Hierarchical Classification

Author: Ayush Kumar Tiwary

Introduction

A hierarchical classifier is a classifier that maps input data into defined subsumptive output categories. The classification occurs first on a low-level with highly specific pieces of input data. The classifications of the individual pieces of data are then combined systematically and classified on a higher level iteratively until one output is produced. This final output is the overall classification of the data. Depending on application-specific details, this output can be one of a set of pre-defined outputs, one of a set of on-line learned outputs, or even a new novel classification that hasn't been seen before. Generally, such systems rely on relatively simple individual units of the hierarchy that have only one universal function to do the classification. In a sense, these machines rely on the power of the hierarchical structure itself instead of the computational abilities of the individual components. This makes them relatively simple, easily expandable, and very powerful.

Research

1. Branched CNN

A B-CNN model outputs multiple predictions ordered from coarse to fine along the concatenated convolutional layers corresponding to the hierarchical structure of the target classes, which can be regarded as a form of prior knowledge on the output. To learn with B-CNN's a novel training strategy, named the Branch Training strategy (BT-strategy), is introduced which balances the strictness of the prior with the freedom to adjust parameters on the output layers to minimize the loss.

Architecture of Branch Convolutional Neural Network (B-CNN)

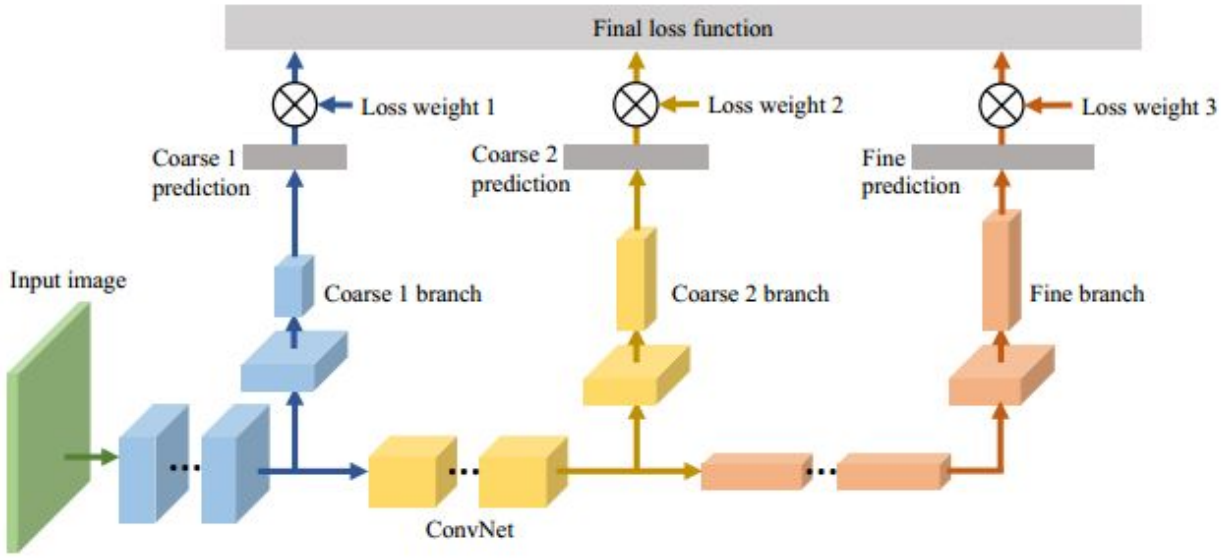

Figure 1: Architecture of Branch Convolutional Neural Network (B-CNN). The network at the bottom can be an arbitrary ConvNet. There can be multiple branch networks and each of them outputs a coarse prediction. The final loss function is a weighted summation of all coarse losses.

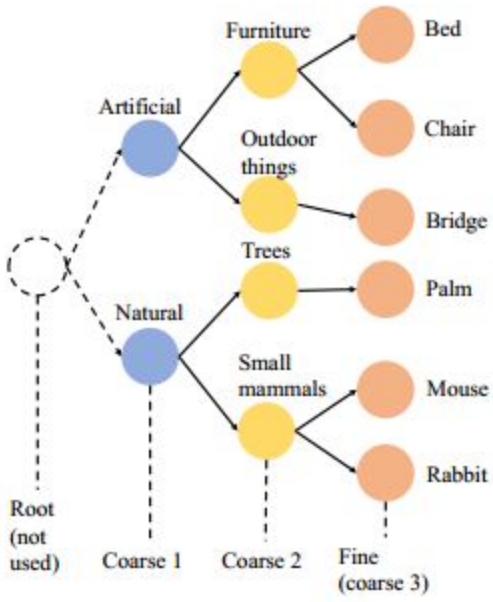

Figure 2: A sample hierarchical label tree where classes are taken from CIFAR-100 dataset.

A B-CNN model uses existent CNN components as building blocks to construct a network with internal output branches. The network shown at the bottom in Figure 1 is a traditional convolutional neural network. It can be an arbitrary ConvNet with multiple layers. The middle part in Figure 1 shows the output branch networks of a B-CNN. Each branch net produces a prediction on the corresponding level in the label tree (Figure 2, shown in same colour). On the top of each branch, fully connected layers and a softmax layer are used to produce the output in one-hot representation. Branch nets can consist of ConvNets and fully connected neural networks. But for simplicity, in our experiments, we only use fully connected neural networks as our branch nets.

When doing classification, a B-CNN model output as many predictions as the levels the corresponding label tree has. For example, considering the label tree shown in Figure 2, an image of a mouse will contain a hierarchical label of [natural, small mammals, mouse]. When the image is fed into B-CNN, the network will output three

corresponding predictions as the data flow through and each level's loss will contribute to the final loss function base on the loss weights distribution.

Branch Training Strategy

Branch Training strategy (BT-strategy) exploits the potential of loss weights distribution to achieve an end-to-end training procedure with low impact of vanishing gradient problem [11]. BT-strategy modifies the loss weights distribution while training a B-CNN model, e.g., for a two-level classification, the initial loss weights can be assigned as [0.9, 0.1], and then they can be changed to [0.2, 0.8] after 50 epochs.

The greatest value in a loss weights assignment can be regarded as a "focus", e.g., 0.5 is the focus of [0.2, 0.3, 0.5]. A "focus" is not necessary in a distribution as all levels can be equally important. However, in our implementation, we usually set a "focus" to explicitly tell the classifier to learn this level with more efforts. Usually, the "focus" of a distribution will shift from lower level to higher level (from coarse to fine). This procedure requires the classifier to extract lower features first with coarse instructions and fine tune parameters with fine instructions later. It to an extent prevents the vanishing gradient problem which would make the updates to parameters on lower layers very difficult when the network is very deep.

Baseline Configuration

Base A	Base B	Base C
$28 \times 28 \times 1$ image	$32 \times 32 \times 3$ image	
conv3-32	$(\text{conv3-64})_{\times 2}$	$(\text{conv3-64})_{\times 2}$
maxpool-2 *	maxpool-2	maxpool-2
conv3-64	$(\text{conv3-128})_{\times 2}$	$(\text{conv3-128})_{\times 2}$
	maxpool-2 *	maxpool-2
conv3-64	$(\text{conv3-256})_{\times 2}$	$(\text{conv3-256})_{\times 3}$
	maxpool-2 **	maxpool-2 *
	$(\text{conv3-512})_{\times 2}$	$(\text{conv3-512})_{\times 3}$
maxpool-2	maxpool-2	maxpool-2 **
		$(\text{conv3-512})_{\times 3}$
Flatten		
FC-128	FC-1024	FC-4096
FC-10	FC-1024	FC-4096
	FC-x	FC-x
softmax layer		

*Figure 3:Baseline networks. Base A model is tested on MNIST dataset. Both B and C are tested on CIFAR-10 and CIFAR-100. The x in last ConvNet layer of Base B and C can be replaced by 10 or 100. Base C model is VGG16 [27] without last max-pooling layer because images in CIFAR datasets are very small. Symbol * means there is a branch network attached to that layer for corresponding B-CNN models, and the specific branch configurations are shown in figure 4.*

Branches of A	Branches of B	Branches of C
* Flatten		
FC-64	FC-256	FC-512
FC-c_{A1}	FC-256	FC-512
	FC-c_{B1}	FC-c_{C1}
	** Flatten	
	FC-512	FC-1024
	FC-512	FC-1024
	FC-c_{B2}	FC-c_{C2}

*Figure 4: Branch networks for each B-CNN model. Symbol * means this branch is connected to the layer with the same * in figure 3, e.g., there is a branch of network [Flatten, FC-64, FC-cA1] after the first maxpooling layer in B-CNN A model.*

Models	MNIST	CIFAR-10	CIFAR-100
Base A	99.27%	-	-
B-CNN A	**99.40%**	-	-
Base B	-	82.35%	51.00%
B-CNN B	-	**84.41%**	**57.59%**
Base C	-	87.96%	62.92%
B-CNN C	-	**88.22%**	**64.42%**

Figure 5: Performance of each model on MNIST, CIFAR-10 and CIFAR-100 test sets.

2. Visual Tree CNN

Constructing the VT-CNN model contains two main steps. The first step is constructing a visual tree for image categories. Inspired by the idea of confusion graph in CNN models from, we propose to use the community hierarchical detection algorithm to construct the visual tree, called Confusion Visual Tree(CVT). With this method, we can construct the visual tree structure automatically and fully utilize the information from the output of the CNN models. Then we embed this tree structure into the CNN model. Borrowing the idea from, we divide the structure of the VT-CNN model into two parts. The one is the base layers and the other is the branch layers. Branch layers have several branches and each branch contains a series of layers such as convolutional layers and fully-connected(FC) layers. All of these branches share the base layers. The number of the branch layers is equal to the number of levels in the tree model without the root level, which means each branch refers to a layer-grained categories classification.

Architecture of VT CNN

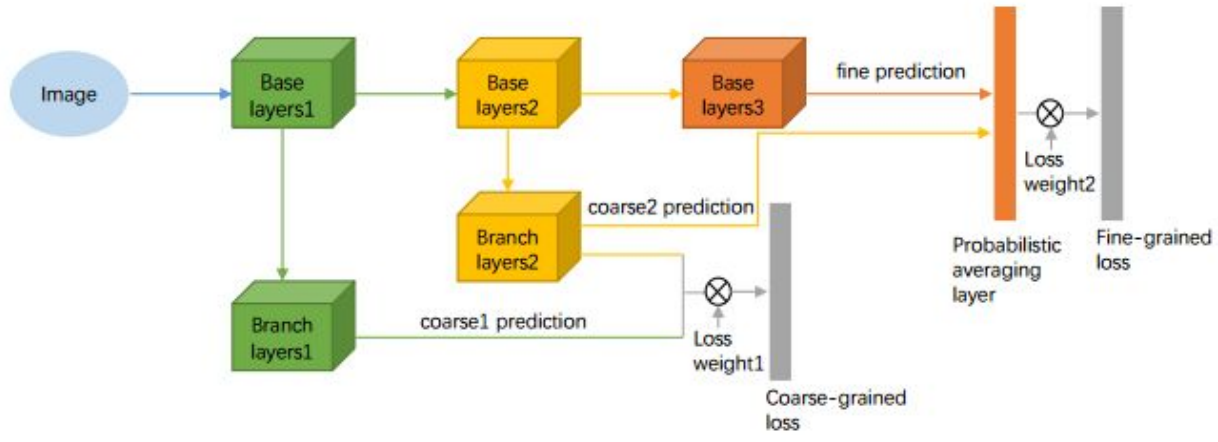

Figure 6: A four-level Confusion Visual Tree where the classes are taken from CIFAR-100 dataset.

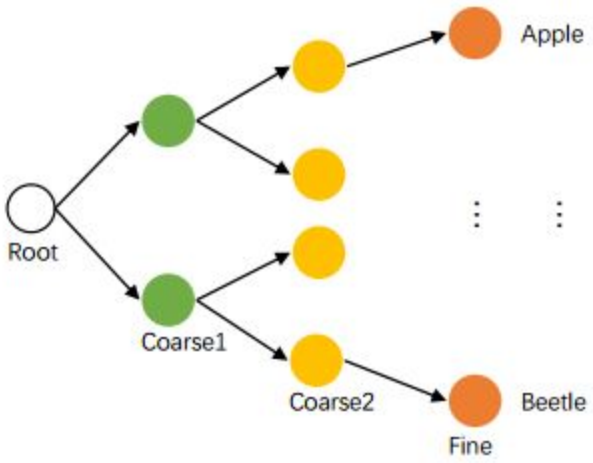

Figure 7: Visual Tree Convolutional Neural Network(VT-CNN) architecture.

The Branch Convolutional Neural Network(B-CNN) model proposed, outputs multiple predictions ordered from coarse-grained to fine-grained along the concatenated convolutional layers corresponding to the hierarchical structure of the target categories, which can be regarded as a form of prior knowledge on the output.

VT-CNN focuses on the fine-grained categories in same communities instead of treating all the categories equally, which is different to original CNN models.

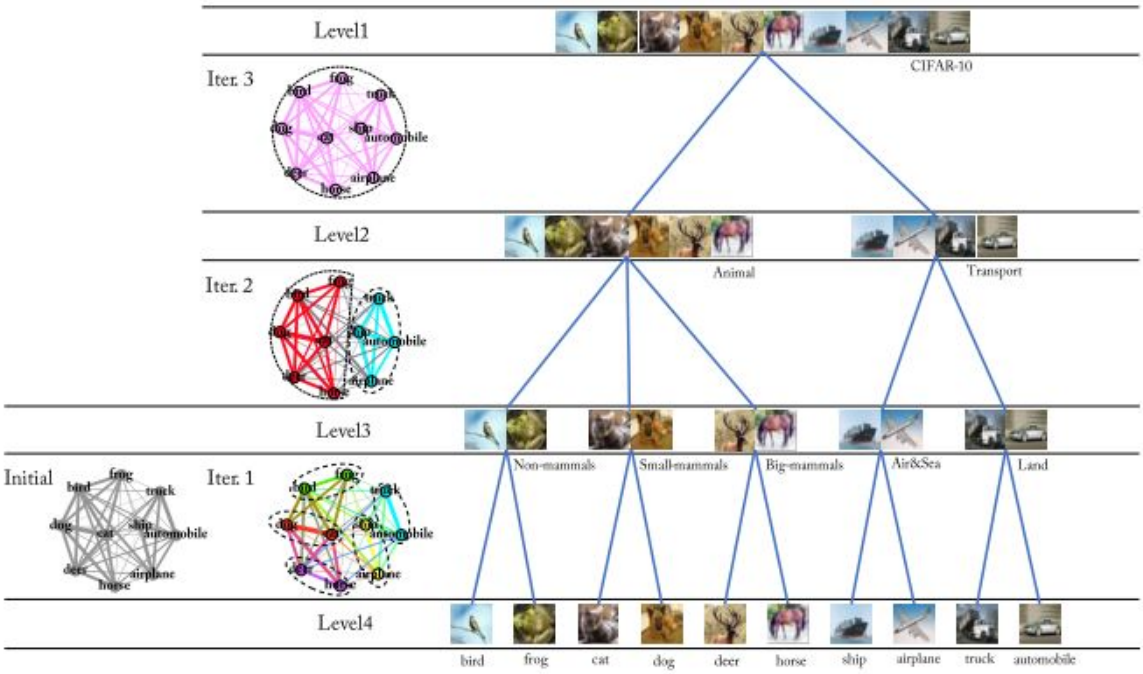

Figure 8: The construction process of confusion visual tree for CIFAR-10 image set.

Baseline Configuration

Layers of Base A	Layers of Base B
(Conv3-98)-MaxPool	$(\text{Conv3-64})_{\times 2}$-MaxPool
(Conv3-256)-MaxPool	$(\text{Conv3-128})_{\times 2}$-MaxPool
(Conv3-384)	$(\text{Conv3-256})_{\times 3}$-MaxPool
conv-3 Flatten	
$(\text{Conv3-128})_{\times 2}$	**$(\text{Conv3-256})_{\times 2}$**
FC-256	**FC-512**
FC-256	**FC-512**
FC-C_{A1}	**FC-C_{B1}**
(Conv3-384)	$(\text{Conv3-512})_{\times 3}$-MaxPool
conv-4 Flatten	
$(\text{Conv3-256})_{\times 2}$	**$(\text{Conv3-512})_{\times 2}$**
FC-512	**FC-1024**
FC-512	**FC-1024**
FC-C_{A2}	**FC-C_{B2}**
(Conv3-256)-MaxPool	$(\text{Conv3-512})_{\times 3}$-MaxPool
conv-5 Flatten	
Probabilistic averaging layer	

Figure 9: The entire architecture for each VT-CNN model. The id number in symbol conv-id denotes this part of branch architecture is connected with the idth convolution layers. Note that the branch layers are illustrated in bold fonts.

Models	Top-1 Accuracy
Base A	82.94%
B-CNN A	84.70%
VT-CNN A	**85.07%**
Base B	87.15%
B-CNN B	88.23%
VT-CNN B	**89.51%**

Figure 10: Performance of each model on CIFAR-10 test set.

Models	Top-1 Accuracy
AlexNet A	57.37%
B-CNN A	58.27%
VT-CNN A	**58.73%**
VGG16 B	71.15%
B-CNN B	71.23%
VT-CNN B	**72.04%**
ResNet-56 C	73.49%
B-CNN C	73.86%
VT-CNN C	**74.13%**

Figure 11:Performance of each model on CIFAR-100 test set.

3. CNN - RNN

In this paper, it is proposed to utilize the CNN-RNN framework to address the hierarchical image classification task. CNN allows us to obtain discriminative features for the input images, and RNN enables us to jointly optimize the classification of coarse and fine labels. This framework can not only generate hierarchical labels for images but also improve the traditional leaf-level classification performance due to incorporating the hierarchical information. Experimental results demonstrate that CNN-RNN can use the coarse-labelled training data to improve the classification of fine categories, and in some cases it even surpasses the performance achieved by fully annotated training data. This reveals that, CNN-RNN can alleviate the challenge of specialized and expensive annotation of fine labels.

The first contribution of this paper is a framework capable of generating hierarchical labels, by integrating the powerful Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNN is used to generate discriminative features, and RNN is used to generate sequential labels.

In this paper, the author has proposed to utilize the cascaded CNN-RNN framework to address a new task, i.e. hierarchical image classification, where we utilize CNN to generate discriminative image features and utilize RNN to model the sequential relationship of hierarchical labels.

Architecture of CNN-RNN

A CNN-RNN generator determines hierarchical predictions using an architecture where the last layer of a CNN is replaced by an RNN.

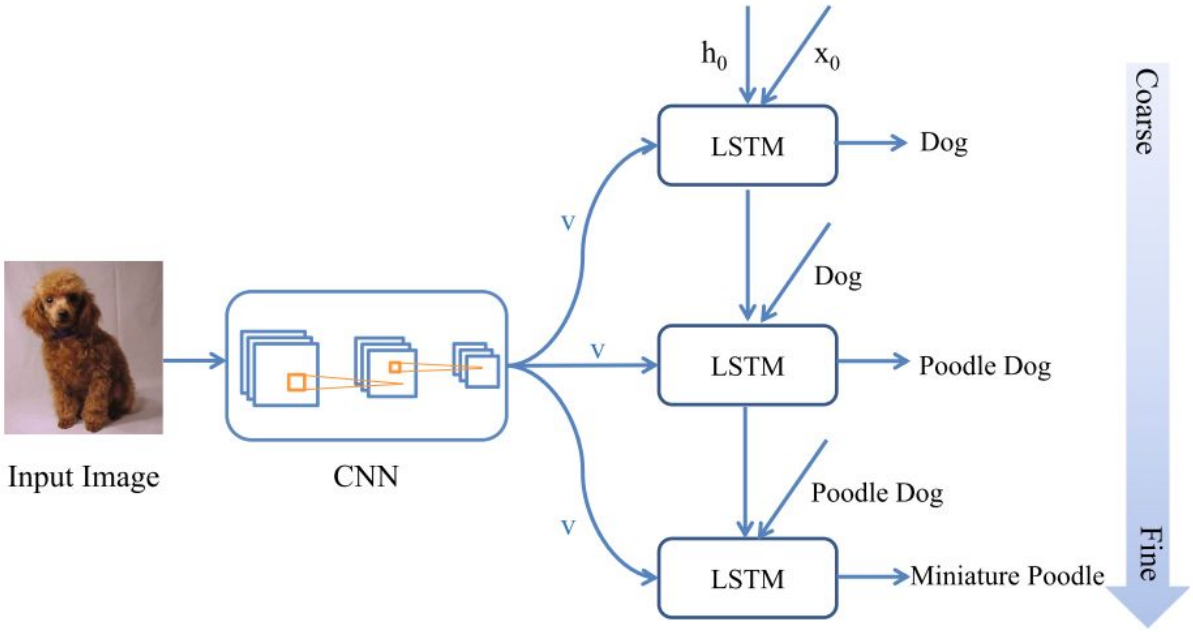

Figure 12: The pipeline for CNN-RNN framework.

As the CNN-RNN generator defines the super categories, and it equally trains and predicts the super categories, we do not need to design specific networks for the categories of different levels. Therefore, the CNN-RNN generator is robust and can be employed to generate hierarchical labels of different lengths.

Baseline Configuration

The CNN-RNN generator can effectively exploit the dependency of the hierarchical labels, and thereby achieving a better classification performance for both the coarse and fine categories.

	C100		C100+	
	Coarse	Fine	Coarse	Fine
Coarse-to-fine	73.88%	58.41%	78.1%	64.16%
Fine-to-coarse	75.02%	61.75%	78.16%	65.56%
Fine-and-coarse	74.72%	61.8%	77.56%	64.87%
CNN-RNN	**80.81%**	**69.69%**	**83.21%**	**72.26%**

'+' indicates a standard data augmentation (translation/mirroring)

Figure 13: The comparison of the accuracy for the coarse categories and fine categories.

The structure of our proposed wider-Resnet is shown in figure 14. We adopt the preactivation residual block, and train the models for a total of 7×104 iterations, with a mini-batch size of 200, a weight decay of 0.0005 and a momentum of 0.9. The learning rate is initialized with 0.1, and is dropped by 0.1 at 4 × 104 and 6 × 104 iterations.

Group name	Output size	wider-Resnet
conv1	32×32	3×3, 64
conv2	32×32	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	16×16	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 512 \end{bmatrix} \times 3$
conv4	8×8	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024 \\ 1\times1, 1024 \end{bmatrix} \times 3$
pool5	1×1	global average pooling

Figure 14: The framework of wider-Resnet.

		C100+	
		Coarse	Fine
CNN-7 [25]	coarse-specific	82.09%	–
	fine-specific	–	72.03%
	CNN-RNN	**83.21%**	**72.26%**
wrn-28-10 [53]	coarse-specific	82.59%	–
	fine-specific	–	74.55%
	CNN-RNN	**85.39%**	**76.23%**
wider-Resnet	coarse-specific	85.38%	–
	fine-specific	–	77.97%
	CNN-RNN	**88.23%**	**79.14%**

'+' indicates standard data augmentation (translation/mirroring)

Figure 15: The comparison of the accuracy for the coarse categories and fine categories. For each network, CNN-RNN could get better results.

4. H-CNN

In this paper, we reflect hierarchical structure of apparel on image classification algorithm. We name the algorithm as Hierarchical Convolutional Neural Networks (H-CNN). We implement H-CNN on VGGNet using Fashion-MNIST dataset. The objective of this paper is, first, to propose an approach of image classification reflecting hierarchical structure on fashion image where hierarchy of each fashion item determines its category label, and, second, to prove that using H-CNN model eases the inference of classification results and gives better results in classifying apparel images than base CNN model.

The proposed model, Hierarchical Deep Convolutional Neural Networks (HD-CNN), will first use an initial coarse classifier CNN to separate the easily separable classes, which denoted as coarse classes, and later the fine classes. HD-CNN is trained by a multinomial logistic loss and a novel temporal sparsity penalty. During HD-CNN training, component-wise pretraining is done, and then global fine-tuning with a multinomial logistic loss regularized by a coarse category consistency term is done. The next study of HD-CNN (Yan et al., 2015b) builds up three different HD-CNN models to compare the performance of HD-CNN with various layer combinations.

Architecture of H-CNN

In VGG16 H-CNN model, the first and second building blocks consist of 2 convolutional layers and 1 pooling layer, and the third and fourth blocks consist of 3 convolutional layers and 1 pooling layer, and the fifth building block has 3 convolutional layers. In VGG19 HCNN model, the first and second building blocks have 2 convolutional layers and 1 pooling layer, and the third

and fourth blocks have 4 convolutional layers and 1 pooling layer, and the fifth building block consists of 4 convolutional layers. However, this model has additional blocks below the five building blocks, we put three additional blocks followed by each prediction block outputting three levels of classified labels, which makes difference with

the base models. The first underneath branch denotes the first coarse-level block, the second branch for the second coarse-level block, and the last branch for the fine-level block. All 3 additional blocks are composed of fully-connected neural networks. As input image goes through the H-CNN model, three prediction values of coarse 1 level, coarse 2 level, and fine level will be computed in the order. For example, when an input image of sweater is inserted, the first coarse level branch will indicate 'clothes', the second coarse level branch will indicate 'tops', and the final branch will indicate 'pullover' as output predictions. H-CNN models use 3x3 sized filters with the stride of 1 in all convolutional layers and follow same number of filters for each building block with the base models. In the pooling layers, 2x2 sized max-pooling is performed by the stride of 2. These H-CNN models also use ReLU for activation function, batch normalization for initialization, and dropout for regularization. In the final branch denoted as the fine prediction block, softmax function is used to classify 10 fine classes.

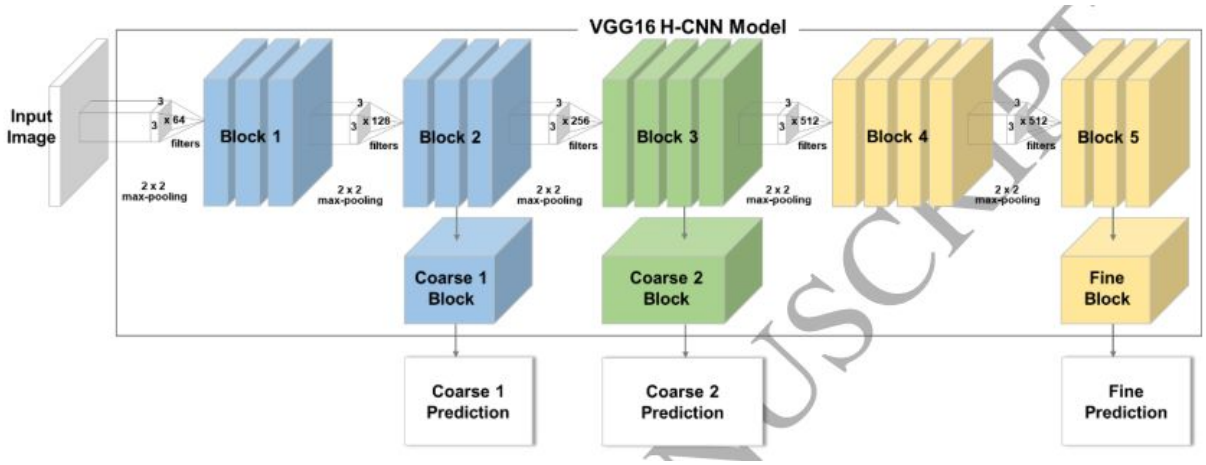

Figure 16: Architecture of VGG16 H-CNN model.

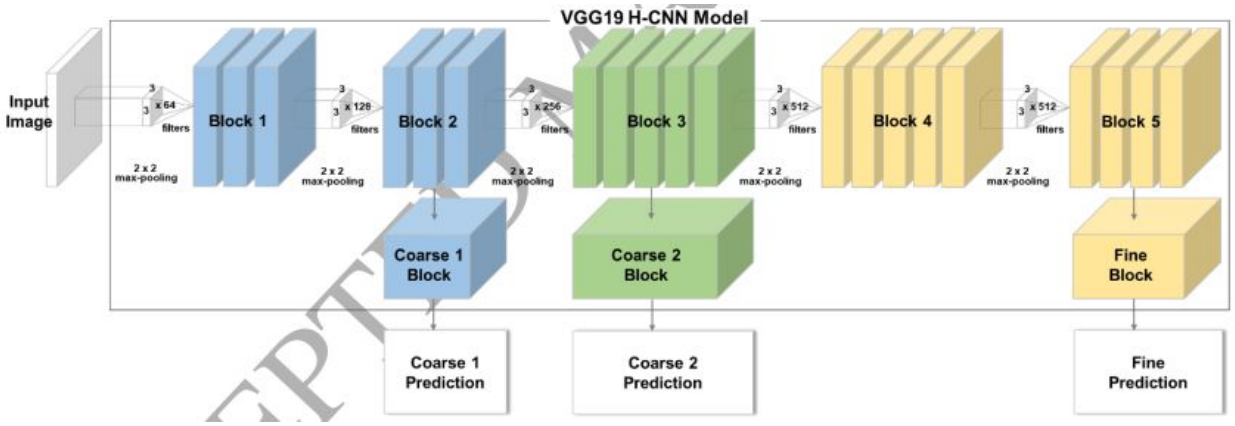

Figure 17: Architecture of VGG19 H-CNN model.

When training H-CNN models, same parameters are used as base models. The number of epoch is limited to 60 and the size of batch to 128. We set learning rates as 0.001 in initial stages, 0.0002 after 42th epoch, and 0.00005 after 52th epoch. Stochastic gradient descent is applied using 0.9 of momentum. However, loss weight values are added when training H-CNN models. To reflect the differences in the importance of each level of classes, different loss weights are applied on each level. In the initializing stage, the low-level feature extraction affects a lot on the result, so we assign higher value than the later stages. The changes in loss weights follow [0.98, 0.01, 0.01] in the first epoch, [0.10, 0.80, 0.10] in the 15th epoch, [0.1, 0.2, 0.7] in the 25th epoch, [0, 0, 1] epoch in the 35th epoch.

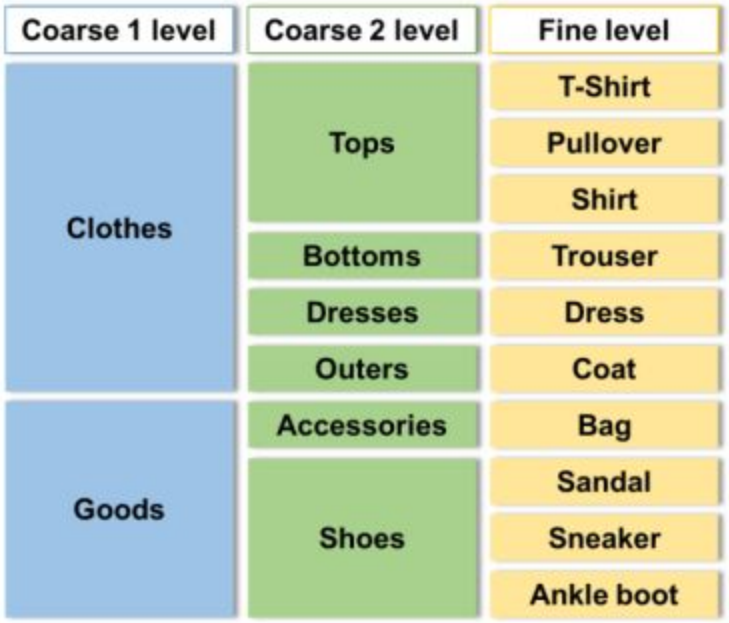

Figure 18: Hierarchical classes of dataset.

Methods	Accuracy	Base CNN Accuracy
BranchedCNN	MNIST - 99.40% CIFAR10 - 88.42% CIFAR100 - 64.42%	MNIST - 99.27% CIFAR10 - 87.96% CIFAR100 - 62.92%
Visual Tree CNN	CIFAR10 - 87.29% CIFAR100 - 68.30%	CIFAR10 - 85.04% CIFAR100 - 67.33%
CNN-RNN	CIFAR100 - 69.69%	CIFAR100 - 58.41%
Hierarchical CNN	Fashion MNIST - 93.52%	Fashion MNIST - 92.89%

Datasets

- [Fashion Landmark Detection in Wild Dataset]

  Fashion Landmark Detection Benchmark evaluates the performance of fashion landmark detection. This is a large subset of DeepFashion, with diverse and large pose/zoom-in variations. It contains

  - **123,016** number of **clothes images**;
  - **8 fashion landmarks (both location and visibility)** for each image;
  - Each image is also annotated by **bounding box**, **clothing type** and **variation type**.

- [Category and Attribute Prediction Benchmark Dataset]

  Category and Attribute Prediction Benchmark evaluates the performance of clothing category and attribute prediction. This is a large subset of DeepFashion, containing massive descriptive clothing categories and attributes in the wild. It contains

  - **289,222** number of **clothes images**;
  - **50** number of **clothing categories**, and **1,000** number of **clothing attributes**;
  - Each image is annotated by **bounding box** and **clothing type**.

  List_bbox.txt

  First Row: number of images

  Second Row: entry names

  Rest of the Rows: <image name> <bbox location>

  List_landmarks.txt

  - First Row: number of images

- Second Row: entry names
- Rest of the Rows: <image name> <clothes type> <variation type>[<landmark visibility 1> <landmark location x_1> <landmark location y_1> , ... <landmark visibility 8> <landmark location x_8> <landmark location y_8>].

Notes:

- The order of landmark labels accords with the order of entry names;
- In clothes type, "1" represents upper-body clothes, "2" represents lower-body clothes, "3" represents full-body clothes. Upper-body clothes possess six fahsion landmarks, lower-body clothes possess four fashion landmarks, full-body clothes possess eight fashion landmarks;
- In variation type, "1" represents normal pose, "2" represents medium pose, "3" represents large pose, "4" represents medium zoom-in, "5" represents large zoom-in;
- In landmark visibility state, "0" represents visible, "1" represents invisible/occluded, "2" represents truncated/cut-off;
- For upper-body clothes, landmark annotations are listed in the order of ["left collar", "right collar", "left sleeve", "right sleeve", "left hem", "right hem"]; For lower-body clothes, landmark annotations are listed in the order of ["left waistline", "right waistline", "left hem", "right hem"]; For upper-body clothes, landmark annotations are listed in the order of ["left collar", "right collar", "left sleeve", "right sleeve", "left waistline", "right waistline", "left hem", "right hem"].

List_category_cloth.txt

- First Row: number of categories
- Second Row: entry names
- Rest of the Rows: <category name> <category type>

List_category_img.txt

- First Row: number of images
- Second Row: entry names
- Rest of the Rows: <image name> <category label>

Notes:

- In category type, "1" represents upper-body clothes, "2" represents lower-body clothes, "3" represents full-body clothes;

- The order of category labels accords with the order of category names;
- In category labels, the number represents the category id in category names;
- For the clothing categories, "Cape", "Nightdress", "Shirtdress" and "Sundress" have been merged into "Dress";
- Category prediction is treated as a 1-of-K classification problem.

Deep Fashion

- **In Shop Clothes retrieval**

  In-shop Clothes Retrieval Benchmark evaluates the performance of in-shop Clothes Retrieval. This is a large subset of DeepFashion, containing large pose and scale variations. It also has large diversities, large quantities, and rich annotations, including

  - **7,982** number of **clothing items**;
  - **52,712** number of **in-shop clothes images**, and **~200,000 cross-pose/scale pairs**;
  - Each image is annotated by **bounding box**, **clothing type** and **pose type**.

- **CFPD | Colorful Fashion parsing Data**

  Details

  - 2,682 images
  - 600 x 400 (height, width)
  - pixel-level annotated (segmentation map)
    - 23 categories
    - 13 colors
  - make_label.py makes the followings from fashon_parsing_data.mat.

    - label/bbox.json: bounding box (for object detection not semantic segmentation).
    - label/categories.tsv
        - category_id
        - category
- label/main_categories.tsv is selected from categories.tsv for object detection.

- **Clothing Co-Parsing (CCP) Dataset**

  Clothing Co-Parsing (CCP) dataset is a new clothing database including elaborately annotated clothing items.

  - 2, 098 high-resolution street fashion photos with totally 59 tags
  - Wide range of styles, accessaries, garments, and pose
  - All images are with image-level annotations
  - 1000+ images are with pixel-level annotations

  Files

  Root directory contains following files and folders:

  - photos/ - directory of original photos
  - annotations/ - directory of annotations
    - pixel-level/ - pixel-level annotations (1004 files)
    - image-level/ - image-level annotations (1094 files)
  - show_pixel_anno.m - demo code for using pixel-level annotations
  - show_image_anno.m - demo code for using image-level annotations
  - label_list.mat - [1*59] cell array which maps label numbers to label names
  - samples.jpg - sample annotations

Implementation

Branched-CNN

- As per the requirement and dataset available, we implemented the concept of branched-CNN.

Experiment ID	Experiment Variable			Accuracy on test Dataset using B-CNN	
	Alpha	Beta	Epochs	H1 Layer	H2 Layer
1	• 0.7 • 0.3	• 0.3 • 0.7	• 1-12 • 12-60	92.76%	69.61%
2	• 0.7 • 0.3 • 0.0	• 0.3 • 0.7 • 1.0	• 1-12 • 12-42 • 42-60	93.76%	69.38%

- Hierarchical_model1.h5
  - h1_mapping = {1: 0, 2: 1, 3: 2}
  - h2_mapping = {'Sweatshorts': 5, 'Leggings': 3, 'Sweater': 9, 'Romper': 1, 'Hoodie': 7, 'Tank': 11, 'Cutoffs': 2, 'Coat': 0, 'Skirt': 6, 'Jeans': 4, 'Cardigan': 12, 'Tee': 8, 'Poncho': 10}
  - Accuracy

```
loss : 1.6753195054714496
h1_prediction_loss : 0.21557559072971344
h2_prediction_loss : 1.4828540086746216
h1_prediction_accuracy : 0.9276922941207886
h2_prediction_accuracy : 0.6961538195610046
```

  - Confusion Matrix for h1 layer

```
Confusion Matrix for h1 layer:
     1    2    3
1  538   19   43
2    4  492    4
3   20    4  176
```

➢ Classification Report for h1 layer

```
Classification Report for h1 layer:
             precision    recall  f1-score   support

          1       0.96      0.90      0.93       600
          2       0.96      0.98      0.97       500
          3       0.79      0.88      0.83       200

avg / total       0.93      0.93      0.93      1300
```

➢ Confusion Matrix for h2 layer

```
Confusion Matrix for h2 layer:
            Coat  Romper  Cutoffs  Leggings  Jeans  Sweatshorts  Skirt  Hoodie  Tee  Sweater  Poncho  Tank  Cardigan
Coat          85       3        0         0      0            0      2       1    0        2       0     0         7
Romper         2      91        0         0      1            0      0       0    0        1       2     2         1
Cutoffs        0       0       90         0      0            8      2       0    0        0       0     0         0
Leggings       0       0        4        73     13            7      2       1    0        0       0     0         0
Jeans          0       0        0         8     92            0      0       0    0        0       0     0         0
Sweatshorts    0       0        7        10     11           64      4       0    0        1       0     0         3
Skirt          0       1        3         1      1            1     88       0    0        0       0     3         2
Hoodie         2       1        0         0      0            0      0      57   12       15       2     3         8
Tee            1       0        0         0      0            0      0       5   79        2       1    12         0
Sweater        8       1        0         1      0            0      1      11    6       50       6     5        11
Poncho        11       3        0         0      1            0      1      10    8       21      17     1        27
Tank           1       3        0         0      0            0      0       4    1        3       0    85         3
Cardigan      13       4        1         1      0            1      3      11    7       13       8     4        34
```

➢ Classification Matrix for h2 layer

```
Classification Report for h2 layer:
             precision    recall  f1-score   support

       Coat       0.69      0.85      0.76       100
     Romper       0.85      0.91      0.88       100
    Cutoffs       0.86      0.90      0.88       100
   Leggings       0.78      0.73      0.75       100
      Jeans       0.77      0.92      0.84       100
Sweatshorts       0.79      0.64      0.71       100
      Skirt       0.85      0.88      0.87       100
     Hoodie       0.57      0.57      0.57       100
        Tee       0.70      0.79      0.74       100
    Sweater       0.46      0.50      0.48       100
     Poncho       0.47      0.17      0.25       100
       Tank       0.74      0.85      0.79       100
   Cardigan       0.35      0.34      0.35       100

avg / total       0.68      0.70      0.68      1300
```

- Hierarchical_model2.h5
  - h1_mapping = {1: 0, 2: 1, 3: 2}
  - h2_mapping = {'Sweatshorts': 5, 'Leggings': 3, 'Sweater': 9, 'Romper': 1, 'Hoodie': 7, 'Tank': 11, 'Cutoffs': 2, 'Coat': 0, 'Skirt': 6, 'Jeans': 4, 'Cardigan': 12, 'Tee': 8, 'Poncho': 10}
  - Accuracy

```
loss : 1.6753195274793184
h1_prediction_loss : 0.21557559072971344
h2_prediction_loss : 1.4828540086746216
h1_prediction_accuracy : 0.9276922941207886
h2_prediction_accuracy : 0.6961538195610046
```

➢ Confusion Matrix for h1 layer

```
Confusion Matrix for h1 layer:
     1    2    3
1  552   13   35
2    6  491    3
3   22    2  176
```

➢ Classification Report for h1 layer

```
Classification Report for h1 layer:
             precision    recall  f1-score   support

          1       0.95      0.92      0.94       600
          2       0.97      0.98      0.98       500
          3       0.82      0.88      0.85       200

avg / total       0.94      0.94      0.94      1300
```

➢ Confusion Matrix for h2 layer

Confusion Matrix for h2 layer:

	Coat	Romper	Cutoffs	Leggings	Jeans	Sweatshorts	Skirt	Hoodie	Tee	Sweater	Poncho	Tank	Cardigan
Coat	84	4	0	0	0	0	1	0	0	4	0	0	7
Romper	1	93	0	0	1	0	1	0	0	1	1	1	1
Cutoffs	0	0	90	1	0	5	3	0	0	1	0	0	0
Leggings	0	0	5	78	8	5	3	0	0	0	0	0	1
Jeans	0	0	0	16	83	1	0	0	0	0	0	0	0
Sweatshorts	0	0	8	14	7	65	4	0	0	1	0	0	1
Skirt	0	3	1	1	0	3	89	0	1	1	0	0	1
Hoodie	3	1	0	0	0	0	0	58	13	9	2	4	10
Tee	1	0	1	0	0	0	0	4	75	4	0	13	2
Sweater	6	1	0	1	0	0	0	11	5	51	9	2	14
Poncho	14	5	0	0	0	1	4	9	7	18	20	2	20
Tank	1	2	0	0	0	0	0	4	3	2	0	85	3
Cardigan	12	1	1	2	0	0	4	13	5	15	10	6	31

➢ Classification Matrix for h2 layer

```
Classification Report for h2 layer:
             precision    recall  f1-score   support

       Coat       0.69      0.84      0.76       100
     Romper       0.85      0.93      0.89       100
    Cutoffs       0.85      0.90      0.87       100
   Leggings       0.69      0.78      0.73       100
      Jeans       0.84      0.83      0.83       100
Sweatshorts       0.81      0.65      0.72       100
      Skirt       0.82      0.89      0.85       100
     Hoodie       0.59      0.58      0.58       100
        Tee       0.69      0.75      0.72       100
    Sweater       0.48      0.51      0.49       100
     Poncho       0.48      0.20      0.28       100
       Tank       0.75      0.85      0.80       100
   Cardigan       0.34      0.31      0.32       100

avg / total       0.68      0.69      0.68      1300
```

CNN

❖ Here we implemented the CNN model for classification on each layer differently.

Experiment ID	Experiment Variable			Accuracy on test Dataset using CNN	
	Alpha	Beta	Epochs	H1 Layer	H2 Layer
1	• 1.0	• 0.0	• 1-60	93.00%	NULL
2	• 0.0	• 1.0	• 1-60	NULL	68.53%

- H1 Layer
  - Model name: h1_model1.h5
  - Accuracy

```
loss : 0.22833989554586318
accuracy : 0.9300000071525574
```

  - Confusion matrix

```
Confusion Matrix for h1 layer:
     1    2    3
1  548   15   37
2    8  489    3
3   25    3  172
```

  - Classification report

```
Classification Report for h1 layer:
             precision    recall  f1-score   support

          1       0.94      0.91      0.93       600
          2       0.96      0.98      0.97       500
          3       0.81      0.86      0.83       200

avg / total       0.93      0.93      0.93      1300
```

- H2 layer
  - Model name: h2_model1.h5
  - Accuracy

```
loss : 1.4168329693720891
accuracy : 0.6853846311569214
```

  - Confusion matrix

Confusion Matrix for h1 layer:

	Coat	Romper	Cutoffs	Leggings	Jeans	Sweatshorts	Skirt	Hoodie	Tee	Sweater	Poncho	Tank	Cardigan
Coat	85	6	0	0	0	0	2	0	1	2	0	0	4
Romper	0	94	0	0	1	0	2	0	0	1	0	0	2
Cutoffs	0	0	90	0	1	6	3	0	0	0	0	0	0
Leggings	0	0	3	77	10	5	4	0	0	1	0	0	0
Jeans	0	0	0	7	92	1	0	0	0	0	0	0	0
Sweatshorts	1	0	11	11	7	60	8	0	0	1	0	0	1
Skirt	0	3	4	0	1	2	86	0	1	0	0	2	1
Hoodie	2	0	0	0	0	0	0	56	13	15	3	2	9
Tee	1	0	0	1	0	0	0	8	69	10	1	9	1
Sweater	6	2	0	0	0	0	1	16	6	50	9	3	7
Poncho	13	6	0	0	0	0	1	11	5	17	24	7	16
Tank	0	2	1	0	0	0	0	5	1	3	1	84	3
Cardigan	18	2	1	0	0	3	4	13	5	19	5	6	24

➢ Classification report

Classification Report for h1 layer:

	precision	recall	f1-score	support
Coat	0.67	0.85	0.75	100
Romper	0.82	0.94	0.87	100
Cutoffs	0.82	0.90	0.86	100
Leggings	0.80	0.77	0.79	100
Jeans	0.82	0.92	0.87	100
Sweatshorts	0.78	0.60	0.68	100
Skirt	0.77	0.86	0.82	100
Hoodie	0.51	0.56	0.54	100
Tee	0.68	0.69	0.69	100
Sweater	0.42	0.50	0.46	100
Poncho	0.56	0.24	0.34	100
Tank	0.74	0.84	0.79	100
Cardigan	0.35	0.24	0.29	100
avg / total	0.67	0.69	0.67	1300