridae	Hydroprogne
144.Common_Tern	Charadriiformes	Laridae	Sterna
145.Elegant_Tern	Charadriiformes	Laridae	Thalasseus
146.Forsters_Tern	Charadriiformes	Laridae	Sterna
147.Least_Tern	Charadriiformes	Laridae	Sternula
148.Green_tailed_Towhee	Passeriformes	Emberizidae	Pipilo
149.Brown_Thrasher	Passeriformes	Mimidae	Toxostoma
150.Sage_Thrasher	Passeriformes	Mimidae	Oreoscoptes
151.Black_capped_Vireo	Passeriformes	Vireonidae	Vireo
152.Blue_headed_Vireo	Passeriformes	Vireonidae	Vireo
153.Philadelphia_Vireo	Passeriformes	Vireonidae	Vireo
154.Red_eyed_Vireo	Passeriformes	Vireonidae	Vireo

155.Warbling_Vireo	Passeriformes	Vireonidae	Vireo
156.White_eyed_Vireo	Passeriformes	Vireonidae	Vireo
157.Yellow_throated_Vireo	Passeriformes	Vireonidae	Vireo
158.Bay_breasted_Warbler	Passeriformes	Parulidae	Setophaga
159.Black_and_white_Warbler	Passeriformes	Parulidae	Mniotilta
160.Black_throated_Blue_Warbler	Passeriformes	Parulidae	Setophaga
161.Blue_winged_Warbler	Passeriformes	Parulidae	Vermivora
162.Canada_Warbler	Passeriformes	Parulidae	Cardellina
163.Cape_May_Warbler	Passeriformes	Parulidae	Setophaga
164.Cerulean_Warbler	Passeriformes	Parulidae	Setophaga
165.Chestnut_sided_Warbler	Passeriformes	Parulidae	Setophaga
166.Golden_winged_Warbler	Passeriformes	Parulidae	Vermivora
167.Hooded_Warbler	Passeriformes	Parulidae	Setophaga
168.Kentucky_Warbler	Passeriformes	Parulidae	Geothlypis
169.Magnolia_Warbler	Passeriformes	Parulidae	Setophaga
170.Mourning_Warbler	Passeriformes	Parulidae	Geothlypis
171.Myrtle_Warbler	Passeriformes	Parulidae	Setophaga
172.Nashville_Warbler	Passeriformes	Parulidae	Oreothlypis
173.Orange_crowned_Warbler	Passeriformes	Parulidae	Oreothlypis
174.Palm_Warbler	Passeriformes	Parulidae	Setophaga
175.Pine_Warbler	Passeriformes	Parulidae	Setophaga
176.Prairie_Warbler	Passeriformes	Parulidae	Setophaga
177.Prothonotary_Warbler	Passeriformes	Parulidae	Protonotaria
178.Swainson_Warbler	Passeriformes	Parul	Limnothlypis
179.Tennessee_Warbler	Passeriformes	Parulidae	Oreothlypis
180.Wilson_Warbler	Passeriformes	Parulidae	Cardellina
181.Worm_eating_Warbler	Passeriformes	Parulidae	Helmitheros
182.Yellow_Warbler	Passeriformes	Parulidae	Setophaga
183.Northern_Waterthrush	Passeriformes	Parulidae	Parkesia
184.Louisiana_Waterthrush	Passeriformes	Parulidae	Parkesia
185.Bohemian_Waxwing	Passeriformes	Bombycillidae	Bombycilla
186.Cedar_Waxwing	Passeriformes	Bombycillidae	Bombycilla
187.American_Three_toed_Woodpecker	Piciformes	Picidae	Picoides

188.Pileated_Woodpecker	Piciformes	Picidae	Dryocopus
189.Red_bellied_Woodpecker	Piciformes	Picidae	Melanerpes
190.Red_cockaded_Woodpecker	Piciformes	Picidae	Picoides
191.Red_headed_Woodpecker	Piciformes	Picidae	Melanerpes
192.Downy_Woodpecker	Piciformes	Picidae	Picoides
193.Bewick_Wren	Passeriformes	Troglodytidae	Thryomanes
194.Cactus_Wren	Passeriformes	Troglodytidae	Campylorhynchus
195.Carolina_Wren	Passeriformes	Troglodytidae	Thryothorus
196.House_Wren	Passeriformes	Troglodytidae	Troglodytes
197.Marsh_Wren	Passeriformes	Troglodytidae	Cistothorus
198.Rock_Wren	Passeriformes	Troglodytidae	Salpinctes
199.Winter_Wren	Passeriformes	Troglodytidae	Troglodytes
200.Common_Yellowthroat	Passeriformes	Parulidae	Geothlypis

Table A.1: The order, family, genus and species for each class name in the Caltech-UCSD Birds-200-2011 dataset [68], obtained by parsing the American Ornithologists' Union Checklist of North American Birds [9].

Related Publications

- Riddhiman Dasgupta, Anoop Namboodiri. **Leveraging Multiple Tasks To Regularize Fine-grained Classification**. *The 23rd International Conference on Pattern Recognition (ICPR 2016)*, Cancun, Mexico. [Accepted]

Bibliography

[1] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia. Multi-task cnn model for attribute prediction. *IEEE Transactions on Multimedia*, 2015.

[2] Y. S. Abu-Mostafa. Learning from hints in neural networks. *Journal of complexity*, 1990.

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International Conference On Machine Learning*, 2009.

[4] S. Branson, O. Beijbom, and S. Belongie. Efficient large-scale structured learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[5] S. Branson, G. Van Horn, S. Belongie, and P. Perona. Bird species categorization using pose normalized deep convolutional nets. In *British Machine Vision Conference*, 2015.

[6] R. Caruana. Multitask learning. *Machine learning*, 1997.

[7] R. Caruana. A dozen tricks with multitask learning. *Neural Networks: Tricks Of The Trade*, 1998.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[9] R. T. Chesser, R. C. Banks, K. J. Burns, C. Cicero, J. L. Dunn, A. W. Kratter, I. J. Lovette, A. G. Navarro-Sigüenza, P. C. Rasmussen, J. Remsen Jr, et al. Fifty-fifth supplement to the american ornithologists' union check-list of north american birds. *The Auk*, 2015.

[10] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[11] X. Chu, W. Ouyang, W. Yang, and X. Wang. Multi-task recurrent neural network for immediacy prediction. In *IEEE International Conference on Computer Vision*, 2015.

[12] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[13] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, 2008.

[14] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, 2014.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[17] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision*, 2015.

[18] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[19] Z. Ge, A. Bewley, C. McCool, P. Corke, B. Upcroft, and C. Sanderson. Fine-grained classification via mixture of deep convolutional neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, 2016.

[20] Z. Ge, C. McCool, C. Sanderson, and P. Corke. Subset feature learning for fine-grained category classification. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015.

[21] R. Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, 2015.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[23] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014.

[24] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[25] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. In *NIPS Workshop on Continual Learning and Deep Networks*, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016.

[28] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 201y.

[29] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016.

[30] S. Huang, Z. Xu, D. Tao, and Y. Zhang. Part-stacked cnn for fine-grained visual categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[32] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Neural Information Processing Systems*, 2015.

[33] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 2017.

[34] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *CVPR Workshop on Fine-Grained Visual Categorization*, 2011.

[35] S. Kong and C. Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[36] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[37] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[38] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 and cifar-100 datasets, 2009. *URL: https://www. cs. toronto. edu/˜ kriz/cifar. html*, 2009.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012.

[40] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *International Conference on Artificial Intelligence and Statistics*, 2014.

[41] S. Li, Z.-Q. Liu, and A. B. Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[42] D. Lin, X. Shen, C. Lu, and J. Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[43] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations*, 2014.

[44] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *IEEE International Conference on Computer Vision*, 2015.

[45] C. Linnaeus. Systema naturae per regna tria naturae. tomus i. editio decima, reformata. *Holmiae, Laurentii Salvii*, 1758.

[46] X. Liu, J. Wang, S. Wen, E. Ding, and Y. Lin. Localizing by describing: Attribute-guided attention localization for fine-grained recognition. In *AAAI Conference on Artificial Intelligence*, 2017.

[47] X. Liu, T. Xia, J. Wang, Y. Yang, F. Zhou, and Y. Lin. Fully convolutional attention networks for fine-grained recognition. *arXiv preprint arXiv:1603.06765*, 2016.

[48] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[49] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[50] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[51] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv preprint arXiv:1603.01249*, 2016.

[52] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshops*, 2014.

[53] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[54] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*, 2017.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter Learning Internal Representations by Error Propagation. 1986.

[56] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[57] P. Sermanet, A. Frome, and E. Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[58] D. L. Silver and R. E. Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science*, 1996.

[59] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[60] J. T. Springenberg, A. Dosovitsky, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*, 2015.

[61] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[62] N. Srivastava and R. R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Neural Information Processing Systems*, 2013.

[63] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Neural Information Processing Systems*, 2015.

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Ieee Conference On Computer Vision And Pattern Recognition*, 2015.

[65] P. Teterwak and L. Torresani. Shared roots: Regularizing deep neural networks through multitask learning. Master's thesis, Dartmouth College, 2014.

[66] Y. Tian, P. Luo, X. Wang, and X. Tang. Pedestrian detection aided by deep learning semantic tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[67] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2008.

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

[69] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang. Multiple granularity descriptors for fine-grained categorization. In *IEEE International Conference on Computer Vision*, 2015.

[70] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[71] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[72] 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 *IEEE International Conference on Computer Vision*, 2015.

[73] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[74] A. R. Zamir, T.-L. Wu, L. Sun, W. Shen, J. Malik, and S. Savarese. Feedback networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[75] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[76] C. Zhang and Z. Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[77] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, 2014.

[78] N. Zhang, E. Shelhamer, Y. Gao, and T. Darrell. Fine-grained pose prediction, normalization, and recognition. *arXiv preprint arXiv:1511.07063*, 2015.

[79] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, 2014.

[80] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[81] B. Zhao, J. Feng, X. Wu, and S. Yan. A survey on deep learning-based fine-grained object classification and semantic segmentation. *International Journal of Automation and Computing*, 2017.

[82] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Transactions on Multimedia*, 2017.