Improved deep learning techniques for recognition and labeling

Abrar Hamzeh Saad Abdul Nabi

2017

Abrar Hamzeh Saad Abdul Nabi. (2017). Improved deep learning techniques for recognition and labeling. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/72748>

<https://doi.org/10.32657/10356/72748>

Improved Deep Learning Techniques for Recognition and Labeling

Abrar Hamzeh Saad Abdul Nabi

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2017

*To my very loving husband Zaid, my father Hamzeh, my mother Muna,
my sisters: Tahani, Amani, Razan, Isra'a, Welda and my brother Anas*

Acknowledgments

I am very grateful to my supervisor Professor Lap-Pui Chau and appreciate his kind help and support. I would also like to thank a lot my previous supervisor Professor Gang Wang, who has supported me throughout my Ph.D. study with his kind patience, knowledge and help. I would also like to thank my co-supervisor Dr. Stefan Winkler for his kind help and support. While my years are passing in Singapore, I really have learned a lot. I have been always encouraged and motivated to deliver high research quality. I will not regret any single day. I would like to thank all my group members and friends. Special thanks to Dr. Jiwen Lu, Dr. Jiangbo Lu, Bing Shuai and Zhen Zuo for their help and support. I really appreciate every single discussion, suggestion, encouragement, laughing moment and help from all my friends. I am very grateful to my Ph.D. journey, it has been a very maturing experience in my life. I have changed a lot. Last, I can not express in words how much my gratitude is towards my beloved husband, dad, mom, my most adorable sisters and brother. Indeed, I am really very thankful to Allah for everything.

Contents

Acknowledgments	ii
Abstract	vii
List of Tables	xviii
List of Abbreviations	xxi
List of Notation	xxii
1 Introduction	1
1.1 Background	1
1.2 Motivations	5
1.3 Contributions of the Thesis	8
1.4 Organization of the Thesis	10
2 Related Works	11
2.1 Convolutional Neural Networks (CNNs)	11
2.2 Recurrent Neural Networks (RNNs)	13
2.3 Attention-based Models	14
2.4 Semantic Attributes	15
2.4.1 Definition of the Attribute	15
2.4.2 Global vs. Local Attributes	15
2.4.3 Correlated Attributes	16
2.4.4 Attributes and Multi-labeling	17
2.5 Multi-task Learning	18

2.6	RGB-D Scene Labeling	19
2.7	Multi-modal Learning	20
2.8	Scene Labeling and Contextual Modeling	21
3	Multi-task CNN Model for Attribute Prediction	25
3.1	Introduction	25
3.2	Related Work	30
3.2.1	Single-label and multi-labels with CNNs:	30
3.2.2	Convex optimization and CNN:	31
3.3	Multi-task CNN Models	33
3.3.1	Sharing the Latent Task Matrix in MTL	34
3.3.2	Feature Sharing and Competition in MTL	35
3.3.3	Formulations of the Multi-task CNN model	37
3.3.4	Optimization Steps	39
3.4	Experiments and Results	42
3.4.1	Datasets and Grouping	42
3.4.2	Attribute Prediction Accuracy	43
3.4.3	Implementation Details	47
3.5	Chapter Summary	48
4	Multi-modal Recurrent Neural Networks with Information Transfer Layers for Indoor Scene Labeling	49
4.1	Introduction	50
4.2	Multi-modal RNNs Framework	53
4.2.1	Multimodal RNNs via Information Transfer Layers	55
4.3	Experiments and Results	65
4.3.1	Datasets	65
4.3.2	CNNs and RNNs Training	65
4.3.3	Baselines	66

4.3.4	Results	68
4.3.5	Observations	78
4.4	Chapter Summary	79
5	Episodic CAMN: Contextual Attention-based Memory Networks With Iterative Feedback For Scene Labeling	80
5.1	Introduction	81
5.2	Related Work	83
5.2.1	Recurrent Feedback	83
5.3	Framework	84
5.3.1	FCN for Local Patch Representation	85
5.3.2	Contextual Attention-based Memory Model	85
5.3.3	Model Optimization	90
5.4	Experiments	91
5.4.1	Datasets	91
5.4.2	Implementation Details	92
5.4.3	Evaluation Results	93
5.5	Chapter Summary	98
6	Beyond Forward Shortcuts: Fully Convolutional Master-Slave Networks (MSNets) with Backward Skip Connections for Semantic Segmentation	103
6.1	Introduction	104
6.2	Related Work	106
6.2.1	Evolution of Skip Connections in CNNs	107
6.2.2	Contextual modeling and CNNs	107
6.3	Fully Convolutional MSNets	108
6.3.1	Baseline Models	109
6.3.2	Master-Slave Network Design	112

6.4	Experiments	115
6.4.1	Datasets	115
6.4.2	Baselines and Methods	116
6.4.3	Optimization Details	116
6.4.4	Results	117
6.5	Chapter Summary	121
7	Conclusions and Future Works	124
7.1	Conclusions	124
7.2	Future Works	126
	Publications	129
	References	130

Abstract

Representing images in robust, discriminative and informative features is deemed to be crucial for good recognition performance on many fundamental computer vision problems such as semantic attribute prediction and semantic image segmentation. Researchers tend to rely on hand-crafted features like SIFT and HOG which lack the ability to be data-adaptive, hence, feature learning becomes more favorable in the last few years. Various deep learning methods such as CNNs and RNNs achieve impressive state-of-the-art performance in almost all vision fields. In this thesis, we present various improved deep feature learning models that mainly utilize the idea behind sharing knowledge between feature classifiers and representations. We evaluate our proposed deep models on various competitive benchmarks addressing the problems of semantic attribute prediction, RGB-D and RGB semantic segmentation. Our models achieve substantial improvements over the baseline methods and comparable performance to (and in many settings, outperform) other state-of-the-art works.

In the first part of the thesis, we tackle the binary semantic attribute prediction. Usually, the lack of training data for some semantic attributes is a main struggle to learn adequate informative attribute feature representations. Hence, we propose a new joint multi-task deep CNNs model to allow different CNNs to simultaneously share visual knowledge. Each CNN is assigned the task to learn the representation of a single binary semantic attribute. Sharing knowledge allow under-sampled CNN classifiers to leverage shared statistics from other CNN classifiers to further improve their performance. Natural grouping of attributes is applied such that attributes in the same group are encouraged to

share more knowledge and attributes in different groups will generally compete, and consequently share less knowledge. We evaluate our multi-task CNNs model on two popular semantic attribute prediction datasets and outperform other state-of-the-art methods.

In the second part of the thesis and different than attributes we address a more challenging recognition problem that requires to classify detailed semantics; i.e. pixels in input images. Semantic segmentation (also known as scene labeling) is a sequence-sequence prediction task (pixels-labels), and it is quite important to leverage relevant contextual information to enhance the performance of pixel classification. We address two main problems in semantic segmentation which are RGB-D and RGB scene labeling. In RGB-D, an extra input data source is provided which is the Depth maps and to be leveraged during labeling. First, we propose a multi-modal RNNs to tackle RGB-D scene labeling. Our proposed deep model allow multiple RNNs to share knowledge from different data modalities. The multi-modal RNNs exploit the Depth maps beside the RGB color channels to provide more informative contextual cues for local pixel classification. It simultaneously performs training of two RNNs that are crossly connected through information transfer layers. The transfer layers are learned to adaptively extract relevant cross-modality features and in the same time retain model-specific features within each modality. We evaluate our model on two popular RGB-D scene labeling datasets and achieve competitive performance to other state-of-the-art methods and outperform the baseline models significantly.

In the third part of this thesis, we find that contextual information contained within the RGB image are indeed very informative for local pixel classification. Thus, we tackle the problem of RGB semantic segmentation. We propose an improved framework of the Fully Convolutional Network (FCN) and allow it to generate contextually-aware local feature representations for each pixel patch in the input image. We introduce an episodic attention-based memory network to achieve the goal. We present a unified framework that mainly consists of FCN and an attention-based memory module with feedback connections to perform context selection and refinement. The attention-based

memory module is formed of feed-forward neural network and attention-based model for context selection, and an RNN with feedback mechanism for context aggregation and refinement over multiple iterations (episodes). The full model produces context-aware representation for each target pixel patch by aggregating the activated context and its original local representation produced by the convolution layers. We evaluate our model on three competitive scene labeling benchmarks and achieve impressive performance.

Finally, in the last part of the thesis, we continue to address RGB scene labeling and different from all recent works that are about improving CNN performance with the aid of other models, we propose an orthogonal architecture to improve the internal design of the FCN. Our main focus is to allow FCN to independently generate richer contextually-aware feature representations. Many recent works have proposed efficient versions of deep CNNs that mainly contains various designs of forward shortcut connections; i.e. skip connections between the input and the output layers. Lower-level convolutional layers have higher resolutions (location information) or more detailed feature maps that can improve the higher-level layers that lack these details. In terms of training, skip connections create direct shorter paths to the supervisory error signal thus provide more efficient propagation and better training. We observe and study the influence of the backward skip connections which are in inverse direction to the forward shortcuts; paths from the high-level layers to the low-level layers. Aggregating the higher-level feature maps in low-level layers allow the low-level layer features to be contextually consistent in higher-level abstraction. To achieve this which is indeed opposed to the nature of the feed-forward networks, we propose a new fully convolutional model that consists mainly of a pair of networks. A ‘Slave’ network is dedicated to provide the backward connections from its top layers to the ‘Master’ network bottom layers. The Master network contains both forward and backward skip connections and is able to generate contextually-aware features. We also evaluate our proposed new model on three most popular scene labeling benchmarks and achieve competitive performance to other state-of-the-art methods.

List of Figures

1.1	Example image from the Animals with Attributes dataset [1]. $\{1, 0\}$ indicates the existence/absence of the corresponding binary semantic attribute.	3
1.2	The overall organization of Thesis chapters (Roadmap).	4
1.3	Example image from the NYU v2 dataset [2]. Overview of the general pipeline for semantic segmentation. First feature learning/extraction followed by classification to generate the prediction label map.	5
2.1	Visualization examples of different abstractions that are learned in the convolution layers. The first convolution layers usually learn edge features, HOG-like features and color blobs features. With more application of the convolution layers a higher abstraction is learned. For example, the second layer starts to capture more complex shapes and features. Meanwhile Layer 5 is able to capture the whole face of human, dog or cat. The complete content of this Figure is originally from [3].	12
2.2	Illustration of the one-dimensional RNN (right side) and multidimensional (e.g. two-directional) RNN (left side) [4]. The key idea is to replace the single recurrent connection in the 1D-RNNs with as many recurrent connections as there are dimensions in the input data.	14

2.3	Some intuitive example images with their attribute descriptions. From left to right, the first and second images are from the Clothing attribute dataset [5] while the third image is from the AwA dataset [1]. For example, the local attribute ‘striped’ is describing the donkey’s body parts directly. But a global attribute such as ‘middle age’ is describing a holistic property of a human which is not only related to specific part; but to several locations and parts of the man image.	16
2.4	Some example images with their correlated attributes. The images are from the AwA dataset [1]. Notice that ‘water’ and ‘eats fish attribute usually co-occur together. Also for the zebra; ‘white’, ‘black’ and ‘stripes’ attribute usually appear together in the images.	17
2.5	An example scene image that have both the RGB image and the depth plane. The corresponding depth information are utilized to better classify local pixels.	19
2.6	The top part illustrates the concept behind the multi-modal/multi-view learning method where usually different data modalities are combine by various techniques to exploit all information sources. Multi-modal is also efficient to combine and fuse the RGB color channels with Depth planes (bottom part)	22
3.1	Illustration of some binary semantic attributes. The shown examples are from the Clothing Attribute Dataset [5]. Yes/No indicates the existence/absence of the corresponding attribute.	28

3.2	Our proposed multi-task CNN model. The input image (in the left) with attribute labels are forwarded through the model. Each CNN will predict one binary semantic attribute. The shared latent layer L together with the S layer act as weight matrices of the last fully connected layer. The L layer is a shared latent matrix between all CNN models. Each vector in S is CNN-specific weight matrix. The soft-max and loss layers are replaced by our multi-task squared hinge loss. Group information about attribute relatedness is encoded during the training of the network.	32
3.3	An illustration example of the intra-group feature sharing and inter-group feature competition.	39
3.4	Qualitative examples where single-task CNN classifiers misclassify them and our Multi-task CNN classifiers are able to correctly classify these samples. The first 4 rows have samples from AWA dataset [1]. The last 2 rows are samples from Clothing attributes dataset [5]. Yes/No indicates whether the attribute is presented or absent in the image.	46
4.1	2D-RNN structure is a generalization of 1D-RNN. A visualization of processing patches in 2D image is shown. The scan is only presented in one direction (top-left). Here, the previous hidden states from the top and the left sided are encoded into the current hidden state, i.e. the previous top and left neighbored hidden state information are engaged in calculation of the current network state alongside the current patch at the same location. An image is approximated through four directions: top-left, bottom-left, top-right and bottom-right accordingly. Thus, we use the quad-directional 2D design to accumulate information from all sides.	54

4.2	A conceptual illustration to show our principal design in our proposed multimodal 2D-RNNs. This illustration is based on simple 1D-RNNs and intended to clarify the design structure of two crossly-connected 1D-RNNs through the introduced transfer layers. In our model, since we process 2D input signals (images) we use 2D-RNNs instead of 1D-RNN to retain the spacial dependencies within each scene image.	54
4.3	An abstract overview of our multimodal-RNNs model which consists mainly of an RGB-RNN and a Depth-RNN. The RGB-RNN will process a patch from the RGB input plane in addition to its own previous hidden states and the selections made by the transfer layer from the previous hidden states of the Depth-RNN. The Depth-RNN will also process the corresponding patch from the Depth input plane in addition to its own previous hidden states and the selections made through the transfer layer from previous hidden states of RGB-RNN. For the simplicity in this figure, we only show one hidden plane from the quad-directional RNN hidden planes. Every hidden plane is crossly connected to its peer in the multimodal-RNNs through information transfer layers.	57
4.4	An illustration of the forward and backward passes in the top-left plane sequence in both RGB-RNN and Depth-RNN. (Best viewed in color) . .	64
4.5	Per class accuracy comparison between our multimodal RNNs, baselines CNN-RGBD and RNN-RGBD on NYU V2 dataset [2](14 semantic categories).	72
4.6	The relationship between the hidden layer dimensions and the pixel-wise accuracy in a single RNN model trained on the NYU V2 dataset [2] . . .	73

4.7	7 example results from NYU-V1 (13 semantic categories). Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th:ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color)	74
4.8	7 example results. Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th:ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color) .	75
4.9	7 example results from NYU-V2 (14 semantic categories). Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th:ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color)	76
5.1	An illustrative overview of our unified framework. The model consists of FCN convolution layers, our episodic contextual attention-based memory network, and the upsampling alongside the FC (classification) layers. Given an input image, our model generates a pixelwise label map.	82

5.2	An illustration of our Episodic CAMN including its contextual attention-based network, the episodic recurrent memory and the feedback. The model consists of two major components: (1) the CAN adaptively selects the relevant context to contextualize the reference patch representation ($\mathbf{x}_i$); (2) the episodic memory summarizes the activated context in the past. The feedback from the episodic memory enables the CAN to iteratively refine the selected context over multiple iterations.	86
5.3	Examples of qualitative labeling results (best viewed in color). Each row shows the original test image, its ground truth label map, and three label prediction maps of our convolutional episodic attention-based contextual memory network with different iterations. The test images are from the PASCAL-Context [6]. The quality of the label prediction maps is remarkably improved as CAN is engaged (iteration 1), and is gradually refined after additional iterations.	95
5.4	Comparison with CRF-RNN method [7]. The samples are from PASCAL VOC 2011 dataset [8].	96
5.5	Some sample images from SIFT Flow [9] and PASCAL VOC 2011 [8] alongside the visualization of the weighted selections for specific local patches. From left to right: RGB image, label map, episode 1 heatmap and the final episode 3 heatmap.	98
5.6	IU performance changes in terms of iterations.	99
5.7	Supplemental Material. Examples of qualitative labeling results (best viewed in color). The images are from PASCAL VOC 2011 dataset. . . .	100
5.8	Supplemental Material. Examples of qualitative labeling results (best viewed in color). The images are from SIFT Flow dataset.	101

6.1	(Best viewed in color). Refer to Section 6.4.2 for baseline definitions. Our MSNet model (last column) can correctly label object regions as the backward skip connections allow the lower layers to learn more informative features by receiving information from higher layers (contextually-aware). The sample images are from PASCAL VOC 2011 [10] validation set.	105
6.2	‘Forward skip’ is an illustration of one forward skip connection in our own baseline FCN-8s-F1. ‘Backward Skip’ is our proposed design where we align two networks, a Master and Slave. The Slave network provides the backward skip connections from its higher layers to the Master network lower layers. ‘Block in MSNet’ shows an illustration of our final model’s skip connections, where additional forward skip connections have been added to the Master.	106
6.3	(Best viewed in colors). A detailed overview of our MSNet-FB1 model. The model consists of two networks; Master and Slave. Slave is a basic FCN-8s. The Master is a FCN-8s-F1 (as our baseline) and contains the backward skip connections taken from the Slave network. The forward propagation starts in the Slave network, and then the generated feature maps are fed back to lower layers in the Master network serving as backward skip connections within the Master. Notice that the backward skip connection (blue arrows) are exactly the inverse of the forward skip connections (purple arrows).	109
6.4	(Best viewed in colors). A detailed overview of the Baseline model networks; FCN-8s, FCN16-s and FCN-32s. The early design ‘FCN-32s’ does not include skip connections, whereas FCN-16s has one skip connection from <i>pool4</i> layer, and FCN-8s has skip connections from both <i>pool4</i> and <i>pool3</i> layers.	110

6.5	(Best viewed in colors). An illustration of several convolutional blocks of our MSNet-FB2 with dense skip connections. Notice that the Master network is FCN-8s-F2. We set $P = 3$ and $N = 3$. Here the dense backward skip connections are in inverse directions to the dense forward skip connections. Refer to Figure 6.3 for the Legend.	114
6.6	(Best to be zoomed in and viewed in colors) Examples of some qualitative labeling results. For each row, we show the input image, its ground truth, label prediction maps for FCN-8s, FCN-8s-F1, FCN-8s-F2 and our proposed models MSNet-FB1 and MSNet-FB2. The test images are from ADE20K validation set [11] (top) and PASCAL-Context [6] (bottom). The MSNet-FB1 and MSNet-FB2 have extra backward skip connections and improve the quality of the label prediction maps compared with baselines. The FCN-F2 and FCN-F2 baselines improve over FCN-8s as they have more forward skip connections.	119

List of Tables

3.1	Examples of attribute groups from AwA dataset [1]	36
3.2	Solving the Optimization Problem of Equation —2	41
3.3	Grouping information used in Clothing dataset[5]	42
3.4	The prediction accuracy of the semantic binary attributes before sharing the L layer, after sharing and other methods on the clothing dataset [12]. We refer to G1 as the group of Color attributes, G2 refers to the Pattern group, G3 refers to Cloth-parts awhile G4 refers to the Appearance group. MG-CNN is our overall proposed multi-task CNN model.The higher, the better.	44
3.5	Attribute prediction scores of our Multi-task CNN model compared with other methods on AwA [1]. The higher, the better. (mean average precision)	45
3.6	The group-level accuracy results of our Multi-task framework on AwA [1]. The higher, the better.	45
4.1	Results on NYU V1 dataset [13]. The performance of all baselines and other state-of-the-art methods is measured on total pixel-wise, average class-wise accuracy and average IOU (intersection over union) among 13 semantic classes. The higher the better.	69

4.2	Results on NYU V2 dataset [2]. The performance of all baseline and other state-of-the-art methods is measured on total pixel-wise, average class-wise accuracy and average IOU (intersection over union) among 4 semantic classes: Ground, Furniture, Props and Structure. The higher the better.	71
4.3	Results on NYU V2 dataset [2]. The performance of all baseline and other state-of-the-art methods is measured on total pixel-wise, average class-wise and average IOU (intersection over union) among 14 semantic classes. The higher the better.	73
5.1	Results on the PASCAL-Context [6], SIFT Flow [9] and PASCAL VOC 2011 (validation set) [8]. All models are trained jointly with FCN models (best performing ones).	92
5.2	Performance comparison on PASCAL-Context[6].	97
5.3	Performance comparison on SIFT Flow [9].	99
5.4	Average performance comparison on the test set from PASCAL VOC 2011 [8].	102
6.1	Results on the ADE20K [11] (validation set), PASCAL-Context [6] and PASCAL VOC 2011 (validation set) [10]. Refer to Section 6.4.2 for detailed definitions.	114
6.2	We report training and testing run time on the validation set of ADE20K (averaged over 20 trials for an input image of 384×384 on NVIDIA GeForce GTX Titan X and the companion CPU is Intel Xeon E5-2650 v2 at 2.60GHz).	117
6.3	Results on the validation and testing sets of PASCAL VOC 2011 [10]. MSNet- $\ast$ +CRF refers to applying fully-connected CRF on top of the label predictions generated from our MSNet models (post processing step). . .	118
6.4	Performance comparison on PASCAL-Context [6].	121
6.5	Performance comparison on the validation set from ADE20K [11].	122

6.6	Average performance comparison on the test set from PASCAL VOC 2011	
	[10].	122

List of Abbreviation

ADE20K	ImageNet Scene Parsing Dataset
APG	Accelerated Proximal Gradient Descent
AwA	Animals with Attributes Dataset
BP	Back Propagation
BPTT	Back Propagation Through Time
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
DBN	Deep Belief Network
FCN	Fully Convolutional Network
GRU	Gated Recurrent Unit
HMM	Hidden Markov Models
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LBP	Local Binary Patterns
LFDA	Local Fisher Discriminant Analysis
LSTM	Long Short Term Memory
MAP	Maximum <i>a posteriori</i>
MML	Multi-modal Learning
MRF	Markov Random Fields
MSCOCO	Microsoft Common Objects in Context
MTL	Multi-task Learning
ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red, Green and Blue color channels
RGB-D	Red, Green, Blue and Depth
RNN	Recurrent Neural Network
PASCAL VOC	The PASCAL Visual Object Classes Challenge
PCA	Principal Component Analysis
SGD	Stochastic Gradient Descent
SPGD	Smoothing Proximal Gradient Descent
SPM	Spatial Pyramid Matching
SVD	Singular Value Decomposition
SVM	Support Vector Machine

List of Notation

$(\cdot)^T$	Matrix or vector transpose
$\sqrt{}$	Square root
$\mathbf{x} \in \mathbb{R}^D$	A D –dimensional vector
$\mathbf{X} \in \mathbb{R}^{D_1 \times D_2}$	A matrix $\mathbf{X}$ of size $D_1 \times D_2$
$ \mathbf{x} $	Absolute of $\mathbf{x}$
$*$	Convolution
$\odot$	Element-wise multiplication
$\exp(\cdot)$	Exponential function
$\tanh(\cdot)$	Hyperbolic tangent function
$\sigma(\cdot)$	Sigmoid function
$\log(\cdot)$	Natural logarithm function
$\delta(\cdot)$	Indicator function
$\ \mathbf{x} - \mathbf{y}\ _2$	Euclidian distance between vectors $\mathbf{x}$ and $\mathbf{y}$
$\ \mathbf{x} - \mathbf{y}\ _2^2$	Squared Euclidian distance between $\mathbf{x}$ and $\mathbf{y}$
$\ \mathbf{x}\ _2$	L2 norm of the vector $\mathbf{x}$
$\ \mathbf{x}\ _1$	L1 norm of the vector $\mathbf{x}$
$\ \mathbf{x}\ _F$	Frobenius norm of the vector $\mathbf{x}$
$\ \mathbf{X}\ _F^2$	Frobenius norm of the matrix $\mathbf{X}$
$[\mathbf{x}; \mathbf{y}]$	Concatenation of the vectors $\mathbf{x}$ and $\mathbf{y}$

Chapter 1

Introduction

1.1 Background

Perceiving and analyzing an image in human visual system is relatively a simple task compared with a visual recognition system in computers. The complex variations and compositions of the location, size, shape and appearance of objects and stuffs in scenes around us require high non-linear processing of those visual perceptions. With time, this becomes an easier task to computers because of the huge availability of visual data that can be exploited to train various complex recognition systems. Fortunately, the popular deep learning frameworks such as CNNs and RNNs provide complete data-adaptive automated recognition systems with satisfying degree of capacity and complexity to recognize and understand objects in image scenes and videos.

Usually, a recognition task contains two main principle steps to produce the final predictions: featurization and classification; i.e. feature learning or extraction from the input (images or multimedia) and classification or regression. Generating useful, discriminative and robust feature representations is a critical component for good classifier performance. Techniques for featurization of input images include: basic feature extraction (e.g. Gabor features [14], SIFT [15], shape context [16], HOG [17], deep feature learning [18]), descriptor encoding (e.g. vector quantization coding [19], sparse coding [20]), CNN applied on raw image pixels [21], and summarization via pooling (e.g. aver-

age pooling [19], max pooling [20]). Many studies also perform classification using the popular Support Vector Machines (SVM) [22], and using Bayesian models [23].

The well designed hand-crafted methods are human labor-exhaustive, highly-engineered and less adaptive to data variations compared to deep learning models [21], [24]. Deep models overcome these problems and provide natural end-end trainable systems. Many works on deep feature learning have been proven to outperform hand-crafted features on different recognition tasks and more challenging datasets [25–31].

The first step towards ultimately understanding an image is to recognize the categories of the objects and stuffs within. This can be done through multiple approaches, e.g. learning to describe objects/stuffs in terms of human language; i.e. semantic attributes (e.g. Figure 1.1) and then exploiting those descriptions for final object/stuff categorization. Another way is to perform parsing of the image in terms of its primitive contents; i.e. pixels and semantically segment each object/stuff category directly (e.g. Figure 1.3). While semantic attributes provide more human-like language descriptions which is useful for some applications, the later task provides access to more detailed and fine-grained descriptions of the image since the classification is performed on pixel-level.

Both semantic attribute prediction and semantic segmentation (we also use scene labeling or image parsing interchangeably to semantic segmentation) tasks are considered as fundamental and important topics in computer vision. These tasks are very crucial in many important applications such as image search and retrieval, object detection, scene analysis, multimedia content analysis and recommendation, etc.

Learning the attributes of an object provides semantic descriptions about it that are useful for the classifier. They serve as mid-level knowledge to link between low-level features and high-level semantics (categories). One key strategy to boost the prediction performance is to share visual knowledge between classifiers. For example, in semantic attribute prediction the attribute classifiers can share visual knowledge across object categories or within the same object category. Sharing visual knowledge helps other under-sampled classifiers to generalize better.

Attributes :

Orange: 1
Stripes: 1
Black: 1
Blue: 0
White: 1
Furry: 1
Vegetation: 0
⋮

Figure 1.1: Example image from the Animals with Attributes dataset [1]. $\{1, 0\}$ indicates the existence/absence of the corresponding binary semantic attribute.

In semantic segmentation, typically research is done on both RGB and RGB-D input. The later one provide extra input data source which is the depth maps to improve local pixel classification. However, in both tasks, the visual knowledge within the image are very important to be shared with the local representation of each pixel. Global information on the image level helps to clarify some ambiguous local representations of some regions/patches. For example, if two local patches look alike and hard to be distinguished, the model can benefit from the contextual surrounding cues to correctly classify them. Usually as humans, we rely on contexts to understand the objects in our surroundings, the case also holds true for scene semantic segmentation. Contextual information within the input image scene impact directly the perception of a target object or stuff. Hence, the visual knowledge contained in an image should be shared in someway among the pixel classifiers or encoded into their local representations.

Similarly, in RGB-D semantic segmentation the extra data modality which is the depth maps provide another source of useful knowledge to be shared beside the main data modality which is the RGB color channels. The main idea is to exploit and share the mutual knowledge from extra available data modality to boost the performance of

Figure 1.2: The overall organization of Thesis chapters (Roadmap).

local pixel classification. Processing multiple data modalities provide different useful cues.

Ultimately, all these tasks require knowledge sharing to improve the recognition performance, i.e. feature sharing on representation-level or classification-level. Recently, most of the methods that address these tasks rely on deep learning models. However, most of these deep models lack the concept of having the ability to allow useful knowledge sharing on feature classifier-level or representation-level. In this thesis, we address this problem and propose to improve popular deep models such as CNNs and RNNs and allow these models to perform various knowledge sharing types on different tasks. We explore multiple techniques to do the sharing while performing the following recognition tasks: binary semantic attribute prediction, RGB-D semantic segmentation and RGB semantic segmentation. Figure 1.2 show the overall organization of the Thesis chapters (Roadmap).

Figure 1.3: Example image from the NYU v2 dataset [2]. Overview of the general pipeline for semantic segmentation. First feature learning/extraction followed by classification to generate the prediction label map.

1.2 Motivations

In this thesis, our goal is to develop improved deep learning models that mainly utilize the idea behind sharing different kind of useful knowledge and information between feature classifiers and representations.

In semantic attribute prediction, two concerning issues arise which are the image multi-labeling and the correlation-based learning. Multi-labeling is more difficult when compared to the single label classification problem. Multi-labeling is when the input image need to be tagged with a set of labels from a predefined pool, and this task is more popular in real word scenarios [32, 33]. For example, in image retrieval, if the user types a query to search for an image that has ‘white striped shirt’, the engine search model should actually be able to retrieve consistent results for a white shirt that has stripes. Computationally, this is a difficult task because of the huge hypothesis space of attributes (e.g., X attributes require 2^X). This actually limits our ability to address the problem in its full form, thus, we usually transform it into multiple single classification problems [34]. Typically, the correlations are explored between all those single classifiers to allow appropriate and adequate sharing of visual knowledge. Multi-task learning is an effective approach for feature sharing, as well as competition among classifiers (so-called ‘tasks’ in

the term multi-task)[35, 36]. If the tasks are related and specially when some task lacks training data, it can receive visual knowledge from other, more fit tasks through the sharing [37–41]. In our work, we benefit from multi-task learning approach and develop an improved version of the state-of-the-art deep CNN model that can perform multiple attribute prediction while allowing knowledge sharing between attribute classifiers. Our proposed approach is different than other recent works in both multi-task learning and deep learning. We develop a new fusion between CNNs and blend them in a new multi-task framework where they are allowed to share visual knowledge effectively.

In typical semantic segmentation (RGB and RGB-D), one of the key strategies that is useful for classification is to leverage the neighborhood/contextual information for each pixel. Usually, the feature representation of a pixel is extracted from a local patch (cropped from the scene image) containing that target pixel and used for classification. Long-range/global contextual information (distant image patches) are important as well for local pixel classification. However, both local and global contextual information should be utilized adequately to maintain a good balance between locally discriminative and abstract/top-level feature representation of each pixel. Most of the recent deep methods rely on CNNs to perform local feature learning of each patch/pixel. RNNs (or LSTMs and GRUs) may be engaged to encode global information into the local representations of the pixels.

In case of RGB-D semantic segmentation, depth plans are provided as extra information to benefit local pixel classification. This problem is usually addressed as a multi-modal learning problem where the task is to exploit and fuse both RGB and depth modalities to better label each pixel. It is a challenging classification problem since the indoor scene image (because the depth cameras are affordable usually indoor) tends to contain multiple messy objects. To well address this problem, we develop a new deep model which is multi-modal RNNs. Our method is different from all of the existing deep multi-modal learning methods [42–45]. They usually concatenate inputs at the beginning

or concatenate some learned middle level features to extract high-level common features as the representations of the multi-modal input data. These methods mainly focus on discovering common patterns between different modalities. However, some modality specific information is high discriminative, e.g., the texture patterns of the RGB channels, and is missed in such methods. In contrast, our model retains modality-specific information by assigning an RNN model to learn features from each modality. Our method allows sharing information between modalities by introducing the information transfer layers between RNNs to adaptively transfer relevant cross-modality patterns. Our model is trained end-to-end and the transfer layers are learned to extract relevant across-modality information for each patch of the image (select the related and useful ones).

While in RGB semantic segmentation, each pixel representation is also learned from a local patch that contains some local surrounding context to locally discriminate the pixel during labeling. But these local feature representations can be ambiguous, e.g. a sofa patch can be visually indistinguishable from a bed patch. As a natural solution, many papers utilize context to distinguish locally ambiguous patches [46–48]. RNNs are usually considered as encoding methods to leverage pixel feature dependencies. However in all these works, the long-range context is not effectively leveraged. Besides, the surrounding contextual information is incorporated without selection. We believe that different contextual information is not always equally important or useful to one local region. For example, when recognizing a local region belonging to computer monitor, it is more helpful to leverage contextual information from desk rather than from other regions such as window. Hence, special attention should be paid to certain context conditioned on the target region. To fulfill the goal, we develop end-to-end trainable deep model that mainly consists of CNNs, attention-based models and RNNs. Different than all those recent deep models, our model adaptively and progressively select related contextual information and produce powerful contextually-aware local feature representations.

We follow up with the RGB semantic segmentation task, and further investigate CNN models only and study their ability to produce contextually-aware features on their own

without the support of RNNs or attention-base models. During our close investigation of CNN models, we observe that the high-level semantics and abstractions are usually learned in top layers inside typical deep CNN structures. High-level semantics are very informative to local pixel representations. Different from all recent deep models, we are the first to propose to backward the connections from top to bottom layers in order to transfer those important high-level knowledge. We propose a new architecture design which is the Master-Slave network to achieve our goal.

1.3 Contributions of the Thesis

In the first part of this thesis, we tackle the problem of binary semantic attribute prediction and address it as a multi-task learning problem. In the second part, we tackle RGB-D semantic segmentation and address it as multi-modal learning problem. Meanwhile the third and forth parts tackle the typical RGB semantic segmentation and address it as contextual modeling problem.

The major contributions of this thesis are:

a) We propose an enhanced deep CNN structure that allows different CNN models to share knowledge through multi-task framework while performing binary semantic attribute prediction. To fulfill the task, we propose a new multi-task method where we naturally leverage the grouping information to encourage attributes in the same group to share feature statistics and discourage attributes in different groups to share knowledge. We decompose the overall model parameters into a latent task matrix and a linear combination weight matrix. The latent task matrix is shared among all CNN models and allow learning more localized features (chapter 3).

b) We propose multi-modal RNNs model for RGB-D indoor scene semantic segmentation. The model is optimized to classify image pixels given two input sources: RGB color channels and Depth maps. It simultaneously performs training of two RNNs that are crossly connected through information transfer layers, which are learned to adaptively

extract and select relevant cross-modality useful features. Each RNN model learns its representations from its own previous hidden states and transferred patterns from the other RNNs previous hidden states, thus, both model-specific and cross-modality features are retained (chapter 4).

c) We propose a contextual attention-based memory network model for RGB semantic segmentation. The model is able to adaptively select relevant context for each target patch given each image. Importantly, it is a novel contextual modeling method that can effectively engage long-range context within the image to contextualize local pixel representations. We also introduce an episodic memory module to accumulate contextual selections and aid the contextual attention-based model to refine the previous context selections through recurrent feedback connections over multiple iterations/episodes (chapter 5).

d) We propose a new fully convolutional model that consists of a pair of FCN models to perform semantic segmentation. The model is designed in a way to allow sharing features from high-level layers into low-level layers. A ‘Slave’ network is dedicated to provide the backward connections from its top layers to the ‘Master’ network’s bottom layers. The Master network is used to produce the final label predictions. Our model is different than the recent deep CNNs which contain only forward shortcut connections; i.e. skip connections from low to high layers. Reusing features from lower layers that have higher resolution (location information) benefit higher layers to recover lost details and mitigate information degradation. However, during inference the lower layers do not know about high layer features, although they contain contextual high semantics that benefit low layers to adaptively extract informative features for later layers. In our proposed model, we study the influence of backward skip connections which are in the opposite direction to forward shortcuts, i.e. paths from high layers to low layers. To achieve this – which indeed runs counter to the nature of feed-forward networks – we propose our Master-Slave model design (chapter 6).

e) We evaluate our proposed deep models on most competitive benchmarks addressing the problems of semantic attribute prediction, RGB-D and RGB semantic segmentation. Our deep models achieve substantial improvements over all the baseline methods and competitive performance to other state-of-the-art works. In many cases, our models outperform other recent methods and achieve the state-of-the-art results on these datasets.

1.4 Organization of the Thesis

This thesis is organized as follows: In Chapter 2, a general overview of relevant works is provided. In Chapter 3, our proposed Multi-task CNN Model for Attribute Prediction is presented. In Chapter 4, our Multi-modal Recurrent Neural Networks with Information Transfer Layers for RGB-D Indoor Scene Labeling is introduced. In Chapter 5, we propose Episodic CAMN: Contextual Attention-based Memory Networks With Iterative Feedback For Scene Labeling. In Chapter 6, we present our new Fully Convolutional Master-Slave Networks (MSNets) with Backward Skip Connections for Semantic Segmentation. Chapter 7 concludes the thesis, and presents several future research directions.

Chapter 2

Related Works

This chapter includes a survey of important works related to this thesis. We first review recent works about CNNs, RNNs and Attention-based models in Section 2.1, 2.2 and 2.3 respectively. Then we review the most related methods that address semantic attribute prediction, RGB-D and RGB semantic segmentation. In Section 2.4 a detailed review of semantic attribute prediction is introduced while in Section 2.5 the related multi-task learning works are presented. Then we review RGB-D scene labeling and the related works in multi-modal learning in Sections 2.6 and 2.7. We also review the related works that address scene labeling as contextual modeling problem in Section 2.8.

2.1 Convolutional Neural Networks (CNNs)

CNNs are the most popular models in Deep Learning [49–53]; their goal is to model high-level abstractions of visual data by using multiple non-linear transformation architectures. CNN has proven to show extraordinary performance, specifically in object recognition and image classification applications [50, 51]. Deep versions of CNN achieve higher performance, dominating the top accuracy benchmarks in various vision problems [50, 51, 54–56]. CNN model is able to generate generic and robust features [25, 49]. CNN learns image features from raw pixels through several convolutional layers (with several pooling and non-linearity layers stacked in between), constructing a complicated,

Figure 2.1: Visualization examples of different abstractions that are learned in the convolution layers. The first convolution layers usually learn edge features, HOG-like features and color blobs features. With more application of the convolution layers a higher abstraction is learned. For example, the second layer starts to capture more complex shapes and features. Meanwhile Layer 5 is able to capture the whole face of human, dog or cat. The complete content of this Figure is originally from [3].

non-linear mapping between the input and the output. The lower convolution layers capture basic ordinary phrasing features (e.g., edges and color blobs), and the top layers are able to learn more complicated structure (e.g., car wheel)[3]. Figure 2.1 show some visualization examples of the abstractions that are learned in the convolution layers. Higher layers tend to capture higher abstractions of the input image. Subsequently, it is believed that such implementation of artificial Neural Networks mimics the visual cortex in the human brain [50], it may not be really the same as the real brain functionality but CNN generates good hierarchical representations and tries to model the non-linearity in the input.

Two concerning issues when training a CNN; the number of training samples and the time that is required to train the network until satisfying convergence. Thus, to have an effective CNN model, the training dataset and time should be relatively large enough for the CNN model to well perform the target task [50, 54]. The learned features generated by the CNN are shown to be generic, robust, and more effective than hand-crafted features [25, 57]. Fortunately, the popular implementations in [50, 57] with the technique of initializing from some pre-trained CNN models like on the Imagenet dataset [50], this makes it an easy task to fine-tune different CNN models on many computer vision datasets.

2.2 Recurrent Neural Networks (RNNs)

A recurrent model refers to a model which has connections between its units to form a directed cycle, for example, when a feedback connection from the current state is engaged in the calculation of the future state. This structure creates a sort of complementary internal state for the network, which then allows it to exhibit dynamic temporal behavior. RNN is effectively used in tasks which require sequence modeling, like speech recognition, handwriting recognition, and other natural language processing tasks [4, 58–60].

One major drawback in the standard RNN is the vanishing gradient problem [61]. This drawback limits the context range of the input data, because the capacity of the model is limited to capture enough long dependencies. To address this problem, Hochreiter and Schmidhuber [62] propose the Long Short Term Memory (LSTM), where they treat the hidden layer as multiple recurrently connected subnets, known as memory blocks, thus allowing the network to store and access information over long periods of time. Graves et al. extend the idea of unidirectional LSTM network into bidirectional networks which have shown good improvements over the unidirectional networks [63, 64].

Graves et al. also extend the one-dimensional RNN into multidimensional one [4, 58]. Figure 2.2 illustrates the one-dimensional RNN (right side) and multidimensional (e.g. two-directional) RNN (left side). The key idea is to replace the single recurrent connection in the 1D-RNNs with as many recurrent connections as there are dimensions in the input data. Another interesting work from Graves et al. investigates deep structures of RNNs [60], which is also successfully applied in opinion mining by Irsoy et al. [59]. In our work we evaluate both deep and LSTM structures alongside the basic quad-directional 2D-RNNs. We found that there is no significant difference in the label prediction performance before and after stacking multiple layers of the network or engaging the LSTM units with the basic quad-directional 2D-RNNs. Thus, we only show the results of our multi-modal-RNNs model using the basic quad-directional 2D-RNNs.

Figure 2.2: Illustration of the one-dimensional RNN (right side) and multidimensional (e.g. two-directional) RNN (left side) [4]. The key idea is to replace the single recurrent connection in the 1D-RNNs with as many recurrent connections as there are dimensions in the input data.

2.3 Attention-based Models

Attention-based models have been successfully applied on a broad range of Natural Language Processing (NLP) related tasks, which include machine translation [65] [66], speech recognition [67], image caption generation [68], reading comprehension [69], sentence summarization [70], part-of-speech (POS) tagging [71], question answering [72, 73], etc. Some works benefited from attention models for image classification tasks [74–76] and object recognition [77]. The attention mechanism was first introduced in neural machine translation [65] to automatically align the words in the original sentence with the words in the target sentence using encoder-decoder RNN model. The aligned (attention-based) model is embedded in the sequence to sequence learning framework [78]. Concurrently, in the image caption generation task, Xu et al [68] use the attention-based model to roughly localize image regions of interest, which are deemed as relevant in producing the next word. Another interesting work [79] use attention to select from different multi-scale features. Hermann et al [69] and Rush et al [70] employ the attention-based model to discover the keywords/sentences that are informative to comprehend the paragraph or sentence. Chorowski et al [67] also leverage the attention-based model to filter noisy frames in a short window-based audio to interpret the desired phoneme. Neural Memory Networks [80–83] became popular with the aid of attention mechanism where separate

memory structures are involved to stably store information. One related work is the Dynamic Memory Network (DMN) proposed by Kumar et al. for NLP [72]. DMN consists of five modules: input module, semantic memory module, question module, episodic memory module, and answer module. The modules are mainly constructed using Gated Recurrent Networks (GRU) [84]. Another line work discuss the visual attention [85] [86] in a different perspective. Their attention model mainly localizes the visually catchy image regions. In contrast, the image regions activated by our attention-based model may not necessarily be visually salient. To the best of our knowledge, we are the first work that explores the attention-based model to achieve contextual modeling in semantic segmentation (scene labeling).

2.4 Semantic Attributes

2.4.1 Definition of the Attribute

An attribute is a visual property concerning some object or stuff in an image, if this property appears then the image has this attribute. When the property is expressible in human language, it is called a semantic attribute. Attributes are descriptions of different image features like patterns, colors and shapes [87]. Some recent works focused on how to enhance human-interaction applications using such mid-level language (attributes). It always important to have a consistent alignment between a human query expressions (query attribute phrases) and the corresponding generated interpretations from the recognition model. Attributes satisfy the need as they are usually expressions/phrases in human-language.

2.4.2 Global vs. Local Attributes

An attribute is global if it describes a holistic or global property in the image, e.g., ‘middle aged’ man. Global attributes do not usually involve specific object parts or locations [27, 35, 88]. The localized attributes are used to describe parts or several locations of an

Figure 2.3: Some intuitive example images with their attribute descriptions. From left to right, the first and second images are from the Clothing attribute dataset [5] while the third image is from the AWA dataset [1]. For example, the local attribute ‘striped’ is describing the donkey’s body parts directly. But a global attribute such as ‘middle age’ is describing a holistic property of a human which is not only related to specific part; but to several locations and parts of the man image.

object or stuff, e.g. ‘striped’ donkey. Figure 2.3 shows some intuitive example images with their attribute descriptions. It is not an easy task to recognize both types, because if the classifier is only trained on high-level labels without spatial information like bounding boxes, the performance of some under-sampled classifiers can degrade. However, some works in [35, 36] show that sharing visual knowledge can overcome this issue and mitigate the effects when a classifier need more training samples to generalize better.

2.4.3 Correlated Attributes

If attributes are related and co-occur together in the same image then they are considered to be correlated. Plus, some attributes will naturally imply others (e.g., ‘green trees’ and ‘open sky’ can imply ‘natural scene’), so this configuration can impose some hierarchical relationship on these attribute classifiers. From feature representation perspective, attributes can be weaved from the same part of the feature space and can be also close to each other, e.g., ‘black’ and ‘brown’ attribute classifiers are both color-based classifiers and belong naturally to the same group [35]. Figure 2.4 shows some example images from

Figure 2.4: Some example images with their correlated attributes. The images are from the AWA dataset [1]. Notice that ‘water’ and ‘eats fish’ attribute usually co-occur together. Also for the zebra; ‘white’, ‘black’ and ‘stripes’ attribute usually appear together in the images.

AWA dataset [1] where some attributes tend usually to co-occur together (e.g. ‘black’, ‘white’ and ‘striped’ attributes). While most of the existing methods train independent classifiers to predict attributes [87, 89, 90], typical statistical models, like structured SVM models and Bayesian models, are used to address the problem. In [87], here the authors use a probabilistic generative model to classify attributes. In the work of [89], objects are categorized based on the discriminative attribute feature representations. Some works model the relationships between classes with stating some pre-assumptions about attribute correlations [41]. Different from this work, the decorrelation attribute method is proposed in [35], where they assume that attribute groups are mutually exclusive such that an attribute can’t be shown in two groups at the same time. Other work in [88] proposes jointly learning several ranking objective functions for relative attribute prediction.

2.4.4 Attributes and Multi-labeling

Image multi-labeling refers to learning to assign multiple labels to an image [33, 91]. It is a challenging problem when to be addressed as is, as when the number of labels increases thus the potential output label combinations become huge and intractable [33]. To miti-

gate this, a common transformation strategy can be performed by dividing the problem into a set of single binary classification problems [32]. Predicting the co-occurring attributes in an image can be also seen as a multi-label learning problem. Most of the related works [35, 88] tend to apply multi-task learning to allow sharing or using some label relationship heuristics a priori [92]. Another interesting work applies ranking functions with deep CNN to rank label scores [55].

2.5 Multi-task Learning

Multi-task learning is usually applied to computer vision problems when there are some under-sampled tasks [37, 41, 93, 94]. MTL is intended mainly to encourage knowledge sharing between multiple correlated tasks simultaneously. Sharing knowledge can improve the performance of some tasks or sometimes all of the tasks and increase their generalization ability [37]. Many knowledge sharing strategies have been explored. The first strategy considers designing various methods to discover the relationships on task-level [95], and the other considers the relationships of feature-level. It mainly considers finding some common feature structures shared by all tasks or like mining the related features between the tasks [96]. Max-margin is known by its discriminative power, while Bayesian is more flexible and thus better suited to engage any a priori [38]. Recent frameworks, like Bayesian [97], Max-margin [98], and their joint extension [38], try to discover task and feature correlations. The work in [99] claims that since usually the dimension of the data is high, the assumption that all the tasks should share a common set of features is actually not reasonable. Alternatively, they simultaneously capture the shared features between tasks while identifying the outliers through introducing an outlier matrix [99].

In other works, [36, 100], the authors relax these constraints naturally by decomposing the model parameters into a shared latent task matrix and linear combination matrix. Hence, all the tasks are encouraged to select what to share through this matrix. In most of these techniques, they apply different regularizations on the model parameters, such as structure sparsity for feature selection and feature competition [41].

Figure 2.5: An example scene image that have both the RGB image and the depth plane. The corresponding depth information are utilized to better classify local pixels.

2.6 RGB-D Scene Labeling

Many papers propose different methods to solve RGB-D scene labeling [2, 13, 101–104]. The most popular indoor scene datasets are the NYU depth datasets V1 [13] and V2 [2]. The initial results [13] are generated by extracting SIFT features on the RGB color images in addition to depth maps. Their results prove that depth information can refine the prediction performance. Figure 2.5 shows some intuitive example of a scene image where the depth information are important during local classification.

Further improvements are made to the NYU V1 by the work [103], where they adapt a framework of kernel descriptors which converts the local similarities (kernels) to patch descriptors. They further use a superpixel Markov Random Field (MRF) and a segmentation tree for contextual modeling. Other works [102, 105] explore depth information through a feature learning approach; [102] learns their features using convolution neural network on four channels; three from the RGB image and the fourth is the Depth image. Wang et al. [106] adapt an unsupervised feature learning approach by performing the learning and encoding simultaneously to boost the performance. Another interesting work done by Tang et al. [107] proposes new feature vectors called Histogram of Oriented Normal Vectors (HONV), designed specifically to capture local geometric properties for object recognition with a depth sensor. Couprie et al. [108] adopts multi-scale convo-

lution neural network to learn discriminative features based on the four channel input images (RGB-D).

In the perspective of contextual modeling, Silberman et al. [109] propose to use appearance cues and room-aligned 3D cues to parse indoor scenes into floor, walls, supporting surface, and object regions. They model contextual dependencies as support relations. Ren et al. [110] exploits kernel descriptors to convert local similarities to patch descriptors, and combined superpixel Markov Random Field (MRF) and segmentation tree for contextual modeling. In [111], Kahler and Reid first over-segments RGB-D images, and then computed labels for segments using decision tree field and regression tree field. They model contextual dependencies by prior knowledge of scene layout. Deng et al. [112] proposes hard mutual mutex constraints to estimate global object co-occurrence, relative height relationship and local support relationship. All of these methods model dependencies in the label space.

Meanwhile some works in RGB image parsing utilize the label dependency modeling approaches, such as graphical models (Conditional Random Fields CRFs [113]), and other works perform feature learning to generate hierarchical multiscale features that are capable of capturing the input context, which is successfully applied through the aid of deep convolutional neural networks [114, 115]. In our work, we addressed the problem of RGB-D scene labeling as a multi-modal learning problem (Figure 2.6). In our proposed multi-modal recurrent neural networks, we are able to capture short and long range context within and between input modalities.

2.7 Multi-modal Learning

In contrast to single-view approaches, most multi-modal learning methods introduce a separate function to model one data modality and jointly optimize them together [116–119]. Deep networks have been applied to learn features over multiple views [42–45]. Multimodal learning improves the classification performance through exploiting the sharable knowledge of the data across modalities.

Typical methods in the literature can be categorized based on whether they combine on feature level [120–122], or classifier level [123, 124]. Some works perform preprocessing (module level) steps on various modalities to generate helpful cues before the learning process, as in [125–127]. Fusion can be done by combining all features into one high-dimensional vector, or by jointly training multiple classifiers to maximize the mutual agreement on distinct modalities of the input data [128, 129]. These methods employ typical regularizations which are applied to explore shared visual knowledge, such as group structured sparsity [130].

We can also group multi-modal methods according to their training procedures into three main categories: Co-training [116, 119], Multiple kernel learning [117, 119], and Subspace learning [118, 119, 131, 132]. Co-training algorithms tend to train alternately to maximize the mutual agreement on two distinct views of data. Multiple kernel learning approaches improve the performance by exploiting different types of kernels that correspond naturally to different views, and combine these kernels either linearly or non-linearly. Meanwhile, Subspace learning methods are mainly similar in obtaining a latent subspace that is shared by multiple modalities. Another interesting line of work proposed by Gupta et al. [133], where they propose to transfer supervision between images from different modalities. Their model is able to learn representations for unlabeled modalities and can be used as a pre-training procedure for new modalities with limited labeled data.

2.8 Scene Labeling and Contextual Modeling

Contextual cues from other patches are usually important for local patch reasoning and prediction. Many recent successful semantic segmentation systems are developed based on CNNs, specifically, FCNs. Usually, the receptive fields of the neurons in the convolution layer correspond to local regions of the input image. They implicitly engage contextual information through cascading multiple layers. However, it would be ineffective to enlarge the receptive fields of the neurons to explicitly and directly model long-range context as it may degrade the local discriminative features [134].

Conceptual illustration of the typical Multi-modal/Multi-view learning method

Figure 2.6: The top part illustrates the concept behind the multi-modal/multi-view learning method where usually different data modalities are combine by various techniques to exploit all information sources. Multi-modal is also efficient to combine and fuse the RGB color channels with Depth planes (bottom part)

FCNs try to overcome this limitation by applying skip connections from earlier layers and ultimately aggregating the intermediate features for classification [135]. Other earlier works try to learn hierarchical and multi-scale CNN features to capture the context across a multi-scale image pyramid [114, 115, 136].

Another line of work revisited convolution operations and applied ‘atrous’ and dilated convolutions to perform contextual modeling [137–139]. Recently, RNN-like layers, e.g. Long-short Term Memory (LSTM) and Gated Recurrent Units (GRUs) [84] are commonly inserted after convolution layers to explicitly capture and encode long-range context into local representations [134, 140–142]. Sharma et al. [48] proposed a recursive context propagation network (rCPN) based on a recursive neural network architecture to propagate context. It recursively aggregates contextual information from local neighborhoods to the entire image and then disseminates global contextual information back to individual local features via random binary parse trees. In a later work [143], Sharma et al. improved rCPN model by including the classification loss of the internal nodes of the random parse trees in the original rCPN loss function, and using an MRF on the parse tree nodes to model the hierarchical dependency present in the output. However, RNN-like models are limited in modeling very long-range sequences.

Another interesting works like Jeremy and Daphne [144], they estimated contextual relationships between regions by a proposed things and stuff context model. Singh and Kosecka [145] proposed a global semantic label descriptor in a three-layer spatial pyramid to refine image retrieval. Their context information is extracted from initial semantic labels through a feedback mechanism. Yang et al. [146] built global and local context descriptors based on classification likelihood maps to refine image retrieval and pixel matching. Their context information are learned through feedback-based mechanism as well. Mottaghi et al. [147] proposed a deformable part-based model, which exploits both local context around each candidate detection as well as global context at the level of the scene, for object detection and semantic segmentation. Tung and Little [148] exploited

adaptive overlapping windows to partially capture contextual information. These methods demonstrate the effectiveness of contextual information in revising scene labeling.

Most of these works model the label-level context. Their contextual information characterizes co-occurrence relationships between labels. We also claim that patch-level context is also crucial for scene labeling. Patch-level context signifies that the occurrence of a patch/region strengthens or inhibits the occurrence of other regions. The co-occurrence relationship is then may be integrated with the local patch’s features to annotate the patch.

Conditional Random Fields (CRFs) and Markov Random Fields (MRFs) are also used to model the label-level context [7, 137, 138, 149, 150]. They characterize co-occurrence relationships between labels. Alternatively, we aim to encode contextual information into local feature representation.

Chapter 3

Multi-task CNN Model for Attribute Prediction

In our first work, we propose a joint multi-task learning algorithm to allow multiple CNN models to share visual knowledge and better predict attributes in images. Each CNN will predict one binary semantic attribute and thus generate attribute-specific feature representations. Through our proposed MTL framework, CNN models are able to simultaneously share visual knowledge among different attribute categories. In our MTL framework, we propose a new method to decompose the overall model’s parameters into a shared latent task matrix and specific linear combination matrix. Furthermore, under-sampled CNN classifiers can leverage shared statistics in the latent matrix from other classifiers to boost their performance. Natural grouping of attributes is encoded such that attributes in the same group are encouraged to share more knowledge. Meanwhile, attributes in different groups will generally compete with each other, and consequently share less knowledge. We show the effectiveness of our method on two popular attribute datasets.

3.1 Introduction

Semantic Attributes are semantic properties that have name and are intended to describe objects. Predicting attributes attracted a lot of attention in visual recognition research

[87, 89, 151]. This is because that learning the attributes of an object provides very useful and detailed knowledge about it. Attributes are also considered to be the bridge between low-level features and high-level categories. The learned attribute features can be further used for later tasks such as object recognition. Various multimedia applications can benefit from attributes, such as: information sharing between different target tasks, knowledge transfer, multimedia content analysis and recommendation, multimedia search and retrieval [87, 89, 152–159].

Usually, many approaches apply discriminative learning techniques to learn semantic attributes [87, 89, 160]. Figure 3.1 shows two example images that have different attribute labels. Other types of attributes, such as data-driven ones, are learned in an unsupervised or weakly supervised manner from the data directly and don’t have names (not semantic) [161]. Another type of attributes which are the Relative attributes are also introduced.

Relative attributes have real values to describe the strength of the attribute presence and they are learned through ranking methods [88, 162]. Most of the existing works apply the discriminative classifiers independently to predict the semantic attributes over typical hand-crafted low-level image features [87, 89, 163]. Very few works model the relationship between semantic attributes, considering that they may co-occur simultaneously in the same image and thus correlated [35, 164, 165]. Hand-crafted low-level features like SIFT and HOG are used in combination with various loss-objective functions for attribute prediction purposes [35, 87, 89].

Some works show that learning attributes is beneficial for knowledge transfer tasks, for example, the task of recognizing unseen classes through the transfer of attributes from another seen class [89, 153]. In the work of [87], attribute models are learned to generalize across different categories by training a Bayes classifier model on the ground truth of semantic attribute labels. Then, they train linear SVMs to learn non-semantic feature patterns and choose those which can be well predicted through validation. Such generalization technique can be exploited across different object categories and also across instances within a category.

Semantic attribute prediction is usually address as a multi-task learning problem. The main issue here is that the there is not enough training examples to train good classifiers for all attributes per each image. Hence, multi-task learning is used to mitigate this case and improve the performance of the under-sampled classifiers through allow visual knowledge sharing.

As we previously discussed, two concerning issues arise which are the image multi-labeling and the correlation-based learning. Multi-labeling is more difficult when compared to the single label classification problem. Multi-labeling is when the input image need to be tagged with a set of labels from a predefined pool, and this task is more popular in real word scenarios [32, 33].

For example, in image retrieval, if the user types a query to search for an image that has ‘white striped shirt’, the engine search model should actually be able to retrieve consistent results for a white shirt that has stripes. Computationally, this is a difficult task because of the huge hypothesis space of attributes (e.g., X attributes require 2^X). This actually limits our ability to address the problem in its full form, thus, we usually transform it into multiple single classification problems [34].

Typically, the correlations are explored between all those single classifiers to allow appropriate and adequate sharing of visual knowledge. Multi-task learning is an effective approach for feature sharing, as well as competition among classifiers (so-called ‘tasks’ in the term multi-task)[35, 36]. If the tasks are related and specially when some task lacks training data, it can receive visual knowledge from other, more fit tasks through the sharing [37–41]. In the work of [35], they jointly learn groups of attribute models through multi-tasking using a typical logistic regression loss function.

In our work, we benefit from multi-task learning approach and develop an improved version of the state-of-the-art deep CNN model that can perform multiple attribute prediction while allowing knowledge sharing between attribute classifiers. Our proposed approach is different than other recent works in both multi-task learning and deep learning.

Black: Yes, Necktie: Yes,
Gender: Male, Strips: No,
Yellow: No, White: No,
Skin Exposure: No ...

Black: Yes, Necktie: No,
Gender: Female, Strips: Yes,
Yellow: No, White: Yes,
Skin Exposure: Yes ...

Figure 3.1: Illustration of some binary semantic attributes. The shown examples are from the Clothing Attribute Dataset [5]. Yes/No indicates the existence/absence of the corresponding attribute.

We develop a new fusion between CNNs and blend them in a new multi-task framework where they are allowed to share visual knowledge effectively.

Beside all these methods, deep CNNs demonstrate superior performance dominating the top accuracy benchmarks in various vision problems[50, 51, 54, 55]. CNN is able to generate powerful features[25, 49]. In our work, we rely mainly on deep CNN structure to extract attribute features.

We train each CNN model to extract binary attribute-specific representation. We propose an enhanced multi-task learning framework to allow visual knowledge sharing between all CNN models. Usually, the structure of the CNN network is big, hence it requires powerful computational power. Concretely, we propose two methods: if the number of attributes is relatively large, we fine-tune each CNN model independently on each semantic attribute to generate attribute-specific feature representations, then we

apply our proposed MTL framework to jointly train the classifiers for attribute predictions.

If the number of attributes is relatively small, we train multi-task CNN models simultaneously together through our MTL framework, where each CNN model is mainly dedicated to learning one binary attribute. This approach is more or less applicable depending on the available hardware resources (CPU/GPU and Memory). The shared visual knowledge of different attribute classes will boost the performance of each individual CNN model and increase its generalization ability.

Among the existing methods in multi-task learning literature, the work in [36] propose a flexible method for feature selection. They introduce a latent task matrix, where all categories are selected to share only the related shared knowledge through this latent matrix. This latent matrix can learn localized features that can be shared among CNN models. While the work in [35] utilizes the side information of the semantic attribute relationships. They used structured sparsity to encourage feature competition between groups and sharing within each of these groups. Unlike the work in [36], we introduce to encode the natural grouping information and in the same time propose the decomposition method to obtain the sharable latent task matrix. This decomposition is important to allow flexible global sharing and competition between groups through learning localized features. Additionally, unlike the work in [35], we have no mutual exclusive pre-assumptions, such that groups may not overlap with each other in terms of attribute members.

We test our method on two popular attribute datasets: Animals with Attributes (AwA)[1] and the Clothing Attributes Dataset [5]. Our results demonstrate the effectiveness of our proposed method compared to standard baselines and other state-of-the-art works. In the case of the Clothing dataset which contains a small number of attributes, we successfully train our multi-task CNN model simultaneously. While because the AwA dataset contains a relatively large number of attributes, we first train each single CNN

model on a target attribute to generate their convolutional feature. Then, we apply our MTL framework on the generated features without fully (end-to-end) back-propagating the errors through the early convolution layers.

Our main contributions in this chapter are summarized as follows: 1) We propose an enhanced deep CNN structure that allows different CNN models to share visual knowledge through our multi-task framework; 2) We propose a new MTL method where we naturally leverage and encode the group information, thus, to encourage attributes in the same group to share features and discourage attributes in different groups to share features. We propose to decompose the model parameters into a latent task matrix and a linear combination weight matrix. The latent task matrix can learn localized features.

The remaining parts of our chapter are summarized as follows: We first discuss some related work in Section 2. Our proposed method for the multi-task CNN model alongside the details of our MTL framework are presented in Section 3. Experiments and the results on two well known attribute prediction benchmarks are demonstrated in Section 4. Finally, we conclude the chapter in Section 5.

3.2 Related Work

Because this work is mainly related to the topics of Semantic attributes, Multi-task learning and Deep CNN, see Section 2.4 ,Section 2.5 and Section 2.1 respectively for detailed reviews. We also briefly review the most recent literature including the following:

3.2.1 Single-label and multi-labels with CNNs:

It is very popular to use CNN for single label prediction and it has been intensively studied [49, 50]. But for multi-labeling case, there are many challenges, as previously discussed in the ‘Attributes and Multi-labeling’ section. Hence, training CNN directly is impractical and infeasible. However, one recent interesting work proposes a kind of work-around solution for the multi-label prediction problem [34]. In this work [34], a shared CNN is

fed with an arbitrary number of object segment hypotheses (image batches), which are extracted or generated by some techniques, like the binarized normed gradients (BING) [166].

The final CNN output results for all of these hypotheses are aggregated by max-pooling to give the final format of the multi-label predictions. Different than their approach, our proposed model keeps the key concept of tagging one image with multiple labels through our multi-task CNN framework. The CNNs in our method are simultaneously trained through MTL and able to share the visual knowledge. Another direction for multi-labeling is proposed by [25], where the CNN is mainly used to generate the activation maps / features; then, they apply SVM for later classifications. In our approach, when the number of attributes is large, we first fine-tune all the needed CNN models, each is mainly dedicated to learning attribute-specific representations. Then we freeze the learning rate inside the convolutional layers. The generated features from them are then passed to the MTL part (in other words, while optimizing the multi-task loss function, we freeze the training of the convolutional layers).

3.2.2 Convex optimization and CNN:

Some interesting work in [167, 168] show that the convex optimization can indeed improve the performance of the highly non-convex CNN models. The authors in [167] propose to modify the last two layers in the CNN model by having a linear combination of many sub-models and then they replace the original loss function one of the convex optimization family. One of their findings is that hinge loss is one of the preferable convex functions that performs well during backpropagation. Another work [168] confirms their finding that using the popular SVM squared hinge loss can improve the final performance after training the CNN.

We further utilize these experimental findings and adopt the squared hinge loss in our MTL framework to jointly optimize all CNN models while allowing them to naturally

Figure 3.2: Our proposed multi-task CNN model. The input image (in the left) with attribute labels are forwarded through the model. Each CNN will predict one binary semantic attribute. The shared latent layer L together with the S layer act as weight matrices of the last fully connected layer. The L layer is a shared latent matrix between all CNN models. Each vector in S is CNN-specific weight matrix. The soft-max and loss layers are replaced by our multi-task squared hinge loss. Group information about attribute relatedness is encoded during the training of the network.

share visual knowledge between attributes. However, our proposed approach is to train multi-task classifier models on deep CNN features for attribute prediction. We also propose to leverage a sharable latent task matrix that can be very informative for generating a full description of the input image in terms of attributes.

3.3 Multi-task CNN Models

Figure 3.2 shows the overall structure of our proposed method. Starting from a raw input image and ending with its attribute predictions. Given a vocabulary of M attributes, each CNN model will learn a binary semantic attribute. After the forward pass in all of the CNN models, the features that are generated from the last convolution layers will be fed into our joint MTL loss layer. In details, the weight parameter matrix that is learned in the loss layer will be decomposed into a latent task matrix and a linear combination matrix. The latent matrix can be seen as a shared fully-connected layer between all the CNN models.

While the linear combination matrix contains the specific information for each CNN model and it can be regarded as a specific last fully-connected layer stacked after the shared latent fully-connected layer. After optimizing the joint loss layer and allow the share of the visual knowledge, each CNN model will take back its error signals through backpropagation in the backward pass. By forwarding the images that are annotated with several attributes, we iteratively train the whole structure until convergence.

We adopt the popular CNN model proposed by Alex Krizhevsky [50], the model consists of 5 convolutions (where pooling and non-linearities layers such as ReLU are stacked in between the convolutions), followed by 2 fully-connected layers and finally the softmax and the loss. We will use the task/CNN model as an interchangeable meaning for classifier because in all cases we employ the same underlying MTL framework. It is important to note that in one of our approaches, the attribute-specific feature learning is on-line, and the optimization of joint MTL loss function will affect the bottom layers

in all CNN models through back-propagation. Thus, any shared knowledge will also be back-propagated to the bottom layers. Meanwhile, in another approach, we learn these attribute-specific features in isolation of optimizing the joint MTL loss function, as training many on-line CNN models on a large number of attributes is impractical.

3.3.1 Sharing the Latent Task Matrix in MTL

Given M semantic attributes, the goal is to learn a binary linear classifier for each of them. Each classifier or task has model parameters, which are denoted by w^m and are dedicated to predicting the corresponding attribute. W is the total classifier weights matrix, which can also be considered as a softmax weight matrix but stacked from all CNN softmax layers. Given N training images, each of them has a label vector Y of $M - dimension$, such that Y_m^i is either $\{1\}$ or $\{-1\}$, indicating whether a specific training image i contains the corresponding m attribute having a value of $\{1\}$ or not $\{-1\}$. Suppose that the output from the last convolution layer in each CNN model forms our input feature vectors, such that each CNN model will generate an attribute-specific training pool. Thus, we will have X_M^N training examples aggregated from all CNN models.

Our decomposition assumption is inspired from the work of [36], where the parameter weight matrix of a classifier can be reconstructed from a number of shared latent tasks and a linear combination of these tasks. Through this decomposition, the simultaneous CNN models can share similar visual localized patterns while performing flexible selection from this shared latent layer. We denote L to be this latent task matrix, and s^m is an attribute-specific linear combination column vector. In total, we have S linear combination matrices for all attribute classifiers.

Now, we want to decompose W into two matrices L and S , as we assume that W is a result of multiplying the shared L latent matrix and the combination matrix S , $W = LS$. To be more specific about each attribute classifier, the weight parameter vector can be formed by multiplying L with the corresponding s^m vector:

$$w^m = Ls^m \tag{3.1}$$

where m is the index of the m -th attribute, $m = \{1, 2, 3 \dots M\}$.

Given the CNN models, our goal is to learn the matrix W which is originally formed by stacking the parameter weight matrices of the last fully-connected layers in each CNN model. The main idea behind our proposed model is to decompose this weight matrix W into two matrices L and S , where the latent L matrix will be a shared layer between all CNN models, S is a linear combination matrix, and each column corresponds to one CNN classification layer.

Through this decomposition, every CNN can share visual patterns with other CNN models through selecting the related feature vectors from the latent task matrix L . All CNN models will collaborate jointly during training to optimize the shared layer. Each CNN predicts whether the image contains the corresponding semantic attribute. The benefit of learning the shared layer through the MTL is that each CNN can leverage the visual knowledge from learning other CNN models even if its training samples are not enough for some CNNs.

3.3.2 Feature Sharing and Competition in MTL

Semantic attribute grouping is proposed in [35], group details are used as discriminative side information. It helps to promote which attribute classifiers are encouraged to share more visual knowledge, due to the group membership privileges. Meanwhile, different groups tend to compete with each other and share less knowledge. Table 3.1 shows some group information examples from the Animal with Attribute (AwA) dataset[1]. Because attributes are naturally grouped, we also propose to encode the grouping information and encourage attributes to share more if they belong to the same groups and compete with each other if they belong to different groups. Our attribute group information is shown in Table 3.3.

Suppose we have M attributes and G groups, where each group contains a variable number of attributes, for example, g_1 contains $[a_1, a_2, \dots, a_C]$ as shown in the left side of

Table 3.1: Examples of attribute groups from AWA dataset [1]

<i>Texture:</i> patches, spots, stripes, furry, hairless, tough-skin
<i>Shape:</i> big, small, bulbous, lean
<i>Colors:</i> black, white, blue, brown, gray, orange, red, yellow
<i>Character:</i> fierce, timid, smart, group, solitary, nest-spot, domestic

Figure 3.2, and each group can have a maximum of M attributes. Our model have no restrictions on the intra-group attributes. Even if two groups have the same attribute, the latent layer will still be able to learn localized features. However, in our experiments, we follow and use the existing grouping information provided in the benchmark datasets.

The groups are mutually exclusive (one attribute can be seen only in one group). Solving the problem while having overlapped groups requires some more complicated methods as discussed in [169], which is computationally expensive. Typically, structured learning methods like group lasso [170] are applied in many areas employing such grouping information. Knowing some prior information about the statistical structure of features usually help. In our MTL framework, we utilize the information of groups and also adopt a flexible decomposition to learn different localized features through the latent matrix. We follow the work in [35], as they also applied such group information of attributes.

In our framework, we use different regularizations strategies to calibrate and balance between feature sharing of intra-group attribute classifiers and feature competition between inter-group attribute classifiers. The idea is that when applying the L_1 norm as $\sum_{m=1}^M \|w\|_1$ [171], it will consequently encourage the sparsity on both rows (features) and columns (tasks) of W . The effect of sparsity on the rows will generate a competition scenario between tasks; meanwhile, the sparsity effect on the columns will generate sparse vectors. In contrast, applying the L_{21} norm such as $\sum_{d=1}^D \|w_d\|_2$; this will encourage

sharing the features among all tasks. This L_{21} norm is known as all-sharing norm as it promotes sharing among all tasks. Also proposed in the joint feature selection through the multi-task lasso approach [41]. Here D is the feature dimension space, since it is extracted from the previous layer, D is 4096. Through the L_{21} norm, the effect is similar as applying the L_1 norm on the zipped column-wise output of the L_{21} , which forces tasks to select only dimensions that are sharable by other tasks; hence it is as a way to encourage feature sharing. As a middle solution [35], if the semantic group information is used when applying the L_{21} norm, the competition can be applied on the groups (between groups); meanwhile, the sharing can be applied inside each group (between the tasks within each group).

In our framework, we encourage intra-group feature sharing and inter-group feature competition through adapting the L_{21} norm. Thus, we apply this on the vector set $s = \sum_{k=1}^K \sum_g^G \|s_k^g\|_2$ [35, 41], where K is the number of latent tasks (latent dimension space) and G is the number of groups, where each group contains a certain number of attributes. Specifically, s_k^g is a column vector corresponding to a specific attribute classification layer, and given a certain latent task dimension, s will contain all the intra-group attribute vector sets. This will encourage attribute classifiers to share specific parts of the latent dimension space, if they only belong to the same group. Meanwhile, different groups will compete with each other as each of them tries to learn a specific part from the latent dimension space. Additionally, the L_1 norm is applied on the latent matrix $\|L\|_1$ [36, 171], to learn more localized visual patterns.

3.3.3 Formulations of the Multi-task CNN model

Given the above discussions about decomposing W and by applying regularization alongside grouping information for better feature competition and sharing, we propose the following objective function:

$$\min_{L, S} \sum_{m=1}^M \sum_{i=1}^{N_m} \frac{1}{2} [\max(0, 1 - Y_m^i (L s^m)^T X_m^i)]^2 + \mu \sum_{k=1}^K \sum_{g=1}^G \|s_k^g\|_2 + \gamma \|L\|_1 + \lambda \|L\|_F^2 \quad (3.2)$$

This is the typical squared hinge loss function, in addition to our extra regularizations terms. For the m -th attribute category, we denote its model parameter as Ls^m and the corresponding training data is $(X_m^i, Y_m^i)_{i=1}^{N_m} \subset \mathbb{R}^d \times \{-1, +1\}$ ($m = 1, 2, \dots, M$), where N_m is the number of training samples of the m -th attribute, and K is the total latent task dimension space. In the second term and given a specific latent task dimension k , s_k^g is a column vector that contains specific group attributes.

The effect of this term is to continually elaborate on encouraging intra-group attributes to share feature dimensions. Thus, the columns (tasks) in the combination matrix S will share with one another only if they belong to the same group. Such competition between groups is important but however, if there is some overlap between groups (they are not absolutely disjoint), some mitigation may help through the latent matrix L structure, which is mainly regularized to learn more localized features. The L_1 norm is applied on the latent task matrix L to enforce sparsity between hidden tasks. The last term is the Frobenius norm to avoid overfitting. Moreover, with such a configuration of the latent matrix L , an implicit feature grouping is promoted. Namely, the latent tasks will allow finding a subset of the input feature dimensions D , which are useful for related tasks, where their corresponding parameters in the linear combination matrix S are nonzero. Figure 3.3 shows an illustrative example to clarify and aid the explanation on the intra-group feature sharing and inter-group feature competition.

Accordingly, every CNN is responsible for learning better input feature representations. Later in the testing, the input image will be fed into all CNN models to generate different input feature vectors; then the corresponding classifier weight vector will be applied to produce the attribute predictions.

The bottom layers in each CNN model are defined in the same way as the network structure proposed by [50]. As shown in Figure 3.2, every block of CNN has several hidden layers, mainly 5 convolutions. We replace the last 2 fully connected layers, softmax and the loss by our proposed MTL squared maxing hinge loss layer. Nevertheless, when

Figure 3.3: An illustration example of the intra-group feature sharing and inter-group feature competition.

the number of attributes is large, we freeze the training of the bottom layers and optimize the multi-task loss function to predict attributes, using the output features that are generated from the convolution layers in all CNN models.

3.3.4 Optimization Steps

During the training of M CNN models, each of them is responsible for predicting one single attribute. Our goal is to encourage visual knowledge sharing between all CNN models through optimizing the multi-task objective function. The optimized components of W will serve as the last two fully connected layers. The L component is a shared latent layer between all CNN models. The S is a final classification layer that will generate the attribute predictions. Note that each s is learned to select what is needed and related from the sharable matrix L . The generalization ability of each single CNN is improved by leveraging the shared visual knowledge from other attribute classifiers.

Solving the proposed cost function is non-trivial, because it is not jointly convex on either L or S . The work in [165] solves the non-convex regularized function using the block coordinate descent method. The function hence becomes a bi-form convex function. They employ Accelerated Proximal Gradient Descent (APG) to optimize both L and S

in an alternating manner. Specifically, if S is fixed, the function becomes convex over L , and optimizing it by APG can solve this state and handle the non-smooth effect of the l_1 norm. Likewise, if L is fixed, the function becomes convex over S ; in this form of the function and unlike [165], the mixed norm regularizations require re-representing the 2-norm into its dual form as discussed in [35]. Smoothing Proximal Gradient Descent (SPGD) [35, 169, 170] is applied to obtain the optimal solution of S . These optimizations are common in the literature of structured learning, where various regularizations may disturb convexity and smoothness properties of the functions. Algorithm 1 illustrates the main steps that are applied to optimize equation 2.

Furthermore, when L is fixed, the optimization problem is in terms of S and is described as follows:

$$\min_S \sum_{m=1}^M \sum_{i=1}^{N_m} \frac{1}{2} [\max(0, 1 - Y_m^i (Ls^m)^T X_m^i)]^2 + \mu \sum_{k=1}^K \sum_{g=1}^G \|s_k^g\|_2 \quad (3.3)$$

Optimization by SPGD: Chen et al. [169] propose solving the optimization problems that have some mixed-norm penalty over a priori grouping to achieve some form of structure sparsity. The key idea is to introduce a smooth approximation of the objective loss function and solve this approximation function instead of directly optimizing the original objective function. Some work proposes solving non-overlapping groups, as is the case in [35]. Others extend the solution to overlapping groups, as in [169]. We follow the method of approximating the objective function proposed in tree-guided group lasso [170, 172], which is mainly built on the popular group-lasso penalty [173]. We apply the step of squaring the mixed-norm term [35], which is originally suggested in [173]. Usually, squaring before the optimization makes the regularizer positive, which generates a smooth monotonic mapping, preserving the same path of solutions but making the optimization easier. For further details on this approximation, refer to [173].

Now, after fixing S , the optimization problem is in terms of L as follows:

$$\min_L \sum_{m=1}^M \sum_{i=1}^{N_m} \frac{1}{2} [\max(0, 1 - Y_m^i (Ls^m)^T X_m^i)]^2 + \gamma \|L\|_1 + \lambda \|L\|_F^2 \quad (3.4)$$

Table 3.2: Solving the Optimization Problem of Equation —2

Algorithm 1: Solving the Optimization Problem of Equation —2

Input — Generated features from CNN models : X_M^N

Attributes labels with values $\{-1,1\}$: Y_M^N

Output — Combination weight matrix S

Latent tasks matrix L

Overall Model weight matrix W

Step 1 — Fix L and optimize S by SPGD

Solving equation 3 until convergence

Step 2 — Get S from Step 1, and optimize L by APG

Solving equation 4 until convergence

Step 3 — Repeat Step 1 and Step 2

Solving equation 2 until convergence

Optimization by APG: This method which is Accelerated Proximal Method updates the searching point from the last linear combination of two points in each iteration and it usually converges faster [174]. It also handles non-smooth convex functions using proximal operators. The key idea is to rely on a shrinkage operator [165, 174] while updating the search point given the previous point and the gradient of the smooth part of the function (the non-regularized part). We adopt this method to optimize over the L latent matrix because the proximity operator is straightforward and also the non-smooth l_1 norm has been studied extensively [171, 173].

Beside, the proximity operator cannot be computed explicitly in other general learning problems, specially, the mixed-norm regularization term, so we adopt the SPGD while optimizing S . We can optimize L through SPGD by using approximations for both the gradient and the proximity operator, but the APG is more preferable here as it has a relatively higher convergence rate.

During a training epoch, the forward pass in all the CNN models will generate the input features for the MTL loss layer. After optimizing equation 2 using the proposed Algorithm 1 (see Table 3.2), the output is the overall model weight matrix W , where

Table 3.3: Grouping information used in Clothing dataset[5]

Group	Attributes
Colors	black, blue, brown, cyan, gray, green, many, red, purple, white, yellow
Patterns	floral, graphics, plaid, solid, stripe, spot
Cloth-parts	necktie, scarf, placket, collar
Appearance	skin-exposure, gender

each column in W is actually a specific corresponding CNN model and is taken back in the backward pass alongside the gradients with respect to its input. W is reconstructed using the optimal solutions of L and S , where knowledge sharing is already explored through MTL between all the CNN models via L .

3.4 Experiments and Results

3.4.1 Datasets and Grouping

We conduct our experiments and evaluate our models on two datasets: **Clothing Attributes Dataset**:

This dataset is collected in the work [5]. It contains 1856 images and 23 binary attributes, as well as 3 multi-class value attributes. The ground-truth is provided on each input image, where each image is annotated against all the attributes. We only consider the binary attributes and ignore the multi-class value attributes. The purpose behind such clothing attributes is to provide better clothing recognition. We train our multi-task CNN models to predict the attributes in this dataset. Because no grouping information is suggested in this dataset, we follow the natural grouping strategy as proposed in other attribute datasets similar to [1]. In Table 3.3, we show the details of our proposed attribute grouping on the Clothing Attributes dataset.

AwA Dataset:

The Animals with Attributes dataset is collected in [1], the main purpose originally is to

apply transfer learning and zero-shot recognition [175]. However, it is one of the most popular benchmarks for binary attribute prediction. It consists of 30475 images of 50 animal classes. The attribute per class matrix is provided; thus, the attribute annotations are on each animal class. It provides 85 binary attributes for each class. This dataset has 9 groups: colors, textures, shapes, activity, behavior, nutrition, character, habitat and body parts [35]; Table 3.1 shows some attributes in some of these groups.

3.4.2 Attribute Prediction Accuracy

We conduct different experiments on both datasets. For the clothing dataset, we train multiple CNN models simultaneously. The accuracy of attribute predictions is calculated against the provided ground truth in the dataset. In Table 3.4, S-extract refers to a simple setting. Here, we directly use a pre-trained model of CNN [176] for feature extraction, and then we train single SVM tasks for attribute prediction. Similarity in M-extract, but here we train our MTL framework on these CNN extracted features. S-CNN refers to a single-task CNN model, where we fine-tuned each individual model of CNN to predict one single attribute. M-CNN refers to our MTL framework without encoding the group information [36], and MG-CNN is our proposed whole MTL framework with group encodings.

In MG-CNN, we jointly train all the CNN models with our MTL framework. CF refers to the combined features model with no pose baseline [5], while CRF refers to the state-of-the-art method proposed by [5]. Our model outperforms the state-of-the-art results in [5]. We notice, though, that the overall improvement margin over the single CNN task models is relatively small compared to our results in AwA (see Table 3.5). This is because the accuracy results are already quite high and thus hard to improve further.

We conduct another experiment on the AwA dataset. We fine-tuned each single CNN models separately on each attribute. Then given the input images, the learned

Table 3.4: The prediction accuracy of the semantic binary attributes before sharing the L layer, after sharing and other methods on the clothing dataset [12]. We refer to G1 as the group of Color attributes, G2 refers to the Pattern group, G3 refers to Cloth-parts awhile G4 refers to the Appearance group. MG-CNN is our overall proposed multi-task CNN model. The higher, the better.

Method	G1(%)	G2(%)	G3(%)	G4(%)	Total(%)
S-extract	81.84	82.07	67.51	69.25	78.31
M-extract	84.98	89.89	81.41	81.03	85.29
S-CNN	90.50	92.90	87.00	89.57	90.43
M-CNN	91.72	94.26	87.96	91.51	91.70
MG-CNN	93.12	95.37	88.65	91.93	92.82
CF[177]	81.00	82.08	77.63	78.50	80.48
CRF[5]	85.00	84.33	81.25	82.50	83.95

models are basically used to extract attribute-specific features and not optimized. In other words, we freeze the training of the bottom layers in all CNN models and train only our MTL layer. AwA dataset has large number of attributes, thus, simultaneous training of all CNN models is very expensive. Note that we initialize all our CNNs models with pre-trained models on ImageNet dataset. However, it is also important to note that here the fine-tuning on AwA dataset didn't add much practical difference in terms of accuracy, perhaps due to the fact that AwA and ImageNet datasets have already an overlap of approximately 17 object categories. Some work have also confirm with our results [178], in which they even train the CNN model from scratch to classify objects on the AwA dataset; however, they noticed that using the pre-trained CNN model on the ImageNet dataset directly or fine-tuning the model on AwA will in both cases give the same attribute prediction results.

However, our MTL framework outperforms the single-tasks by a large margin; Table 3.5 shows the performance of our method compared with other baseline methods, where the prediction accuracy is in terms of the mean average over all of the attributes. Compared with previous state-of-the-art results, which are approximately 75% [178], our

Table 3.5: Attribute prediction scores of our Multi-task CNN model compared with other methods on AwA [1]. The higher, the better. (mean average precision)

Tasks	Prediction Score(%)
lasso [171]	61.75
l_{21} all-sharing [41]	60.21
l_2 regression loss	66.87
decorrelated [35]	64.80
category-trained CNN[178]	74.89
single CNN	75.37
multi-task CNN (ours)	81.19

Table 3.6: The group-level accuracy results of our Multi-task framework on AwA [1]. The higher, the better.

Groups	# Attributes	Single CNN(%)	Our Multi-task CNN(%)
Colors	9	76.91	82.28
Texture	6	76.16	82.44
Shape	4	61.67	72.68
Body-Parts	18	75.93	81.82
Activity	10	82.22	85.4
Behavior	5	72.78	74.96
Nutrition	12	74.76	82.67
Habitat	15	80.21	84.72
Character	7	62.01	70.52
Total	85	75.37	81.19

trained MTL CNN models on attributes outperform it by a large margin. Additionally, Table 3.6 shows the accuracy results in terms of mean average precision over each group of attributes (group-level accuracy), before and after applying our multi-task framework. Initialization of the pre-trained model on Imagenet [50] again is used throughout our experiments. Figure 3.4 also shows a number of misclassified test samples from single-task classifiers, which our multi-task classifiers classified correctly.

Orange Attribute	Examples from 'Tiger' class		Examples from 'Giraffe' class		Examples from 'Ox' class

	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes
Spots Attribute	Examples from 'Killer-Wale' class		Examples from 'Seal' class		Examples from 'Giant-panda' class

	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes
Tree Attribute	Examples from 'Squirrel' class		Examples from 'Leopard' class		Examples from 'Bat' class

	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes	Single-task : No Multi-task : Yes Ground-truth : Yes
Vegetation Attribute	Examples from 'Chihuahua' class		Examples from 'Persian+cat' class		Examples from 'Beaver' class

	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No
Red Attribute	Examples from Clothing dataset images

	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No
Stripe Attribute	Examples from Clothing dataset images

	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No	Single-task : Yes Multi-task : No Ground-truth : No

Figure 3.4: Qualitative examples where single-task CNN classifiers misclassify them and our Multi-task CNN classifiers are able to correctly classify these samples. The first 4 rows have samples from AWA dataset [1]. The last 2 rows are samples from Clothing attributes dataset [5]. Yes/No indicates whether the attribute is presented or absent in the image.

3.4.3 Implementation Details

CNN model training: We train each CNN model for about 50 epochs. We initialize all our CNN models with the pre-trained models on ImageNet [50] and fine-tune it on the target attribute labels. We resize and normalize all the input images to 256×256 and subtract the mean from them. We use the data augmentation strategy as in [50] to train our model on the Clothing dataset (since it has a relatively small number of training images). We fix the weight decay to 0.0005, the momentum is set to 0.9, and the learning rate is initialized to 0.01 and reduced manually during training (we follow the same heuristic in [176]).

Multi-task optimization: We initialize W by stacking the last fully-connected layers from all our pre-trained CNN models. We perform Singular Value Decomposition (SVD) on W similar to the work in [36] in order to obtain an initialization for L , while S is randomly initialized. The other model parameter values are either selected experimentally or we follow the typical heuristics and strategies in [35, 36, 50]. The latent tasks number is set to relatively high which is 2048. Note that the original feature dimension is 4096. In our multi-task part (see equation 2), the latent task $\lambda \|L\|_F^2$ regularization parameter λ is set to 0.4, and the other two parameters γ and μ are best validated in each dataset experiments with held out unseen attribute data.

We conduct our experiments on NVIDIA TK40 16GB GPU and the overall training time including the CNN models and MTL is approximately 1.5 days for the Clothing dataset. While the testing/inference time including feature extraction from all CNN models is approximately 50 minutes (sequential extraction from models – one by one and not in parallel, where the time needed to extract features in each model is about 1.5 minutes/1000 images). If more attribute CNN models are engaged, the time will eventually increase. For the AwA dataset, we divide the training images into several sets which is about 8, 5 are used for training and 3 for testing (each set contains 3000 images). The total training time is approximately 2 weeks. Because we noticed that

the major performance improvement was mainly from training our MTL part and not from CNN fine-tuning, we re-conducted the experiment and freeze the training of all the convolution bottom layers and train only the MTL layer. In our new experiment, we saved big amount of training time, as it now only takes about 13 hours to train the last two layers in all CNN models alongside our MTL cost function (13 hours on one training set and about 2.5 days to complete training on all sets).

3.5 Chapter Summary

In this chapter, we introduce an enhanced deep CNN models that are jointly trained in new multi-task learning framework to better predict semantic binary attributes. Our proposed multi-task CNN model allow sharing of visual knowledge between different CNNs, thus, under-sampled CNN can benefit from others through the sharing and generalize better. In our MTL framework, we propose to encode the semantic group information to encourage more sharing between attributes in the same group. We also propose to decompose the overall model parameters into a latent task matrix and a linear combination matrix. The purpose from this decomposition is to share the latent task matrix among all classifiers. The latent task matrix can effectively learn localized feature patterns. Sharing the latent features in the latent task matrix help the under-sampled classifier to generalize better through leveraging and selecting important visual knowledge from this sharable latent layer. Our experiments on both attribute benchmark datasets show that our learned multi-task CNN classifiers easily outperform the previous single-task classifiers and achieve competitive performance to otehr state-of-the-art works.

Chapter 4

Multi-modal Recurrent Neural Networks with Information Transfer Layers for Indoor Scene Labeling

In our previous chapter, we improve deep CNN and allow multiple CNN models to share visual knowledge for attribute prediction task. In this chapter, we tackle more detailed task which is indoor RGB-D scene segmentation. This task is a typical segmentation task where the scene image is annotated semantically on pixel-level, however, it is usually addressed as multi-modal problem as there are two data modalities to be processed: RGB color channels and depth maps. In this chapter, we propose a new method called multi-modal RNNs where we improve the RNN model and allow it to share visual knowledge across multiple modalities. Our proposed model is optimized to classify image pixels given two input sources where we dedicate an RNN to process each modality and simultaneously performs training of the RNNs that are crossly connected through information transfer layers. The transfer layers are learned to adaptively extract relevant across-modality features. Each RNN model learns its representations from its own previous hidden states and transferred patterns from the other RNNs previous hidden states, thus, both model-specific and cross-modality features are retained. We exploit the structure of quad-directional 2D-RNNs to model the short and long range contextual information in the input image. We test our multi-modal RNNs model on popular RGB-D benchmarks and

show it outperforms previous methods significantly and achieves competitive results with other state-of-the-art methods.

4.1 Introduction

Data comes from different sources and in different forms; images, videos, texts and audios. Each of which may complement the other in information content. Thus, multiple data modalities are usually more informative for a task than single data modality. With the enormous availability of various electronic and digital multimedia devices, huge volumes of multi-modal data contents are being generated on the Internet daily.

However, for real-world applications, these modalities should be first well integrated and appropriately fused to have more comprehensive information contents. Many methods have been developed in multi-modal learning to exploit both the different characteristics as well as the shared relationships between different data modalities in order to perform various tasks [179–183]. One of the rigid milestones in developing many visual-based data applications is Scene understanding. It is mainly applied to understand the contents of an image or video prior performing the target task (e.g. large-scale video retrieval [182]).

Imperatively, Scene labeling (i.e; semantic segmentation or image parsing) is a crucial part of understanding an outdoor or indoor captured scene image. The task here is to classify each pixel into its semantic category (belonging to some object or stuff) in an input scene image.

In this chapter, we tackle the problem of RGB-D indoor scene labeling where we process two different data modalities; RGB channels and Depth planes. Indoor RGB-D scene labeling is one of the most challenging visual classification problems [2, 13]. Many applications are build on understanding the surrounding scenes, e.g. social behavior understanding [184] and objects detection and recognition [185].

This problem is usually addressed as a multi-modal learning problem where the task is to exploit and fuse both RGB and Depth modalities to better label each pixel. It is a challenging classification problem since the indoor scene image tends to contain multiple messy objects. These objects may also have variations due to factors affecting their appearance and geometry in the image. Hence, one of the key strategies that is useful for classification is to leverage the neighborhood/contextual information for each pixel. Typically, the feature representation of a pixel is extracted from a local patch (cropped from the scene image) containing that target pixel and used for classification. Long-range/global contextual information (distant image patches) are important as well for local pixel classification. However, both local and global contextual information should be utilized adequately to maintain a good balance between the discriminative features and the abstract/top-level features of that pixel feature representation.

Recently, Recurrent Neural Networks (RNN) has been shown to be very successful on encoding contextual information into local feature representations [141, 186, 187]. Recurrent models have feedback connections so that the current state is engaged in the calculation of the future state. Thus, RNN is effectively used in tasks which require modeling the long and short-range dependency within the input sequence, e.g. speech recognition and natural language processing [4, 58–60].

Traditionally, RNN is only used for single modality signals. In this chapter, we introduce a new multi-modal RNNs method which models contextual information into local representations in multi-modal RGB-D data. In our work, we first train Convolution Neural Networks [188] (CNNs) to extract features from local RGB-D image patches. These convolutional local features form the input to our multi-modal RNNs to further contextualize them. Our model can be easily extended to perform prediction tasks considering more modalities (> 2). To the best of our knowledge, no work has addressed the multimodal problem of indoor RGB-D scene labeling using RNN based methods before.

Our new multi-modal RNNs method is built based on the basic quad-directional 2D-RNNs structures [4, 187]. It has two RNNs, one of which is assigned to learn the

representations of a single input modality (either RGB or Depth). Beyond, we develop information transfer layers to connect the two RNNs, which learn to select and transfer relevant patterns from one modality to the other modality, and vice versa. Then for every patch, during the process of learning the RNN hidden representations for one modality, our method does not only encode the contextual information within its own modality, but also learns to encode relevant contextual patterns from the other modality. As a result, our method can learn powerful context-aware and multi-modal features for local pixel representations.

Our method is different from existing deep multi-modal learning methods [42–45]. They usually concatenate inputs at the beginning or concatenate learned middle level features to extract high-level common features as the representations of the multi-modal data. These methods mainly focus on discovering common patterns between different modalities. However, some modality specific information is high discriminative, e.g., the texture patterns of the RGB channels, and is missed in such methods. In contrast, our model retains modality-specific information by assigning an RNN model to learn features from each modality.

Our method also allows sharing information between modalities by using the information transfer layers to adaptively transfer relevant cross-modality patterns. Our method outperforms the baseline that concatenates RGB-D data as the input of RNN models. Additionally, our model is trained end-to-end and the transfer layers are learned to extract relevant across-modality information for each patch of the image. We perform experiments on two popular RGB-D benchmarks, the NYU V1 and V2 and achieve comparable performance with other state-of-the-art methods on them.

For detailed review about the most related work on RGB-D scene labeling, multi-model learning and RNNs, see Sections 2.6, 2.7 and 2.2 respectively. The remaining parts of this chapter are summarized as follows: We first discuss the related work in Section 2. Our proposed model alongside the framework details is presented in Section

3. Experiments on popular RGB-D benchmarks and results are demonstrated in Section
4. Finally, we conclude the chapter in Section 5.

4.2 Multi-modal RNNs Framework

We first extract convolutional features from local RGB-D patches using our trained CNN models. Then, our multimodal-RNNs are developed to further learn context-aware and multimodal features based on the convolutional features. Afterwards, a softmax classifier is trained to classify each patch into its semantic category. Different from traditional single modality RNN, our multimodal-RNNs also have transfer layers to learn to extract relevant contextual information across both modalities at each time step. Below, we first introduce the traditional RNNs.

1D-RNN and 2D-RNN: The popular Elman-type 1D-RNN [189] and its 2D version are designed to capture the dynamic behavior of the signal over time, so that the hidden representations can capture the contextual information from the first time step until the current time step. Its forward pass is formulated as the following:

$$\begin{aligned} h^{(t)} &= f(Ux^{(t)} + Wh^{(t-1)} + b) \\ y^{(t)} &= g(Vh^{(t)} + c) \end{aligned} \quad (4.1)$$

where $x^{(t)}$, $y^{(t)}$ and $h^{(t)}$ are the input, output and hidden neurons at the time t respectively. The functions $f(\cdot)$ and $g(\cdot)$ are element-wise non-linear functions with bias terms b and c , and the matrices U , W and V are the input to hidden, hidden to hidden and hidden to output weights, respectively. The 2D-RNN [4, 186, 187] is generalized from the 1D-RNN so that the data propagation will come from two dimensional neurons instead of one, thus the formulation becomes:

$$\begin{aligned} h^{(i,j)} &= f(Ux^{(i,j)} + Wh^{(i-1,j)} + Wh^{(i,j-1)} + b) \\ y^{(i,j)} &= g(Vh^{(i,j)} + c) \end{aligned} \quad (4.2)$$

We can notice now the propagation is in a 2D-plane from top left regions and continues to flow until the end of the 2D sequence (in our case the image patch sequence), where (i, j) denotes the location of pixels or patches.

Figure 4.1: 2D-RNN structure is a generalization of 1D-RNN. A visualization of processing patches in 2D image is shown. The scan is only presented in one direction (top-left). Here, the previous hidden states from the top and the left sided are encoded into the current hidden state, i.e. the previous top and left neighbored hidden state information are engaged in calculation of the current network state alongside the current patch at the same location. An image is approximated through four directions: top-left, bottom-left, top-right and bottom-right accordingly. Thus, we use the quad-directional 2D design to accumulate information from all sides.

Figure 4.2: A conceptual illustration to show our principal design in our proposed multi-modal 2D-RNNs. This illustration is based on simple 1D-RNNs and intended to clarify the design structure of two crossly-connected 1D-RNNs through the introduced transfer layers. In our model, since we process 2D input signals (images) we use 2D-RNNs instead of 1D-RNN to retain the spacial dependencies within each scene image.

In this work, we adapt 2D-RNN to learn hidden representations for local RGB-D image patches. Figure 4.1 shows one scan direction using 2D-RNN, which scans only the top-left sequence. Scanning the image in only one direction leaves some patches in the top-left sequence without being informed of the contextual information from the bottom-right patches during the forward-pass of the testing images. We approximate images using four directional 2D sequences of patches following the work [187]. The other three directions are top-right, bottom-left and bottom-right sequences. We combine the features that are learned from these four directions to obtain the final features of the image patches.

4.2.1 Multimodal RNNs via Information Transfer Layers

Traditional RNNs are developed to model single modality signals. In this work, we extend RNNs to represent RGB-D signal by our multimodal-RNNs, where we have a pair of single RNNs and each of them is assigned to process one modality (either RGB or Depth). Besides, we propose a transfer layer to connect the hidden planes in one RNN model and the other RNN model’s hidden planes and vice versa. This transfer layers will learn to adaptively extract relevant patterns from one modality to enhance the feature representations of the other modality. If the Depth-RNN is processing the Depth patch in the sequence at location (i, j) , the other RNN, which is RGB-RNN, will be processing the corresponding RGB patch at the same location. The Depth-RNN is also fed with the processed hidden state values obtained from the RGB-RNN and vice versa. The RGB-D data flows concurrently in both models where their internal processing clocks are synchronized, so that at each time step we process a pair of RGB and Depth local patches simultaneously.

The architecture of our model is summarized in Figure 4.3. It is an end-to-end learning framework. Recurrent layers and transfer layers are automatically learned to maximize the labeling performance on the training RGB-D data. Compared to the baseline which

concatenates RGB-D data and thus mixes the multimodality information (both relevant and irrelevant), our method retains modality-specific information and only shares relevant cross-modality information.

Given one 2D direction in the RNN (the first hidden plane from the quad-directional hidden planes), as shown in Figure 4.3, the current state of the network that is being processed at (i, j) depends basically on four main previous states. Two are obtained from the network itself and the others are obtained through the transfer layers from the previous states processed in the other modality network, in addition to the input patch features from either the RGB or the depth images at location (i, j) . Both networks are synchronized and process the input modalities simultaneously.

Given an RGB image I_c where c refers to ‘color’ and is processed by RGB-RNN and a depth image I_d where d refers to ‘depth’ and is processed by Depth-RNN, in this work, we first extract multiple patches from the images and generate their corresponding convolutional feature vectors to form the input to our multimodal-RNNs model. We denote the corresponding convolutional feature maps as x_c or x_d for each patch in I_c or I_d . Concretely, the forward propagation formulation to process one hidden plane out of the four (quad) directional hidden planes is as the following:

$$\begin{aligned}
 h_c^{(i,j)} &= f(U_c x_c^{(i,j)} + W_c h_c^{(i-1,j)} + W_c h_c^{(i,j-1)} \\
 &\quad + S_c h_d^{(i-1,j)} + S_c h_d^{(i,j-1)} + b_c) \\
 h_d^{(i,j)} &= f(U_d x_d^{(i,j)} + W_d h_d^{(i-1,j)} + W_d h_d^{(i,j-1)} \\
 &\quad + S_d h_c^{(i-1,j)} + S_d h_c^{(i,j-1)} + b_d) \\
 z_c^{(i,j)} &= g(V_c h_c^{(i,j)} + c_c) \\
 z_d^{(i,j)} &= g(V_d h_d^{(i,j)} + c_d)
 \end{aligned} \tag{4.3}$$

where $x_c^{(i,j)}$ is a feature vector of a certain patch at location (i, j) in the RGB image and $x_d^{(i,j)}$ is a feature vector of a certain patch at location (i, j) in the Depth image. $h_c^{(i,j)}$ and $h_d^{(i,j)}$ are the hidden states inside each RGB-RNN and Depth-RNN respectively. The

Figure 4.3: An abstract overview of our multimodal-RNNs model which consists mainly of an RGB-RNN and a Depth-RNN. The RGB-RNN will process a patch from the RGB input plane in addition to its own previous hidden states and the selections made by the transfer layer from the previous hidden states of the Depth-RNN. The Depth-RNN will also process the corresponding patch from the Depth input plane in addition to its own previous hidden states and the selections made through the transfer layer from previous hidden states of RGB-RNN. For the simplicity in this figure, we only show one hidden plane from the quad-directional RNN hidden planes. Every hidden plane is crossly connected to its peer in the multimodal-RNNs through information transfer layers.

weight matrices U_c and U_d are responsible for input-hidden mapping in RGB and Depth modalities respectively. In the other hand, here we have two types of hidden-hidden transformation matrices; within-modality hidden-hidden transformation and across-modality hidden-hidden transformation. W_c and W_d are within-modality hidden-hidden mapping inside the RGB-RNN and the Depth-RNN respectively. Meanwhile, $\mathbf{S}_c$ is a transformation weight matrix to transform features from the Depth to RGB hidden states (from the Depth-RNN modality hidden state to the RGB-RNN modality hidden state). $\mathbf{S}_d$ is a transformation weight matrix to transform features from the RGB to Depth hidden states (from the RGB-RNN modality hidden state to the Depth-RNN modality hidden state), i.e. these weight matrices act as the **information transfer layers that crossly connect the RGB and Depth hidden states**. Ultimately, the V_c and V_d are the hidden-output transformation matrices in each corresponding modality.

$\mathbf{S}_c$ and $\mathbf{S}_d$ are learnt to extract shared patterns between the modalities. Notice that the weight matrix that transforms from the top side hidden state in the RGB-RNN $h_c^{(i-1,j)}$ and the weight matrix that transforms from the left side hidden state $h_c^{(i,j-1)}$ are shared which is W_c . Similarly, W_d in the Depth-RNN is shared (transforms from $h_d^{(i-1,j)}$ and from $h_d^{(i,j-1)}$). Also the case in the transfer layers; $\mathbf{S}_c$ and $\mathbf{S}_d$ are shared. In detail, $\mathbf{S}_c$ transforms from $h_d^{(i-1,j)}$ and from $h_d^{(i,j-1)}$. While $\mathbf{S}_d$ transforms from $h_c^{(i-1,j)}$ and from $h_c^{(i,j-1)}$. The nonlinear function $f(\cdot)$ is ReLU $\max(x, 0)$ in our implementation. The function $g(\cdot)$ is the typical softmax and c_c and c_d are biases.

Since we adopt the quad-directions, the forward pass is similar to Equation 4.3 and in addition to the remaining directions beside the top-left sequence, i.e. top-right, bottom-left and bottom-right. To facilitate readability of the equation and to easily distinguish between all quad/four directions; we will use arrow notations to represent different directions. $\searrow$ refers to the top-left processing sequence, $\swarrow$ indicates the top-right sequence, $\nearrow$ is the bottom-left and finally $\nwarrow$ is the bottom-right sequence. Now, the full model

forward propagation pass in the RGB-RNN becomes:

$$\begin{aligned}
h_c^{(i,j)} \searrow &= f(U_c \searrow x_c^{(i,j)} + W_c \searrow h_c^{(i-1,j)} \searrow + W_c \searrow h_c^{(i,j-1)} \searrow \\
&\quad + S_c \searrow h_d^{(i-1,j)} \searrow + S_c \searrow h_d^{(i,j-1)} \searrow + b_c \searrow) \\
h_c^{(i,j)} \swarrow &= f(U_c \swarrow x_c^{(i,j)} + W_c \swarrow h_c^{(i-1,j)} \swarrow + W_c \swarrow h_c^{(i,j+1)} \swarrow \\
&\quad + S_c \swarrow h_d^{(i-1,j)} \swarrow + S_c \swarrow h_d^{(i,j+1)} \swarrow + b_c \swarrow) \\
h_c^{(i,j)} \nearrow &= f(U_c \nearrow x_c^{(i,j)} + W_c \nearrow h_c^{(i+1,j)} \nearrow + W_c \nearrow h_c^{(i,j-1)} \nearrow \\
&\quad + S_c \nearrow h_d^{(i+1,j)} \nearrow + S_c \nearrow h_d^{(i,j-1)} \nearrow + b_c \nearrow) \\
h_c^{(i,j)} \nwarrow &= f(U_c \nwarrow x_c^{(i,j)} + W_c \nwarrow h_c^{(i+1,j)} \nwarrow + W_c \nwarrow h_c^{(i,j+1)} \nwarrow \\
&\quad + S_c \nwarrow h_d^{(i+1,j)} \nwarrow + S_c \nwarrow h_d^{(i,j+1)} \nwarrow + b_c \nwarrow) \\
z_c^{(i,j)} &= g(V_c \searrow h_c^{(i,j)} \searrow + V_c \swarrow h_c^{(i,j)} \swarrow + V_c \nearrow h_c^{(i,j)} \nearrow \\
&\quad + V_c \nwarrow h_c^{(i,j)} \nwarrow + c_c)
\end{aligned} \tag{4.4}$$

As mentioned, we use four quad-directional 2D-RNN to approximate each image. The arrows $\searrow, \swarrow, \nearrow, \nwarrow$ indicate the quad-directions (top-left, top-right, bottom-left and bottom-right), and the $h_c^{(i,j)} \searrow, h_c^{(i,j)} \swarrow, h_c^{(i,j)} \nearrow$ and $h_c^{(i,j)} \nwarrow$ are the corresponding four hidden planes. Each hidden plane has its own weight matrices besides the introduced transfer layers S_c , e.g. $U_c \searrow$ (input-hidden mapping), $W_c \searrow$ (hidden-hidden mapping), $S_c \searrow$ (the transfer layer from Depth-RNN hidden plane to RGB-RNN hidden plane), $V_c \searrow$ (hidden-output mapping) and the bias term which is denoted by $b_c \searrow$. Note that the connection between the two hidden planes $h_d \searrow$ and the corresponding $h_c \searrow$ is weighted by the corresponding weight matrix $S_c \searrow$, which is leaned to extract shared patterns between the modalities.

Simultaneously, the forward pass in the Depth-RNN model is as follows:

$$\begin{aligned}
 h_d^{(i,j)} \searrow &= f(U_d \searrow x_d^{(i,j)} + W_d \searrow h_d^{(i-1,j)} \searrow + W_d \searrow h_d^{(i,j-1)} \searrow \\
 &\quad + S_d \searrow h_c^{(i-1,j)} \searrow + S_d \searrow h_c^{(i,j-1)} \searrow + b_d \searrow) \\
 h_d^{(i,j)} \swarrow &= f(U_d \swarrow x_d^{(i,j)} + W_d \swarrow h_d^{(i-1,j)} \swarrow + W_d \swarrow h_d^{(i,j+1)} \swarrow \\
 &\quad + S_d \swarrow h_c^{(i-1,j)} \swarrow + S_d \swarrow h_c^{(i,j+1)} \swarrow + b_d \swarrow) \\
 h_d^{(i,j)} \nearrow &= f(U_d \nearrow x_d^{(i,j)} + W_d \nearrow h_d^{(i+1,j)} \nearrow + W_d \nearrow h_d^{(i,j-1)} \nearrow \\
 &\quad + S_d \nearrow h_c^{(i+1,j)} \nearrow + S_d \nearrow h_c^{(i,j-1)} \nearrow + b_d \nearrow) \\
 h_d^{(i,j)} \nwarrow &= f(U_d \nwarrow x_d^{(i,j)} + W_d \nwarrow h_d^{(i+1,j)} \nwarrow + W_d \nwarrow h_d^{(i,j+1)} \nwarrow \\
 &\quad + S_d \nwarrow h_c^{(i+1,j)} \nwarrow + S_d \nwarrow h_c^{(i,j+1)} \nwarrow + b_d \nwarrow) \\
 z_d^{(i,j)} &= g(V_d \searrow h_d^{(i,j)} \searrow + V_d \swarrow h_d^{(i,j)} \swarrow + V_d \nearrow h_d^{(i,j)} \nearrow \\
 &\quad + V_d \nwarrow h_d^{(i,j)} \nwarrow + c_d)
 \end{aligned} \tag{4.5}$$

where $x_d^{(i,j)}$ is the feature vector of a certain patch at location (i, j) in the depth image. The $h_d^{(i,j)} \searrow$, $h_d^{(i,j)} \nearrow$, $h_d^{(i,j)} \swarrow$ and $h_d^{(i,j)} \nwarrow$ are the quad hidden planes in the Depth-RNN. Each hidden plane accompanies its own weight matrices $U_d \searrow$ (input-hidden mapping), $W_d \searrow$ (hidden-hidden mapping), $S_d \searrow$ (transfer layer from RNN-RGB hidden plane to Depth-RNN hidden plane), $V_d \searrow$ (hidden-output mapping) and its own bias term $b_d \searrow$. The remaining terms in the quad planes are similar to the case of the first hidden plane. The function $f(\cdot)$ is a nonlinear ReLU unit and the function $g(\cdot)$ is the typical softmax and c_d is a bias term.

We can notice that the cross-connections through the transfer layers are applied in each processing direction (four sequences). By this method, the processing of a specific patch will rely on both previous hidden neighbors from its own modality in addition to the other modalities, thus learns more contextually-aware hidden representations of the RGB-D image patches from both modalities.

Our labeling task is a typical supervised classification problem. We aggregate the cross entropy losses from both modalities and calculate the loss for every patch. The error signal of an image is averaged across all the valid patches (those that are semantically

labeled), which is mathematically formulated as the following:

$$L = -\frac{1}{N} \sum_{n=1}^N \sum_{b=1}^B (\log(z_c^n(b)) \delta(y^n = b) + \log(z_d^n(b)) \delta(y^n = b)) \quad (4.6)$$

where $\delta(\cdot)$ is the indicator function, N is the total number of patches and B is the number of semantic classes, y^n is the ground truth label for the RGB-D patch representation, which is the same as that of the center pixel, z_c^n and z_d^n are the class likelihood generated from both the RGB-RNN and the Depth-RNN for the patch representations $\mathbf{x}_c^n$ and $\mathbf{x}_d^n$ respectively, where they are B -dimensional vectors. Note that we ignore the contribution of unlabeled (invalid) patches in the loss calculation.

Optimization of the multimodal-RNNs model: To learn the multimodal RNNs parameters in Equations 4.4 and 4.5, we optimize the objective function in Equation 4.6 with a stochastic gradient-based method. Both RGB-RNN and Depth-RNN are optimized simultaneously using Back Propagation Through Time (BPTT) [189]. We unfold both networks in time and calculate their gradients which are back-propagated at each time step throughout both networks. This is similar to the typical multilayer feed-forward neural network propagation, but with the difference that the weights are shared across time steps defined by the architecture of the recurrent network. The whole model is differentiable and trained end-to-end. The propagation of the gradients through the RGB-RNN and Depth-RNN is simultaneous.

In more detail, we provide the first plane (top-left sequence) backward pass in the RGB-RNN. The backward pass derivations of the remaining sequences (the remaining three planes) can be easily inferred by following similar derivation strategy as we will explain next in the backward pass formulations for the first plane, but with considering the change of the sequence directions (top-right, bottom-left and bottom-right sequences). Similarity, the total derivations for the Depth-RNN are straightforward and exactly the same as for the RGB-RNN but with swapping the notations of c and d and vice versa.

In the RGB-RNN top-left sequence, we generate the gradients of the loss function in Equation 4.6 by deriving it with respect to the model internal parameters, i.e. top-left sequence weight matrices $U_c^{\searrow}$, $W_c^{\searrow}$, $S_c^{\searrow}$, $V_c^{\searrow}$, $b_c^{\searrow}$ and c_c .

Notice that since the weight matrix $W_c^{\searrow}$ is shared between $h_c^{(i-1,j)\searrow}$ and $h_c^{(i,j-1)\searrow}$ and the transfer layer weight matrix $S_c^{\searrow}$ is shared between $h_d^{(i-1,j)\searrow}$ and $h_d^{(i,j-1)\searrow}$, hence we will rewrite the forward pass as the following to further facilitate the understanding of the weight sharing concept and the backward pass:

$$\begin{aligned}\tilde{h}_c^{\searrow} &= h_c^{(i-1,j)\searrow} + h_c^{(i,j-1)\searrow} \\ \tilde{h}_d^{\searrow} &= h_d^{(i-1,j)\searrow} + h_d^{(i,j-1)\searrow} \\ h_c^{(i,j)\searrow} &= f(U_c^{\searrow}x_c^{(i,j)} + W_c^{\searrow}\tilde{h}_c^{\searrow} + S_c^{\searrow}\tilde{h}_d^{\searrow} + b_c^{\searrow}) \\ z_c^{(i,j)} &= g(V_c^{\searrow}h_c^{(i,j)\searrow} + c_c)\end{aligned}\tag{4.7}$$

Notice that the other remaining terms that present originally in Equation 4.4 (i.e. $V_c^{\swarrow}h_c^{(i,j)\swarrow}$, $V_c^{\nearrow}h_c^{(i,j)\nearrow}$, and $V_c^{\nwarrow}h_c^{(i,j)\nwarrow}$) do not engage any of the internal parameters of the top-left sequence, hence we omit them.

Before we formulate the backward pass, Figure 4.4 show an illustration of the forward and backward passes in the top-left plane sequence. Notice that the derivatives which are computed in the backward pass at each hidden state at some specific location (i, j) are processed in the **reverse order** of forward propagation sequence.

For better readability, we ignore the south-east arrow sign $\searrow$ in the following derivations for the top-left sequence.

Notice that now there are **two** types of error signals: **direct** one which is reachable directly through the loss ($\frac{\partial z_c^{(i,j)}}{\partial h_c^{(i,j)}}$) and **indirect** ones, i.e. the error signals that are coming from neighboring future states at $(i+1, j)$ and $(i, j+1)$ locations; ($\frac{\partial z_c^{(i+1,j)}}{\partial h_c^{(i+1,j)}} \frac{\partial h_c^{(i+1,j)}}{\partial h_c^{(i,j)}} + \frac{\partial z_c^{(i,j+1)}}{\partial h_c^{(i,j+1)}} \frac{\partial h_c^{(i,j+1)}}{\partial h_c^{(i,j)}}$).

Concretely, and given that the derivative of loss function L with respect to the Softmax output function g is $g'(\cdot) = \frac{\partial L}{\partial z_c^{(i,j)}(\cdot)} \frac{\partial z_c^{(i,j)}(\cdot)}{\partial g}$, similarity $f'(\cdot) = \frac{\partial h_c^{(i,j)}}{\partial f}$, hence at location

(i, j) the backward pass of the RGB-RNN (derivations w.r.t. all internal parameters) is formulated as the following:

$$\begin{aligned}
\Delta V_c &= g'(z_c^{(i,j)})(h_c^{(i,j)})^T \\
dh_c^{(i,j)} &= V_c^T g'(z_c^{(i,j)}) + W_c^T dh_c^{(i+1,j)} \circ f'(h_c^{(i+1,j)}) \\
&\quad + W_c^T dh_c^{(i,j+1)} \circ f'(h_c^{(i,j+1)}) \\
\Delta U_c &= dh_c^{(i,j)} \circ f'(h_c^{(i,j)})(x_c^{(i,j)})^T \\
\Delta W_c &= dh_c^{(i+1,j)} \circ f'(h_c^{(i+1,j)})(h_c^{(i,j)})^T \\
&\quad + dh_c^{(i,j+1)} \circ f'(h_c^{(i,j+1)})(h_c^{(i,j)})^T \\
\Delta \mathbf{S}_c &= dh_c^{(i+1,j)} \circ f'(h_c^{(i+1,j)})(h_d^{(i,j)})^T \\
&\quad + dh_c^{(i,j+1)} \circ f'(h_c^{(i,j+1)})(h_d^{(i,j)})^T \\
\Delta b_c &= dh_c^{(i,j)} \circ f'(h_c^{(i,j)}) \\
\Delta c_c &= g'(z_c^{(i,j)})
\end{aligned} \tag{4.8}$$

The term $dh_c^{(i,j)}$ is defined to allow the model to propagate local information internally. The $\circ$ sign is the Hadamard product (element-wise). We can notice through the term $dh_c^{(i,j)}$ that there are two sources of gradients, one is generated directly from the current hidden state and the other generated indirectly from the bottom and right locations (corresponding to the three terms: direct error: $V_c^T g'(z_c^{(i,j)})$, indirect error from neighbor bottom location: $W_c^T dh_c^{(i+1,j)} \circ f'(h_c^{(i+1,j)})$ and indirect from right neighbor location: $W_c^T dh_c^{(i,j+1)} \circ f'(h_c^{(i,j+1)})$). Notice that these two types of error signals (direct and indirect) are due to the weight sharing effect. It is also important to mention that in our derivations we show the error signals coming from the direct bottom and right neighbors, however, in real implementations the equations are recurrently applied to allow propagation of the errors recursively (recurrently) from all potential future neighbors until we reach out the current referenced patch at specific location.

Similarly, the backward passes for the remaining three planes in the RGB-RNN are straightforward to be derived following Equation 4.8 but with paying attention to the

Figure 4.4: An illustration of the forward and backward passes in the top-left plane sequence in both RGB-RNN and Depth-RNN. (Best viewed in color)

directions of the processing sequence. Additionally, the backward pass in the Depth-RNN follows exactly similar derivations but with swapping the notations of c with d and vice versa. Notice that the derivation w.r.t c_c is the same in the other remaining three planes/sequences.

To recap, we adopt BPTT to train both basic quad-directional 2D-RNN (RGB-RNN and Depth-RNN). We use the cross entropy loss in our implementation and the errors are calculated by chain rule as we described previously in detail.

4.3 Experiments and Results

4.3.1 Datasets

We evaluate our model on the benchmark dataset NYU versions 1 and 2 [2, 13]. The NYU V1 dataset contains 2284 RGB-D indoor scene images labeled with around 13 categories. The NYU V2 is also comprised of video sequences from indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect. It contains 1449 densely labeled pairs of aligned RGB and depth images. We follow the settings in [2], where the first task is to predict the pixel label out of four semantic classes: Ground, Furniture, Props and Structure. The second task is to predict the label out of 14 categories. The accuracy is calculated in terms of total pixel-wise, average class-wise and Intersection Over Union (average IOU) among the semantic classes for comparison.

4.3.2 CNNs and RNNs Training

We train our convolutional neural networks (CNNs) on both NYU version 1 and 2 images. We follow the network structure proposed by [187] without considering any spatial information channels (RGB locations). We also didn't perform any hybrid sampling as they mention in their implementation details. The CNN model consists mainly of three convolutional layers (where in between there are max-pooling and ReLU layers) and two last fully-connected layers followed by an B -way softmax loss layer. In more details, conv1($8 \times 8 \times 3$ or 4×32), max pool(2×2), conv2($6 \times 6 \times 32 \times 64$), max pool(2×2), conv3($5 \times 5 \times 64 \times 64$), max pool(2×2), FC1(1024×64) and FC2($64 \times B$). The CNN models are trained on 65×65 patches associated with their centering pixel labels. Each image contains 4661 patches. Each CNN model produces a 64-dimensional vector per patch for each modality. We use these CNN features as the input of our RNN models.

In CNN training, we start by a learning rate of 0.001 and decrease it every 5 epochs by 10 (divided by 10). The momentum is initialized as 0.9 and remains the same throughout the training. We perform the typical normalization as a preprocessing step of images, we

subtract the mean image and divide by the standard deviation. The CNN models nearly converge after 50 epochs (around 6 hours on NVIDIA TK40 GPU), however, we iterate the models in most cases until 100 epochs.

We train our multimodal RNNs using BPTT (back propagation through time). We adapt stochastic gradient descent (SGD) throughout our training. We initialize the learning rate as 0.00001 and decrease it by 0.01 after each epoch. The momentum is set to 0.9. The internal parameter dimension of the networks is set to 64. We apply gradient clipping [190, 191] and set the threshold value to $2 * 10^3$, however, even if the ReLU can potentially cause gradient explosion, it plays a critical role in RNN to mitigate the gradient vanishing problem. Thus, we mainly use ReLU in all of our RNN models. The other RNN model parameters are initialized either by randomly or zeros. Notice that we didn't consider any pixels that have zero labels in our training, we held them out. Our model converges much faster compared to the other competitive baselines. It converged at almost the similar speed of a single RNN model, with the benefit of processing multiple input modalities simultaneously.

4.3.3 Baselines

To show the effectiveness of our proposed model, we develop the following baselines for comparison alongside our proposed model:

- **CNN-RGB:** in this baseline, we train CNN based on RGB images (input is three RGB channels) for label prediction.
- **CNN-Depth:** we train CNN as in 'CNN-RGB' but using Depth images only (input is one Depth channel).
- **CNN-RGBD:** we train CNN as in 'CNN-RGB' with extra Depth images (input is four RGB-D channels) similar to the work presented in [102].

- **RNN-RGB:** in this baseline, we follow the structure of the quad-directional 2D-RNN proposed by [187]. We use the model in ‘CNN-RGB’ to extract RGB features. We only input the RGB features to train the RNN for label prediction.
- **RNN-Depth:** we extract Depth features using the trained model in ‘CNN-Depth’ and use these features to train quad 2D-RNN as in ‘RNN-RGB’. We only input the Depth features to train the RNN for label prediction.
- **RNN-RGBD:** we use the trained model in ‘CNN-RGBD’ for feature extraction. We use these RGB-D features to train quad 2D-RNN similar to ‘RNN-RGB’ baseline to perform label prediction.
- **RNN-Classifiers-Combined:** or as we call it ‘post-fusion’, here we train RGB-RNN and Depth-RNN for label prediction and combine their classification scores on the classifier level. We use the trained models in ‘CNN-RGB’ and ‘CNN-Depth’ for feature extraction.
- **RNN-Features-Combined:** or ‘pre-fusion’, here we use both trained models in ‘CNN-RGB’ and ‘CNN-Depth’ for feature extraction. We concatenate both RGB and Depth features (to form higher dimensional feature vectors) and train one quad 2D-RNN similar to ‘RNN-RGB’ baseline to perform label prediction.
- **RNN-Hiddens-Combined:** or ‘middle-fusion’, we fuse the hidden representations of both RNNs just before classification and train them jointly.
- **Multimodal-RNNs-Ours** in this sitting we implement our proposed multimodal RNNs structure. Here, we have two internal RNN models one is responsible for processing the RGB features and the other is responsible for processing the Depth features. Both models are optimized simultaneously using BPTT where we combine their classification scores to finally obtain the label map per RGB-D image.

- **Multimodal-RNNs-Ours-Multiscale:** similar to our ‘Multimodal-RNNs-Ours’, but in this setting we applied our proposed structure on multiscale convolutional features as proposed by [114]. We follow similar training settings to train multiscale CNN models on different image sizes, then we use them to extract multiscale features. We concatenate these features together to form our final input patch representation to our multimodal-RNNs.

We also compare our results with other state-of-the-art methods.

4.3.4 Results

Given the input RGB images and their corresponding Depth images, we divide each image into non-overlapping patches of size 65×65 . The extracted local CNN features are used as the input of our multimodal RNNs. The RGB-RNN and the Depth-RNN models process RGB and Depth local patch features respectively.

Results on NYU V1 - 13 categories: In this dataset the task is to predict pixel labels out of 12 categories plus an unknown category. Table 4.1 shows the results of our baselines alongside the multi-modal RNNs model and other state-of-the-art methods. RNN models outperform CNN baselines by a large margin. And our multi-modal RNNs further improve the accuracy over the RNN baselines. Our model achieves comparable results with the other state-of-the-art methods.

For example, in the work of Fan et al.[192], their proposed model is somehow similar to ours; however, there is a key major difference between our works; i.e. they propose to fuse the hidden representations of the RNNs into one unified memory, meanwhile our method proposes to transfer information through transfer layers that are crossly-connecting both RNNs hidden representations/states. In terms of qualitative results, our method achieve competitive results to theirs on the NYU-V1 and V2 datasets; as shown in Tables 4.1 and 4.3, respectively).

Algorithm	Pixel Acc	Class Acc	IOU
CNN-RGB	69.21%	56.23%	45.17%
CNN-Depth	58.56%	32.82%	22.21%
CNN-RGBD	69.17%	57.83%	45.24%
RNN-RGB	71.38%	64.43%	52.80%
RNN-Depth	60.67%	46.88%	28.24%
RNN-RGBD	71.09%	67.01%	57.87%
RNN-Features-Combined	72.94%	69.06%	60.54%
RNN-Hiddens-Combined	71.50%	67.00%	55.59%
RNN-Classifiers-Combined	71.28%	66.37%	54.79%
Multimodal-RNNs-Ours	74.68%	72.54%	62.53%
Multimodal-RNNs-Ours-Multiscale	78.89%	75.73%	65.70%
Wang et al.[106]	-	61.71%	-
gradient KDES[103]	-	51.84%	-
color KDES[103]	-	53.27%	-
spin/surface normal KDES[103]	-	40.28%	-
depth gradient KDES[103]	-	53.56%	-
Silberman et al[13]	-	53.00%	-
Fan et al. [192]	78.00%	73.00%	-
Pei et al.[193]	-	50.50%	-
Ren et al.[103]	-	71.40%	-
Eigen et al. [105]	75.40%	66.90%	-

Table 4.1: Results on **NYU V1** dataset [13]. The performance of all baselines and other state-of-the-art methods is measured on total pixel-wise, average class-wise accuracy and average IOU (intersection over union) among 13 semantic classes. The higher the better.

The accuracy comparison with all baselines shows the effectiveness of our method with the proposed transfer layers. Figure 4.7 show some qualitative results generated by our method and the most competitive baseline method ‘RNN-Features-Combined’. Our multi-modal-RNNs model can correctly classify many mis-classifications results compared to the baseline in most cases.

Results on NYU V2 - 4 categories: We also conduct experiments on the NYU V2 dataset. Table 4.2 shows the performance of our multi-modal-RNN model compared with other baselines and state-of-the-art methods. We can notice the improvement gain achieved by RNN models over the CNN models is significant. It is also important to

mention that some other state-of-the-art methods used various combinations of features or complicated graphical models as post processing to reach their accuracy.

For example, Eigen et al. [105] achieve higher results than our model on this task, while we outperform their model on NYU V1 task. Their multi-scale CNN structure is also well designed to address the problem. However, their network appears to be much more complex than ours as they have higher number of computational operations (way more convolutions). But we still believe our work can be potentially orthogonal and complementary to their method if it is trained end-to-end with their method.

Likewise, Couprie et al.[2], use many types of features, including SIFT features, histograms of surface normals, 2D and 3D bounding box dimensions, color histograms, relative depth and their support features. In our model, we only use the transfer layers alongside the basic quad 2D-RNN structures, and can achieve much better performance. Figure 4.8 also show some qualitative results generated by our method and the most competitive baseline method ‘RNN-Features-Combined’. Our multi-modal-RNNs model can correctly classify many mis-classifications results compared to the baseline in most cases.

Results on NYU V2 - 14 categories: On the NYU V2, we evaluate our model to label image pixels with one of 13 categories plus an unknown category. We show the comparison in this task with various baselines and state-of-the-art methods as shown in Table 4.3. This is the most competitive benchmark presented on this dataset. Notice that the work of Cadena and Kosecka [198] used many RGB image and 3D features and formulate the problem in the CRF framework. Meanwhile the work of Wang et al. [106] adapted an existing unsupervised feature learning technique to directly learn features. They stack their basic learning structure to learn hierarchical features. They combined their higher-level features with low-level features and train linear SVM classifiers to perform labeling. Compared to these methods, our model is much simpler and achieve better performance.

Algorithm	Pixel Acc	Class Acc	IOU
CNN-RGB	65.55%	62.07%	45.42%
CNN-Depth	69.61%	65.96%	49.13%
CNN-RGBD	71.60%	69.89%	54.19%
RNN-RGB	68.14%	65.96%	51.05%
RNN-Depth	70.95%	67.76%	52.21%
RNN-RGBD	74.18%	69.99%	56.63%
RNN-Features-Combined	74.36%	72.80%	60.08%
RNN-Hiddens-Combined	73.70%	71.10%	56.00%
RNN-Classifiers-Combined	73.39%	70.64%	55.34%
Multimodal-RNNs-Ours	75.74%	75.01%	62.10%
Multimodal-RNNs-Ours-Multiscale	78.60%	76.69%	65.09%
Wang et al.[106]	-	65.30%	-
Couprie et al. [102]	64.50%	63.50%	-
Stuckler et al.[194]	70.90%	67.00%	-
Khan et al. [195]	69.20%	65.60%	-
Mueller et al.[196]	72.30%	71.90%	-
Gupta et al.[197]	78.00%	-	64.00%
Cadena and Kosecka [198]	-	64.10%	-
Eigen et al. [105]	83.20%	82.00%	-

Table 4.2: Results on **NYU V2** dataset [2]. The performance of all baseline and other state-of-the-art methods is measured on total pixel-wise, average class-wise accuracy and average IOU (intersection over union) among 4 semantic classes: Ground, Furniture, Props and Structure. The higher the better.

Figure 4.5: Per class accuracy comparison between our multimodal RNNs, baselines CNN-RGBD and RNN-RGBD on NYU V2 dataset [2](14 semantic categories).

Figure 4.8 also shows some qualitative results generated by our method and the most competitive baseline method ‘RNN-Features-Combined’. Our multimodal-RNNs model can correctly classify many mis-classifications results compared to the baseline in most cases. We also show our per class accuracy on this sitting, as shown in Figure 4.5. We can notice that the improvement gain of our multimodal RNNs over CNN-RGBD and RNN-RGBD models is significant. This is an evidence that our model can effectively learn powerful context-aware and multimodel features.

We also study the effect of increasing the dimensionality of the internal hidden layer on single RNN performance. We notice that RNN performs almost the same but slightly better when the hidden layer dimension increases, while it becomes extremely slow and takes a lot of time to converge. Thus, we choose the dimension of the hidden layer to be 64. All RNN models in our baselines can achieve good performance and can converge in reasonable amount of time. Figure 4.6 shows the relationship between a single RNN hidden layer dimensionality versus the accuracy (in terms of global pixel-wise) trained on the NYU V2 to predict 4 semantic categories.

Examining our multimodal RNNs with other CNN features - VGG Features on NYU V2 - 14 We also examine our proposed multimodal RNNs while replacing the input CNN features by the extracted features from the VGG-16 pre-trained

Algorithm	Pixel Acc	Class Acc	IOU
CNN-RGB	50.84%	35.37%	21.72%
CNN-Depth	56.08%	36.82%	23.41%
CNN-RGBD	54.40%	35.48%	22.07%
RNN-RGB	56.22%	42.41%	29.19%
RNN-Depth	62.27%	45.99%	33.62%
RNN-RGBD	61.95%	48.15%	35.74%
RNN-Features-Combined	64.30%	51.54%	39.31%
RNN-Hiddens-Combined	64.50%	51.70%	37.30%
RNN-Classifiers-Combined	64.44%	50.27%	37.42%
Multimodal-RNNs-Ours	66.23%	53.06%	40.59%
Multimodal-RNNs-Ours-Multiscale	67.90%	54.67%	43.27%
Wang et al.[106]	-	42.20%	-
Coupric et al. [102]	52.40%	36.20%	-
Fan et al. [192]	70.90%	50.20%	-
Hermans et al. [199]	54.20%	48.00%	-
Khan et al. [195]	58.30%	45.10%	-

Table 4.3: Results on **NYU V2** dataset [2]. The performance of all baseline and other state-of-the-art methods is measured on total pixel-wise, average class-wise and average IOU (intersection over union) among 14 semantic classes. The higher the better.

Figure 4.6: The relationship between the hidden layer dimensions and the pixel-wise accuracy in a single RNN model trained on the NYU V2 dataset [2]

Figure 4.7: 7 example results from NYU-V1 (13 semantic categories). Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th: ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color)

Figure 4.8: 7 example results. Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th: ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color)

Figure 4.9: 7 example results from NYU-V2 (14 semantic categories). Columns 1st and 2nd: RGB color images and their corresponding Depth images. Column 3rd: the prediction results of our most competitive baseline ‘RNN-Features-Combined’ on the NYU V2 four semantic categories. Column 4th: our multimodal-RNNs prediction results. Column 5th: ground truth. Our model is able to correctly classify many miss-classified pixels compared with our baseline. (Best viewed in color)

model (extracted from Conv5-3 layer) [200]. We focused mainly on the most competitive task among our addressed tasks, i.e. classifying the 14 classes in NYU V2. The purpose of these experiments is to validate whether our fusion structure is a network-independent model, and concretely orthogonal to other CNN networks. In other words, replacing the CNN models with more powerful network like VGG[200], ResNet[201], FCNs[202], Dilated Networks[203] and others can boost the overall performance while the relative improvement of our proposed cross-connectivity fusion is maintained. Throughout all of our experiments, we observe that **replacing our CNN features with VGG features result in a constant overall increase in the accuracies in all of our RNN models (including the baselines) of around 5% higher in terms of IOU (most competitive metric).**

Notice that in this work, we didn't perform joint training between the CNN layers and our multimodal RNN layers. In contrast, we perform stage training; we first train the CNN for feature extraction and then train our RNN model for final local classification. This makes the performance comparisons on the NYU-V2 tasks between our model and other state-of-the-art models not fair (many works perform end-to-end joint training of various CNN or multi-scale CNN and CRF-based methods and even with RNN/LSTM). Notice also that end-to-end training with CNN can allow training of efficient deconvolution layers to upsample the output feature maps in order to restore their original resolution, while we use simple bilinear interpolation to upsample our final output map produced by the RNNs. Thus, in order to fairly examine the full performance of our multimodal RNN model combined with other types of recent CNN models like ResNets[201], FCNs[202] and Dilated Networks[203]; a joint end-to-end training is required. In this work, we didn't perform this joint training between the CNN and the RNN models as it is not our main contribution, but we consider it as a very good future work.

We also report the inference (test) run time on the test set of the NYU V2 for the task of classifying 4 categories (averaged over 20 trials for each input image of

size $\sim 640 \times 480$, on an Intel Core i7-5500U CPU at a speed of 2.40GHz, 2394 Mhz, 2 Cores, 4 Logical Processors, accompanied with 8 GB installed Physical Memory - RAM). Notice that we only report the inference run time of the RNN models excluding the CNN models since we mainly use the CNN models for features extraction, where the convolutional features serve as the input to the RNN models. Our proposed model: Multimodal-RNNs-Ours/Multimodal-RNNs-Ours-Multiscale ($\sim 570ms$) and the Baselines: RNN-RGB/RNN-Depth/RNN-RGBD ($\sim 350ms$), RNN-Features-Combined ($\sim 810ms$), RNN-Hiddens-Combined ($\sim 500ms$) and RNN-Classifiers-Combined ($\sim 420ms$). Notice that the hidden feature dimension is twice (double) of a single data modality input dimension in the case of RNN-Features-Combined; which is due to the early features concatenation.

4.3.5 Observations

From Table 4.1, 4.3 and 4.2 we have the following observations:

Modeling contextual dependencies between patches using RNNs helps:

The improvement gain achieved by our RNN models over the CNN models is significant. CNN features are locally learned when performing the convolutions and thus fail to encode long-range contextual information. RNNs are powerful on modeling short and long range dependencies between patches within the image and can learn context-aware features effectively.

Sharing information between RNNs helps: We design the baseline ‘RNN-Classifiers-Combined’ that combines two RNN models on the classifier level. Our multimodal RNNs model outperforms this baseline as it benefits from the shared contextual information extracted through the transfer layers.

Learning transfer layers to connect RNNs helps: The baselines ‘RNN-RGBD’ and ‘RNN-Features-Combined’ are designed to mix RGB and Depth data modalities together before learning the features. Our model outperforms these baselines because it

has an assigned single RNN for each modality to retain the modality-specific information. Plus, the transfer layers are learned to adaptively extracts only the relevant multimodal shared information.

4.4 Chapter Summary

This chapter presents a new method for RGB-D scene semantic segmentation. We introduce information transfer layers between two quad-directional 2D-RNNs. Transfer layers extract relevant contextual information across the modalities and help each modality to learn context-aware features that can capture shared information. Our model is able to retain both modality-specific and across-modality features. Both RNNs are trained simultaneously.

Chapter 5

Episodic CAMN: Contextual Attention-based Memory Networks With Iterative Feedback For Scene Labeling

In previous chapter, we improve RNN model and allow multiple RNNs to share visual knowledge within and between multiple data modalities. In this chapter, we study typical RGB scene labeling, where we address the problem as a contextual modeling problem and process only single data modality which is the RGB image itself. We focus on improving a deep CNN model and allow it propagate and share the visual knowledge contained within each input image. Scene labeling can be seen as a sequence-sequence prediction task (pixels-labels), and it is quite important to leverage relevant context to enhance the performance of pixel classification. In this chapter, we introduce an episodic attention-based memory network to achieve the goal. We present a unified framework that mainly consists of a CNN, specifically, Fully Convolutional Network (FCN) and an attention-based memory module with feedback connections to perform context selection and refinement. The full model produces context-aware representation for each target patch by aggregating the activated context and its original local representation produced by the convolution layers. We evaluate our model on PASCAL Context, SIFT Flow and PASCAL VOC 2011 datasets and achieve competitive results to other state-of-the-art

methods in scene labeling.

5.1 Introduction

This work deals with the problem of scene labeling (or semantic segmentation), which typically aims to relate a unique semantic class label to each pixel in an image. It is a challenging task in computer vision, as different scenes may be full of complex and occluded objects, and the images can be captured under various lighting conditions and viewpoints. Also, objects in an image may appear at any location and scale. In order to classify each pixel in a typical scene labeling pipeline, a feature vector is first extracted from a patch of adequate size containing that pixel. The patch contains the surrounding context to locally discriminate the pixel during labeling. But these local feature representations can be ambiguous, e.g. a sofa patch can be visually indistinguishable from a bed patch. As a natural solution, many papers utilize context to distinguish locally ambiguous patches [46–48]. However in their works, the long-range context is not effectively leveraged. Besides, the surrounding contextual information is incorporated without selection. We believe that different contextual information is not always equally important or useful to one local region. For example, when recognizing a local region belonging to computer monitor, it is more helpful to leverage contextual information from desk rather than from other regions such as window. Hence, special attention should be paid to certain context conditioned on the target region.

In this work we propose a unified framework that includes FCN and episodic attention-based memory network layers to address the problem. In details, it consists of three major components: (1) a basic FCN to generate local convolutional patch-level feature representations; (2) a Contextual Attention-based Network (CAN) that adaptively selects the relevant contextual patches for each referenced patch; (3) an ‘episodic’ recurrent memory module that accumulates and records the selected context over multiple episodes, where its recurrent feedback connections allow the CAN module to refine the activated/selected

Figure 5.1: An illustrative overview of our unified framework. The model consists of FCN convolution layers, our episodic contextual attention-based memory network, and the upsampling alongside the FC (classification) layers. Given an input image, our model generates a pixelwise label map.

context over multiple iterations. CAN alongside the recurrent memory network (CAMN) appears to be an effective contextual modeling method that can easily activate useful long-range context. However, CAMN may falsely activate irrelevant context and wrongly inhibit the relevant context in one shot; therefore we propose to refine the selected context in an iterative framework with the recurrent feedback mechanism. We instantiate the episodic memory as the hidden representation of a recurrent neural network [204], which is able to efficiently memorize local information over a short time (several iterations). Then, we add the feedback connections from the episodic memory to CAN, so that the CAN module can refine the context selection in the next iteration. The episodic memory converges over only few early iterations. We describe our method as ‘episodic’ because it accumulates and refines the selected context over a series of consecutive iterations. Either all of the generated context of these episodes or only the last episode can be utilized.

An overview of the framework is shown in Figure 5.1. The full network is end-to-end trainable. Even without post-processing, we achieve state-of-the-art labeling results on several challenging scene labeling benchmarks. The main contributions of this chapter are summarized as follows:

- We propose the contextual attention-based memory network model that is able to

adaptively select relevant context for each target patch. Importantly, it is a novel contextual modeling method that can effectively engage long-range context.

- We introduce an episodic memory module to accumulate contextual selections and aid the CAMN to refine the previous context selections through recurrent feedback connections over multiple iterations (*‘episodes’*).
- We achieve very competitive results on three public scene labeling benchmarks.

5.2 Related Work

To the best of our knowledge, ours is the first work to approach scene labeling utilizing the differentiable soft attention mechanism alongside recurrent memory networks. Our model can dynamically and interactively select and refine the relevant and informative contextual patches for each referenced patch, therefore its local representation is contextualized. There are three main research directions in the literature that are related to our work in this chapter, see Section 2.8 which reviews contextual modeling works that address scene labeling problem, and Section 2.3 which review recent attention-based models. The third direction is the recurrent feedback mechanize.

5.2.1 Recurrent Feedback

Recurrent feedback has been explored in the tasks of scene labeling and human pose estimation. Auto-context [205] is the pioneering work that utilizes the output of classifiers as feedback to the next classification model. Meanwhile, Pinheiro et al [115] add feedback connections between the output and the input of the convolutional neural networks (CNNs). Namely, the output of the CNN in the previous iteration is fed back to the input of the same CNN in next iteration. This recurrent CNN has been successfully applied to scene labeling. Besides, Carreira et al [206] also add the feedback connection in the CNNs to enable the network to learn from past errors, so that it can predict the

location of human joints more accurately in the next iteration. The idea of our episodic memory feedback is similar to these works. Engaging the recurrent feedback connections can be viewed as cascading multiple attention layers that can form a deep attention model. However, instead of simply adding the feedback from the output of the previous iteration, we introduce an episodic memory to record and accumulate all past information leveraging RNN layers, and then add its feedback to the attention-based module such that the selected context is refined over multiple iterations.

5.3 Framework

Given a scene image $\mathbf{S}$, the task is to map from the pixel-space to label-space. We divide each image into patches and train our episodic CAMN model to extract contextually-aware convolutional features to better represent each local patch. Suppose patches in a scene image $\mathbf{S}$ are represented in terms of convolutional features as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$, where N is the number of patches, our goal is to predict the semantic label of the referenced patch $\mathbf{x}_i$. Specifically, given a referenced raw local patch from $\mathbf{S}$, the data flow in our model is as follows (cf. Figure 5.1):

Input feature map: the convolutional layers in the FCN convert input patches into local feature representations $\mathbf{x}_i$.

Output feature map: the episodic CAMN performs contextual modeling and compute the output which is the contextually-aware features for the input referenced patch. It adaptively and progressively selects the relevant contextual patches for the referenced patch. Two major networks are employed within: (1) the CAN model contains a feed-forward network to calculate the compatibility/similarity scores to perform soft attention between the input $\mathbf{x}_i$ and the remaining patches in the image $\mathbf{x}$; (2) the feedback mechanism is comprised of a recurrent layer that records and aggregates/accumulates the previous generated context (activated context) and links it back to the feed-forward layers so that the CAN can further refine the previous selections over multiple iterations.

This also allows the model to generate deep contextual representations. The recurrent feedback connections with the CAN module allow it to adapt (refine) the activated context over multiple iterations. The episodic CAMN module is run for multiple iterations per training epoch until the selected context converges.

Response: To calculate the response (label) of the input patch, the last classification layers (FCs) alongside the upsampling layers and Softmax are engaged to decode the final contextually-aware dense representations.

5.3.1 FCN for Local Patch Representation

Fully convolutional networks (FCN) originally adapt classification networks like the VGG net [26] into fully convolutional networks and transfer their learned representations to the segmentation task. FCN architecture combines/fuses semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. In their work, they design the basic 32s model that has no intermediate skip connections from earlier layers, and the 16s version has skip connections from *pool4* layer and finally the 8s version that has skip connections from both *pool4* and *pool3* layers. All FCN models have upsampling layers which upsample the downsampled label prediction maps (due to the application of the the pooling layers) to the original image resolution. The proposed episodic CAMN model can be inserted into any location between the last pooling layer and the final FC layer. In our work, we placed it between FC6 and FC7. The input patch representations to episodic CAMN are generated by the early convolution layers and the output of episodic CAMN is the contextually-enriched representations.

5.3.2 Contextual Attention-based Memory Model

Context is of great significance in local prediction. In this chapter, we model context from the perspective of feature representation. Our attention-based module is introduced to learn to activate certain contextual features for each local patch. Concretely,

Figure 5.2: An illustration of our Episodic CAMN including its contextual attention-based network, the episodic recurrent memory and the feedback. The model consists of two major components: (1) the CAN adaptively selects the relevant context to contextualize the reference patch representation ($\mathbf{x}_i$); (2) the episodic memory summarizes the activated context in the past. The feedback from the episodic memory enables the CAN to iteratively refine the selected context over multiple iterations.

it aggregates selected features from surrounding patches with the referenced patch to produce its contextualized representation, thus encodes useful contextual information for local classification. However, for a referenced patch, not all the surrounding patches are equally useful. For example, in terms of the contextual support to a pillow patch in a bedroom image, bed patches are more useful than the wall patches. Hence, the attention mechanism is developed to adaptively select the relevant patches and assign them proper weights. The episodic CAMN model has feedback connections to operate over multiple iterations, thus refining the selected context. Our model can naturally engage long-range contextual dependencies, which makes it very suitable for scene labeling.

5.3.2.1 Contextual Attention-Based Network

In order to attend to patches which produce useful contextual information to $\mathbf{x}_i$, our CAN uses a feed-forward network (Attention model) to evaluate the high-level relevance

between referenced patch representation $\mathbf{x}_i$ and the other surrounding patches $\mathbf{x}_j$. Mathematically, it is expressed:

$$z_{ij} = w_b^T \tanh(W_a \mathbf{x}_i + V_a \mathbf{x}_j + b), \quad (5.1)$$

where W_a, V_a are embedding matrices and w_b is a vector to capture feature similarities; they are jointly learned. z_{ij} reflects the degree of relevance or compatibility (activation/inhibition) between patch representations $\mathbf{x}_i$ and $\mathbf{x}_j$.

In attention-based models, different techniques and alternative methods are used to calculate the patch-level interactions and to capture similarities (generate the relevance scores) between the referenced patch and the rest of the patches in the image. Precisely, content-based functions are applied on the patch feature representations which are usually followed by a normalizer such as Softmax to generate pairwise relatedness scores (normalized probabilities); also known as the alignment functions. For example, the concatenation: $W_a[\mathbf{x}_i; \mathbf{x}_j]$, or cosine similarity: $[\mathbf{x}_i^T \mathbf{x}_j]$, or absolute distance: $|\mathbf{x}_i - \mathbf{x}_j|$. In case of cosine similarity; $[\mathbf{x}_i^T \mathbf{x}_j]$, this multiplicative/angle measure (named also as the *dot* function) performs the dot product between a pair of feature vectors to capture elementwise comparisons of the signs of the their representations. Meanwhile the *absolute-distance* function, $|\mathbf{x}_i - \mathbf{x}_j|$ measures the distance between a pair of feature vector absolute values. In case of the *concatenation* function, it concatenates a pair of features into one feature vector and project it into a fully connected weight matrix that can capture possible associations within this concatenated vector directly. Additionally, there is the *general* function, $x_i^T W_a x_j$, where the relatedness between a pair of feature vectors is calculated through multiplying one of the feature vectors by the other within-feature similarities that are accessed directly through a feedforward layer.

As shown in our model design, we decide to perform the *feedforward* function, $w_b^T \tanh(W_a \mathbf{x}_i + V_a \mathbf{x}_j + b)$, which first calculates the similarities within each feature vector through W_a and V_a embedding matrices in a feedforward layer and then capturing higher associations between these pair by introducing a multiplicative weight vector which accesses

these within-feature similarities directly w_b . We observed that all these functions perform almost the same, while the feed-forward layers perform the best.

CAMN attends and generates the context vector $\mathbf{c}_i$ by soft attention based on $\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{x}_j$, where $\alpha_{ij} = \text{softmax}(z_{ij})$, s.t. $\sum_j \alpha_{ij} = 1$. α_{ij} is a non-negative scalar that modulates how much information from the patch representation $\mathbf{x}_j$ is exposed to $\mathbf{c}_i$. It is obtained by applying the typical softmax function to generate the relevance distribution over all locations. Then, the weighted summarization of the activated patch representations gives rise to the context vector $\mathbf{c}_i$, which is calculated through the convex summation of $\mathbf{x}$ weighted by α_{ij} . Interestingly, as the context vector is able to include features from distant image patches, it can capture the long-range contextual information easily.

We also experiment with the ReLU function instead of Softmax. The ReLU operates directly on the unnormalized probabilities produced by the feed-forward layers. We observe that in some cases the ReLU performs as well as Softmax. However, Softmax produces a more visualization-friendly distribution of weights over the patches.

The final ‘context-aware’ representations of the referenced image patches are generated by aggregating the original features $\mathbf{x}_i$ with their specific context vectors $\mathbf{c}_i$:

$$\mathbf{h}_i = \text{ReLU}(U_{hx}\mathbf{x}_i + U_{hc}\mathbf{c}_i + z_h), \quad (5.2)$$

where U_{hx} and U_{hc} are the transformation matrices that map the input and the context vector respectively to new hidden representation space, z_h is a bias vector, and $\mathbf{h}_i$ is the hidden representation forwarded to the classification layers. All of the parameters in Equation 5.1 and 5.2 are jointly learned. In this scenario, the CAN is trained to maximize the labeling performance on the training images by inserting useful context into the local representations.

To generate the response/class likelihood for each referenced patch, we apply $\mathbf{r}_i = \text{softmax}(V_r\mathbf{h}_i + z_r)$, where V_r is the classification matrix, z_r is a bias vector, and $\mathbf{r}_i$ is the class likelihood for $\mathbf{x}_i$.

5.3.2.2 Episodic Recurrent Memory and Feedback

The CAN produces the context-aware vector that summarizes the relevant contextual information, which is selected based on the attention model. However, if we only forward the attention model once to calculate the attention weights, some irrelevant patches may be mis-activated while some relevant patches may be falsely inhibited. To address this issue, we propose to refine the attention weights iteratively such that the CAN model can attend to undiscovered relevant context and remove irrelevant context. Specifically, we introduce an episodic memory module that is able to memorize the context selections in the past. We employ the feedback from the episodic memory to refine the selected context by CAN. Biologically, it is inspired by discoveries in neuroscience that neuron adaptation actually happens across multiple time-scales through the feedback connections between the dendrites and the axon [207, 208]; feedback of ionic currents are flowing back from the dendrites to the axon to allow refining its adoption of the stimulus [208].

Suppose we generate the corresponding relevance score vector $z_i^t = [z_{i1}^t, z_{i2}^t, \dots, z_{iN}^t]$ of a reference patch representation $\mathbf{x}_i^t$, and a context vector $\mathbf{c}_i^t$ during the t -th iteration. The episodic memory is intended to accumulate and record the relevant context selected by the CAN over multiple iterations. In this chapter, we instantiate the episodic memory as a recurrent neural network [204] (RNN), considering that RNNs are well-known for their memory capability for short sequences. The activities in the episodic memory module can be formulated as:

$$\mathbf{h}_i^t = \text{ReLU}(U_{hx}\mathbf{x}_i + U_{hc}\mathbf{c}_i^t + U_{hh}\mathbf{h}_i^{t-1} + z_h), \quad (5.3)$$

where $\mathbf{c}_i^t$ is the context vector generated in the t -th iteration, $\mathbf{h}_i^t$ represents the memory state in the t -th iteration. $\mathbf{h}_i^t = \mathbf{x}$ as $t = 0$, U_{hh} , U_{hx} and U_{hc} are hidden-hidden, input-hidden and context-hidden transformation matrices, respectively. In each iteration, the episodic memory module accumulates the context vector produced from the CAN module $\mathbf{c}_i^t$ into the memory hidden representations.

With the availability of the episodic memory, we add the feedback connections between the memory module and the attention module. In this case, the episodic memory feedback allows the CAN to refine the selected context in the subsequent iteration. Concretely, the equation that calculates the relevance score is adapted to the following form:

$$z_{ij}^t = w_b^T \tanh(W_a \mathbf{x}_i + V_a \mathbf{x}_j + U_a \mathbf{h}_i^{t-1} + b), \quad (5.4)$$

where z_{ij}^t is the relevance score between patch representations $\mathbf{x}_i$, $\mathbf{x}_j$ and $\mathbf{h}_i^{t-1}$ in the t -th iteration. The memory state ($\mathbf{h}_i^{t-1}$) encodes the context selections over previous iterations. Thus, when they are fed back, the context selection is expected to be refined by allowing $\mathbf{x}_i$ and $\mathbf{x}_j$ to interact with all previous context summarization $\mathbf{h}_i^{t-1}$.

Feeding back the memory vector that contains a summarization of the previous contextual selections allows context-context interactions, therefore new indirect relevant contextual patches can be activated. However, we observe that a few iterations are enough to discover new related context as the selection saturates quickly. We call these forward passes *episodes*. Specifically, after T forward passes by the CAN, context-aware representations generated by the final episode or all the episode states are forwarded to the classification layers. We call our model Episodic CAMN (see Figure 5.2).

5.3.3 Model Optimization

Given an image $\mathbf{S}$, we derive the class likelihood for each constituent patch in $\mathbf{S}$. Next, we calculate the cross entropy loss to train the full model (including the parameters in episodic CAMN). The error signal of an image is averaged across all valid (i.e. semantically labeled) constituent patches $\mathbf{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$:

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{b=1}^B \delta(\mathbf{y}_i = b) \log \mathbf{r}_i(b), \quad (5.5)$$

where $\delta(\cdot)$ is the indicator function, B is the number of semantic classes, y_i is the ground truth label for the patch representation $\mathbf{x}_i$. $\mathbf{r}_i$ is the class likelihood for the patch representation $\mathbf{x}_i$, which is a B -dimensional vector. We ignore the contribution of unlabeled

(invalid) patches in the loss calculation. The whole model is differentiable, thus it is end-to-end trainable by using the back-propagation algorithm. The weights in recurrent layer (episodic memory) are shared, and they are updated via back-propagation through time (iteration) after unfolding the episodic memory network in time (iteration).

5.4 Experiments

In this section we describe our experimental evaluation and we provide an ablation study of our proposed architecture. We apply our framework to the task of scene labeling, and we show competitive performance to other state-of-the-art works on PASCAL-Context, SIFT Flow and PASCAL VOC 2011. We also perform a diagnostic evaluation of our convolutional episodic CAMN and other hyperparameters/design choices.

5.4.1 Datasets

PASCAL-Context [6] comprises 4998 training images and 5105 testing images. Originally, the images are sampled from PASCAL VOC 2010 dataset and re-labeled at pixel-level for the segmentation task, where there are in total 540 classes. Each image has a resolution of about 375×500 pixels. In our experiments, we only consider the task of labeling the 59 most frequent classes for evaluation.

SIFT Flow [9] consists of 2688 images. We follow the training/testing split protocol (2488/200) provided by [9] in our experiments. The images are captured from 8 typical outdoor scenes with a resolution of 256×256 pixels. The segmentation task in this dataset is to assign each pixel to one of the 33 semantic classes. Statistically, this dataset has an imbalanced class distribution. We found that applying class balancing helps similar to [142, 146].

PASCAL VOC 2011 [8] involves 21 categories, including 20 foreground object classes and one background class. There are 736 images as non-intersecting validation set and 1111 testing images provided by their server. In our experiments, we only train

Algorithm	PASCAL-Context			SIFT Flow			PASCAL VOC 2011		
	PA(%)	CA(%)	IU(%)	PA(%)	CA(%)	IU(%)	PA(%)	CA(%)	IU(%)
Baseline-FCN	65.90	46.53	35.11	85.19	54.68	41.45	90.30	75.90	62.70
FCN+CRF	69.19	47.36	38.37	85.70	52.44	43.12	90.98	76.30	64.01
FCN+CAMN	71.27	52.91	39.64	85.90	57.10	43.88	91.41	76.99	66.91
FCN+Episodic-CAMN	72.11	54.28	41.18	86.20	58.69	45.22	92.26	78.59	68.18

Table 5.1: Results on the PASCAL-Context [6], SIFT Flow [9] and PASCAL VOC 2011 (validation set) [8]. All models are trained jointly with FCN models (best performing ones).

our final model from the training set and did not include any images from the validation set.

Following recent literature, we report three performance evaluation scores: Pixel Accuracy (**PA**) (percentage of all correctly classified pixels), Per-class Accuracy (**CA**) and the Intersection-Over-Union (**IU**).

5.4.2 Implementation Details

We adopt the stochastic gradient descent (SGD) with momentum. The learning rate starts at 10^{-5} and decays exponentially with the rate of 10% after 10 epochs. The momentum is fixed as 0.9. The internal dimension of the RNN network (episodic memory h_i^t) is set to the same as that of the input feature vector (dimension = 4096). The reported results are based on the model trained for 50 epochs. The feedforward, FC layers and RNN model parameters are initialized either randomly or with zeros. For fair comparison, we select the strongest FCN models (best performing among 8s, 16s and 32s versions) as our baselines in all datasets. We follow [135] to train the fully convolution network layers. We initialize all our convolutional models with the ImageNet-pretrained model [209]. We didn’t engage any additional training data (such as Microsoft Common Objects in Context ‘COCO’ dataset [210]). We used the publicly available MatConvNet MATLAB implementations [211].

We jointly train FCN and CAMN as one unified model. We introduce the following baselines to prove the effectiveness of our proposal.

Baseline-FCN performs full training of the FCN model [135] for scene labeling. In this baseline, we examine and report the best model among 8s, 16s and 32s models for each dataset.

FCN+CRF performs post-processing on the strongest trained FCN model using the Fully-connected Conditional Random Fields (CRF) implementations [212]. We choose to engage fully-connected CRF because it is one of the strongest contextual modeling methods in scene labeling.

FCN+CAMN performs joint training of our model; FCN with one-episode CAMN. Here, only one iteration of CAMN is engaged.

FCN+Episodic-CAMN is our final overall proposed framework with episodic memory ($T = 3$).

5.4.3 Evaluation Results

Comparison with FCN: The quantitative labeling results on the PASCAL-Context, SIFT Flow and PASCAL VOC 2011 datasets are summarized in Table 5.1. Our FCN+CAMN outperforms FCN on the three datasets by 4.53%, 2.43% and 4.21%, respectively (IU). The results show that the discovered context is indeed informative towards understanding the semantic classes of the referenced patches and strengthen the discriminative power of the local representations. For Baseline-FCN, we perform class weighting while evaluating on SIFT Flow dataset. The original results in [135] are lower than ours on this dataset.

Recurrent Feedback: The introduction of the feedback from the episodic memory module is to steer the CAMN to iteratively refine the selected contextual memories and produce more powerful deep contextual vectors. The performance of our overall model FCN+Episodic-CAMN is better than that of FCN+CAMN. In Table 5.1, the IU accuracy is increased around 1.54%, 1.34% and 1.27% over one-episode CAMN in all the datasets respectively. The most significant performance jump originates from the CAN alone, which indicates that the attention model is able to select very useful contextual patches

just in one-shot. Moreover, as we introduce the episodic memory feedback to iteratively refine the context selections, the recognition performance for some classes can be further improved. As shown in Figure 5.6, the context vector usually saturates after around 2-3 iterations; the performance starts to degrade slightly after many iterations ($T > 5$). This may be due to the effect of over-contextualizing the local features as after several contextual aggregations, some local discriminative features may be overwhelmed by the global contextual features.

Figure 5.3 show the qualitative labeling results and their gradual improvement over iterations. The quality of label prediction is significantly improved over FCN as CAMN is engaged for the first time, and it is further improved as the context refinement evolves. We also visualize in Figure 5.5 the weighted selections (Heatmaps) for several patches belonging to building, sky and computer respectively at episode 1 and 3. In the first row, the heat increases in some areas (e.g. building, road) and decreases in other areas (e.g. sky). The second row visualizes heatmaps for a sky patch belonging to the same scene as in the first row. This shows that attention model is indeed location/patch sensitive. The third row presents heatmaps for computer monitor patch, showing how some patches from table are activated as related context. Our method models the associations between features at patch level, i.e. not necessarily the whole object is activated as related context. In other words, different parts of an object/stuff can be activated for different target object/stuff parts.

We also report (train /test) run time on the validation set of PASCAL VOC 2011 (averaged over 20 trials for an input image of 384×384 on an NVIDIA Tesla K40m with a companion CPU Intel Xeon E5-2643 v3 at 3.40GHz); FCN+Episodic-CAMN: ($\sim 775ms$ / $\sim 270ms$) and Baseline-FCN: ($\sim 546ms$ / $\sim 157ms$).

Comparison with State-of-the-art: We compare the performance of our full model with other state-of-the-art methods for contextual modeling. The quantitative result comparison on PASCAL-Context, SIFT Flow and PASCAL VOC 2011 datasets are listed

Figure 5.3: Examples of qualitative labeling results (best viewed in color). Each row shows the original test image, its ground truth label map, and three label prediction maps of our convolutional episodic attention-based contextual memory network with different iterations. The test images are from the PASCAL-Context [6]. The quality of the label prediction maps is remarkably improved as CAN is engaged (iteration 1), and is gradually refined after additional iterations.

Figure 5.4: Comparison with CRF-RNN method [7]. The samples are from PASCAL VOC 2011 dataset [8].

in Tables 5.2, 5.3 and 5.4 respectively. There are other methods that use different settings like adopting Residual Networks [56] (e.g. [138, 213]) or engaging extra training data to train their models. In our case, we only compare with VGG-based networks with similar settings. Table 5.4 shows evaluation results on PASCAL VOC 2011 on the test set (provided by their server). Our model achieve significant improvement over FCN-8s.

Comparison with Fully Connected CRF and other CRF-based Models: We run the fully-connected CRF model [212] that is used in other state-of-the-art works [137, 138] as a post-processing step based on FCN prediction maps. As shown in Table 5.1, our method outperforms it consistently. This performance gap illustrates that the proposed method is more effective than CRF on leveraging relevant context to perform local classification. Plus, our model is very simple in terms of optimization compared with graphical-based models. It is purely constructed from primary feed-forward and recurrent layers so it is fast and easy to optimize.

As reported in Tables 5.2 and 5.3, our method outperforms other state-of-the-arts that use CRF models such as [149] and [7] in different evaluation metrics. But in Table

Algorithm	PA(%)	CA(%)	IU(%)
FCN-8s [135]	65.9	46.5	35.1
FCN-8s [214]	67.5	52.3	39.1
DeepLab [138]	-	-	37.6
DeepLab-CRF [138]	-	-	39.6
HO-CRF [215]	-	-	41.3
CNN-CRF [149]	71.5	53.9	43.3
CRF-RNN [7]	-	-	39.3
ParseNet [216]	67.5	52.3	39.1
ConvPP-8 [217]	-	-	41.0
PixelNet [218]	-	51.5	41.4
O2P [219]	-	-	18.1
CFM [220]	-	-	34.4
BoxSup [221]	-	-	40.5
FCN+Episodic-CAMN	72.1	54.3	41.2

Table 5.2: Performance comparison on PASCAL-Context[6].

5.4, CRF-RNN [7] outperforms ours on PASCAL VOC 2011 by 2.0% while our method outperforms CRF-RNN on PASCAL Context as shown in Table 5.2 by 1.9%. CRF-based methods are effective for producing more refined object boundaries which contributes a lot to boost the IU score, especially for the object-centric PASCAL VOC (foreground objects are annotated only). Meanwhile, our method can explicitly model the contextual dependencies between image patches from different categories, hence it can better capture the contextual interactions when images are densely annotated with many different categories (PASCAL Context and SIFT Flow). Our method and CRFs can be used in different applications. In Figure 5.4, we show some qualitative examples from PASCAL VOC 2011 dataset where CRF-RNN model performs better (third column).

In terms of speed, the inference time of our Episodic-CAMN alone (as a single block) separately is $\sim 0.077s$ on the previously mentioned CPU (no GPU is engaged). Meanwhile the inference time of the dense CRF alone separately is $\sim 3.010s$ on the same CPU. Aside from simplicity, our model is much faster and more efficient ($\sim 39\times$ speedup) compared with CRF-based method, yet it performs competitively (although with high feature dimension 4096), which makes our method suitable for applications that require

Figure 5.5: Some sample images from SIFT Flow [9] and PASCAL VOC 2011 [8] alongside the visualization of the weighted selections for specific local patches. From left to right: RGB image, label map, episode 1 heatmap and the final episode 3 heatmap.

high speed processing.

Comparison with RNN-like Models: As shown in Table 5.3, our model outperforms many state-of-the-art methods that use RNN on top of convolutional features, e.g. [142] and [141]. Additionally, while it is difficult to visualize the hidden states of an RNN to clearly see whether the network can capture the underlying associations and dependencies among the input sequence, attention-based memory networks can easily reveal this secret. It can simply and explicitly measure the relevance among the patches which eases the visualization and thus help us understand the underlying hidden relationships and interactions within a scene image, as shown in Figure 5.5.

5.5 Chapter Summary

In this chapter, we deal with the problem of scene labeling. In order to effectively leverage the relevant contextual patches to enhance the local classification accuracy, we propose

Figure 5.6: IU performance changes in terms of iterations.

Algorithm	PA(%)	CA(%)	IU(%)
Liu et al [9]	74.8	-	-
Liu et al [222]	76.7	-	-
Tighe et al [223]	75.6	41.1	-
Farabet et al [136]	72.3	50.8	-
Farabet et al [136]	78.5	29.6	-
FCN-16s [135]	85.2	51.7	39.5
FCN-8s [214]	85.9	53.9	41.2
CNN-LSTM [141]	70.1	22.6	-
CNN-CRF [149]	88.1	53.4	44.9
DAG-RNN [142]	81.2	45.5	-
Pinheiro et al [115]	77.7	29.8	-
RCNN [224]	83.5	35.8	-
RCNN [224]	79.3	57.1	-
ClassRare [146]	79.8	48.7	-
Sharma et al [48]	79.6	33.6	-
Sharma et al [48]	75.5	48.0	-
ParseNet [216]	86.8	52.0	40.4
JointCalib [225]	-	55.6	-
Tighe-MRF [223]	78.6	39.2	-
FCN+Episodic-CAMN	86.2	58.7	45.2

Table 5.3: Performance comparison on SIFT Flow [9].

Figure 5.7: Supplemental Material. Examples of qualitative labeling results (best viewed in color). The images are from PASCAL VOC 2011 dataset.

Figure 5.8: Supplemental Material. Examples of qualitative labeling results (best viewed in color). The images are from SIFT Flow dataset.

Algorithm	IU(%)
BerkeleyRC [226]	39.1
SDS [227]	52.6
R-CNN [228]	47.9
FCN-8s [135]	62.7
FCN-8s [214]	67.5
Zoomout [229]	64.4
CRF-RNN [7]	72.4
FCN+Episodic-CAMN	70.4

Table 5.4: Average performance comparison on the test set from PASCAL VOC 2011 [8].

an episodic attention-based contextual memory network. This model presents a unified framework that mainly consists of FCN convolutional layers, attention-based model, and an RNN to perform context selection and refinement. The full model produces context-aware representation for each target patch by aggregating the activated context and its original local representation.

Our experiments show that the proposed method significantly boosts the labeling performance of the original FCN by incorporating the selected context. More importantly, our method achieves competitive performance on the public PASCAL-Context, SIFT Flow and PASCAL VOC 2011 scene labeling benchmarks when compared with other state-of-the-art work.

Chapter 6

Beyond Forward Shortcuts: Fully Convolutional Master-Slave Networks (MSNets) with Backward Skip Connections for Semantic Segmentation

In previous chapter, we propose our deep framework that consists from a deep FCN, attention-based model and an RNN to perform contextual modeling for each local patch in the input image. In this chapter, we zoom in our focus on improving FCN model and allow it to propagate related contextual high-level semantic information on its own. Recently, most of deep CNNs contain forward shortcut connections; i.e. skip connections from low to high layers. Reusing features from lower layers that have higher resolution (location information) benefit higher layers to recover lost details and mitigate information degradation. However, during inference the lower layers do not know about high layer features, although they contain contextual high semantics that benefit low layers to adaptively extract informative features for later layers. In this chapter, we study the influence of backward skip connections which are in the opposite direction to forward shortcuts, i.e. paths from high layers to low layers. To achieve this – which indeed runs counter to the nature of feed-forward networks – we propose a new fully convolutional model that consists of a pair of networks. A ‘Slave’ network is dedicated to provide the

backward connections from its top layers to the ‘Master’ network’s bottom layers. The Master network is used to produce the final label predictions. In our experiments we validate the proposed FCN model on ADE20K (ImageNet scene parsing), PASCAL-Context, and PASCAL VOC 2011 datasets.

6.1 Introduction

Deep Convolutional Neural Networks (CNNs) achieve impressive performance on many computer vision tasks, including image classification [230–232], object detection [233] and semantic segmentation [234–236]. Deep CNNs extract hierarchical features by applying successive pooling and convolution layers with classifiers in an end-to-end trainable network [237].

Fully Convolutional Networks (FCNs) [234] are a natural extension of deep CNNs (e.g. [231] is adapted for image segmentation). Unlike image classification, where only a single prediction is expected per image, the task is to generate a prediction label map for the whole input image. To maintain the original image resolution, FCNs employ upsampling layers to undo the effect of the downsampling produced by the pooling layers. Interestingly, FCNs contain skip connections from early pooling layers (*pool3* and *pool4*) to upsampling layers. Such forward shortcut connections have two benefits: one is to avoid information degradation during the training process as evidenced by the residual block identity mapping in [232]; the other is to integrate bottom layer features, which usually contain detailed low information for label prediction in the inference process.

In FCNs (or general deep CNNs), when extracting bottom layer features, the network is not aware of information at higher layers. This is actually a drawback: if bottom layers understand the context earlier, they can adaptively extract more informative features. For example, when labeling a car image region, once the bottom layers know the region contains a car, they may focus on the car-like visual patterns. Based on this motivation, we propose backward skip connections which transfer information from higher layers to

Figure 6.1: (Best viewed in color). Refer to Section 6.4.2 for baseline definitions. Our MSNet model (last column) can correctly label object regions as the backward skip connections allow the lower layers to learn more informative features by receiving information from higher layers (contextually-aware). The sample images are from PASCAL VOC 2011 [10] validation set.

bottom layers directly. Figure 6.1 shows some interesting examples where our MSNet model (MSNet-FB2) is able to correctly label the object regions meanwhile other baselines fail. We can notice in these sample images that the overall high-level context plays an important role in discriminating the central object. Backward skip connections that are provided in our model help to pass such high-layer information as early as possible to low layers to extract informative features.

Specifically, in our work we observe the following: *higher layers features are substantial to be conditioned on in the lower layers*. Directly skipping and fusing features from the high layers back to lower layers shows consistently better results throughout our experiments.

Given the fact that layers are stacked in a feed-forward manner in CNNs, connecting layers backward seems impossible as it would create cyclic loops inside the network. We overcome this problem by designing a pair of networks, each of which is originally an FCN-8s model. Only one of them – the ‘Master’ – is used to produce label predictions.

Figure 6.2: ‘Forward skip’ is an illustration of one forward skip connection in our own baseline FCN-8s-F1. ‘Backward Skip’ is our proposed design where we align two networks, a Master and Slave. The Slave network provides the backward skip connections from its higher layers to the Master network lower layers. ‘Block in MSNet’ shows an illustration of our final model’s skip connections, where additional forward skip connections have been added to the Master.

The other network – the ‘Slave’ – is only responsible for providing the backward skip connections from its higher layers to the Master’s lower layers. The whole Master-Slave Network (MSNet) model is trained end-to-end. We evaluate our models on competitive benchmarks: ADE20K [11], PASCAL-Context [6] and PASCAL VOC 2011 [10], and we observe that our novel networks significantly outperform baselines which don’t engage backward skip connections.

6.2 Related Work

There are two research directions in the literature to tackle image segmentation, the first is by developing different CNN architectures which mainly evolve the idea of skip connections into different forms and designs, and the second is incorporating contextual modeling methods to collect different (multi-scale) contextual information that benefit local pixel classification. For the later one, see also Section 2.8 for more details.

6.2.1 Evolution of Skip Connections in CNNs

Highway Networks [238] propose to shortcut the flow through gating units to ease the training of a very deep CNN. ResNets [232] show competitive results by proposing to shortcut the flow through similar identity mapping connections. The basic idea is to skip from one layer (leaving some layers in between) and aggregate/sum with other later layer (usually by simple additive function). The skip connection can be gated/learnable or can be engaged directly (identity connection) and accompanied with prior convolutions, pooling or upsampling if needed. When FCNs are developed for segmentation [234], they propose skip connections that are designed to aggregate multi-scale feature maps of different channel dimensions. Similarly, simple convolutions are employed while skipping to further rematch feature dimensions prior fusing the feature maps. Beside, upsampling /bilinear interpolation is also employed to rematch feature spacial dimensions in case if the feature maps are being fused in different resolutions (transposed convolution can be also used instead). ResNets and Highway networks are considered to have shorter skips compared to FCNs which contain much longer skips (many layers are skipped in between) [239]. Follow up works are developed to provide different styles of skips (both long and short) such as Stochastic Depth [240] and Swapout [241]. Meanwhile DenseNets are also proposed to have extremely dense forward skip connections (such as each layer is densely connected to all proceeding layers) [242, 243]. Another orthogonal line of works propose skips but in wider version of CNNs [244, 245]. Different from all these forward skip connections, our model is designed with backward skip connections.

6.2.2 Contextual modeling and CNNs

Engaging larger contextual information from other regions are usually important for local pixel classification. CNNs implicitly employ contextual information through its natural integration of multi-scale (hierarchical) features across the layers. However, excessively

enlarging the receptive fields in the convolutions layers to capture longer/global context is insufficient [134]. Skip connections in FCNs can be considered as one solution to overcome this limitation by aggregating the intermediate features for classification [234]. Other works propose to explicitly encode global contextual information by stacking Recurrent Neural Network layers (RNNs) within CNNs. RNNs and their variants like Long-short Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs) are effective in encoding the dependency between different image regions and thus encode the global contextual information into local representations [134, 142]. Another line of works consider mainly the convolution operations and applied ‘atros’ and dilated convolutions to perform contextual modeling [138, 139]. Conditional Random Fields (CRFs) and Markov Random Fields (MRFs) are also used to model the contextual label dependency [7, 138, 149]. Most of the proposed methods stack CRFs on top of the classification layers in CNNs to serve as a post processing stage and can further capture the co-occurrence relationships between labels. However, our model is intended to improve the original FCNs structure and thus complementary to all these proposed models in the literature (e.g. our model can be easily extended and trained with RNNs or stacked with CRFs)

6.3 Fully Convolutional MSNets

While existing artificial CNNs have only forward connections, the biological brain – for example in the visual system – has complex connections of ample recurrent feedback between neurons [246, 247]. This inspired us to evolve FCNs to have variable length of backward skip connections to mimic those feedback connections in the brain.

The forward skip connections in basic FCN model encourage the reuse of feature maps from bottom layers to top layers and help to recover better resolution. In our model, we upgrade the FCN to contain backward skip connections from top layers to bottom layers. These connections enable the lower layers to be aware of higher-level information and context, as a result, the lower layers can extract features which are more informative

Figure 6.3: (Best viewed in colors). A detailed overview of our MSNet-FB1 model. The model consists of two networks; Master and Slave. Slave is a basic FCN-8s. The Master is a FCN-8s-F1 (as our baseline) and contains the backward skip connections taken from the Slave network. The forward propagation starts in the Slave network, and then the generated feature maps are fed back to lower layers in the Master network serving as backward skip connections within the Master. Notice that the backward skip connection (blue arrows) are exactly the inverse of the forward skip connections (purple arrows).

for higher layers. In this section, we provide the details of our proposed model. First, we briefly review our first baseline which is ‘FCN-8s’ structure [234] since we adopt it as our basic building blocks in our MSNet model. Second, we further extend and improve FCN-8s to contain more forward skip connections and propose another two strong baselines. Third, we introduce our MSNet model that contains the backward skip connections only and discuss about the specifics of the proposed design. Finally, we further enhance our MSNet so that the Master network is upgraded into one of our improved baselines in addition to its backward skip connections.

6.3.1 Baseline Models

FCN originally are designed to extend image classification networks like the VGG Net [231] to tackle image segmentation. FCN architecture consists of: 1) a downsampling path that contains mainly seven convolutional blocks separated by five pooling layers (stacked in an alternating manner); 2) an upsampling path that contains upsampling layers to recover the original input resolution; 3) skip forward connections that are branched from

Figure 6.4: (Best viewed in colors). A detailed overview of the Baseline model networks; FCN-8s, FCN16-s and FCN-32s. The early design ‘FCN-32s’ does not include skip connections, whereas FCN-16s has one skip connection from *pool4* layer, and FCN-8s has skip connections from both *pool4* and *pool3* layers.

the downsampling path and fused to the upsampling path to aid in recovering the lost spatial information and to produce detailed segmentation. The early design ‘FCN-32s’ does not include skip connections, whereas FCN-16s has one skip connection from *pool4* layer, and FCN-8s has skip connections from both *pool4* and *pool3* layers. Figure 6.4 shows the basic structures of the FCN-32s, FCN-6s and FCN-8s networks. As shown in the downsampling path in the network, almost all the convolutional blocks contain 2-3 convolutions in addition to 2-3 non-linear activation layers (ReLU) inserted in between. Except for the last two convolutional blocks, they contain 3 fully-connected layers, 2 ReLUs and 2 dropout layers.

While FCN-8s only has two forward skip connections to the upsampling layers, we add extra forward skip connections to construct stronger baselines. While designing the baseline structures, our main consideration is: ***skipping from where to where and how many skips are adequate***. We gradually add skips and shrink their lengths from low to high layers while recording the performance. Throughout the experimentation we observe that having at least one forward skip connection within the convolutional blocks and multiple skips from the direct preceding blocks provide most of the improvement gain in the performance (notice that blocks are separated by pooling or dropout layers). However, densely connecting each block to all preceding block is costly (as the network capacity will also grow), thus, the number of skip connections from neighboring blocks should be set properly.

We develop two stronger FCN-8s-based baselines with more forward skip connections. If the input to a convolutional block L is $\mathbf{x}_L$ (in our FCN-8s model, $L = \{1, 2, 3, \dots, 7\}$), the output from it is $\mathbf{y}_L$, which becomes $\mathbf{x}_{L+1}$ after a successive pooling or dropout layers. Mathematically, the general formula for adding N forward skip connections from direct preceding blocks to block L is:

$$\mathbf{y}_L = \mathcal{F}(\mathbf{x}_L) + \sum_{l=L-(N-1)}^L W_l \mathbf{x}_l \quad (6.1)$$

where $1 \leq N \leq L$ and W_l is a convolutional transformation to rematch the feature dimensions¹. $\mathcal{F}(\cdot)$ is a convolutional block function that consists of multiple convolutions and non-linear activations (ReLU).

FCN-8s-F1: the design of our first baseline is a special case of Equation (6.1) where $N = 1$. In this case, each block has one skip connection within itself (from its beginning to its end) and the formula becomes:

$$\mathbf{y}_L = \mathcal{F}(\mathbf{x}_L) + W_L \mathbf{x}_L \quad (6.2)$$

FCN-8s-F2: in the design of our second baseline, we add more skips from previous blocks, where we set $N = 3$ and the formula becomes:

$$\mathbf{y}_L = \mathcal{F}(\mathbf{x}_L) + W_L \mathbf{x}_L + W_{L-1} \mathbf{x}_{L-1} + W_{L-2} \mathbf{x}_{L-2} \quad (6.3)$$

where $W_{L|L-1|L-2}$ are convolutional transformations (not shared). Notice that $W_{L-1|L-2}$ are accompanied with extra pooling layers beside the convolutions to rematch feature map spatial dimensions.

6.3.2 Master-Slave Network Design

In this work, we propose to add backward skip connections to FCNs. However, if we simply add them in an FCN, we will create cyclic loops in the network, thus, training becomes a problem. To overcome this challenge, we propose a Master-Slave network design.

Figure 6.3 shows in details our proposed final structure. The model consists of two FCN-8s networks (*none* of their parameters are shared). Only one network which is the Master is considered to produce reliable final predictions. The other network which is the Slave is only responsible to feed the Master with the required backward skip connections. The Slave is not provided with any backward skip connections from the Master network

¹In our experiments, we notice that adding a convolution layer to transform the feature map even if they originally have similar dimensions works better than directly aggregating them.

and its predictions are totally ignored. However, both networks are optimized to minimize similar losses.

Notice that the backward skip connections are the inverse of our forward skip connections as previously formulated in Equation (6.1). Hence, the general formula for adding P backward skip connections to block L of the Master network is denoted as:

$$\mathbf{y}_L^m = \mathcal{F}(\mathbf{x}_L^m + \sum_{l=L}^{L+(P-1)} U_l \mathbf{y}_l^s) \quad (6.4)$$

where $\mathbf{y}_L^m$ is the output of block L in the Master network and $\mathbf{y}_L^s$ is the output of block L in the Slave network (m and s refer to Master and Slave). U_l is a convolutional transformation applied on the feature maps from the Slave network.

Based on Equation (6.4), our Master network can learn to discover visual patterns from $\mathbf{x}_L^m$ that are more compatible with $\mathbf{y}_L^s$.

In our model designs, we define two structures as counterparts to our previously designed baselines. **MSNet-B1** is the inverse of FCN-8s-F1 while **MSNet-B2** is the inverse of FCN-8s-F2. In MSNet-B1, each block in the Master network is provided with one backward skip from the Slave (from the end of the peer block in the Slave). In MSNet-B2, each block in the Master network is provided with multiple backward skip connections from proceeding blocks in the Slave network (we set $P = 3$).

We further upgrade our Master network to also have internal dense forward skip connections similar to our baselines. Overall, the forward and backward skip connections in block L in the Master network are formulated as:

$$\mathbf{y}_L^m = \mathcal{F}(\mathbf{x}_L^m + \sum_{l=L}^{L+(P-1)} U_l \mathbf{y}_l^s) + \sum_{l=L-(N-1)}^L W_l \mathbf{x}_l^m \quad (6.5)$$

where W_l and U_l are convolutional transformations.

In summary, our model mainly contains three types of components in the down-sampling path: 1) convolutional block; 2) forward-skip-fuse (skip forward through a convolution layer and fuse in a summation layer); 3) backward-skip-fuse (skip backward

Figure 6.5: (Best viewed in colors). An illustration of several convolutional blocks of our MSNet-FB2 with dense skip connections. Notice that the Master network is FCN-8s-F2. We set $P = 3$ and $N = 3$. Here the dense backward skip connections are in inverse directions to the dense forward skip connections. Refer to Figure 6.3 for the Legend.

Algorithm	ADE20K			PASCAL-Context			PASCAL VOC 2011		
	PA(%)	CA(%)	IU	PA(%)	CA(%)	IU(%)	PA(%)	CA(%)	IU(%)
FCN-8s	71.3	40.3	29.4	67.5	52.3	39.1	91.2	77.6	65.5
FCN-8s-F1	73.5	41.2	31.4	70.7	52.3	40.7	91.1	76.4	66.0
MSNet-B1	75.1	42.7	32.9	71.8	52.5	41.7	92.1	77.6	68.3
MSNet-FB1	74.8	41.7	32.5	73.3	53.5	42.1	91.9	77.2	67.7
FCN-8s-F2	73.7	42.0	31.7	69.0	51.1	38.8	91.9	77.5	66.9
MSNet-B2	75.2	43.0	33.3	71.9	50.6	39.7	92.0	78.0	68.6
MSNet-FB2	75.7	43.8	33.8	70.1	50.3	39.9	92.3	80.3	69.2

Table 6.1: Results on the ADE20K [11] (validation set), PASCAL-Context [6] and PASCAL VOC 2011 (validation set) [10]. Refer to Section 6.4.2 for detailed definitions.

through a convolution layer and a summation layer). Forward-skip-fuse may contain an extra pooling layer if the fusion is between multi-scale feature maps. Similarly, backward-skip-fuse may have an upsampling layer to properly fuse the multi-scale feature maps. Details of these components are shown in Figure 6.5. The Figure visualize an example with $L = 3$, $N = 3$ and $P = 3$ in MSNet-FB2 model.

6.4 Experiments

We evaluate our method on three competitive image segmentation benchmark datasets: ADE20K, PASCAL-Context and PASCAL VOC 2011. We provide evaluation details of our proposed network and other hyperparameters and design choices.

6.4.1 Datasets

ADE20K [11] has 20210 training, 2000 validation and 3352 testing images. It is created for the ImageNet scene parsing challenge. The segmentation task in this dataset is to label each pixel to one of the 150 semantic classes.

PASCAL-Context [6] has 4998 training images and 5105 testing images. The images are sampled originally from the PASCAL VOC 2010 dataset and re-labeled at the pixel-level for the segmentation task. In total, there are 540 classes, however, in our experiments we only consider the task of labeling 59 classes (most frequent classes) for evaluation.

PASCAL VOC 2011 [10] have around 11287 training images, 736 images as validation set and 1111 testing images. The task here is to classify each pixel into one of 21 categories, including 20 foreground object classes and one background class.

Evaluation We use three main metrics to evaluate our method following the recent literature: the percentage of all correctly classified pixels - Pixel Accuracy (**PA**), Per-class Accuracy (**CA**), and the Intersection-Over-Union (**IU**).

6.4.2 Baselines and Methods

In summary, our baselines and models are:

FCN-8s: the basic FCN structure with two forward skip connections from *pool3* and *pool4*.

FCN-8s-F1: improved FCN-8s structure that contains more forward skip connections within each convolutional block (refer to Section 6.3.1 for details).

FCN-8s-F2: densely connected FCN-8s structure that contains $N = 3$ forward skip connections (refer to Section 6.3.1 for details).

MSNet-B1: a Master-Slave network model where the backward skip connections from the Slave network to the Master network are in inverse direction to the forward skip connections in FCN-8s-F1. Here, the Master and Slave networks are basic FCN-8s models.

MSNet-B2: similar to MSNet-B1, but the backward skip connections from the Slave network to the Master network are in inverse direction to the forward skip connections proposed in baseline FCN-8s-F2. The number of the backward skips are set to $P = 3$.

MSNet-FB1: similar to MSNet-B1, but the Master is upgraded to be our baseline FCN-8s-F1. The Slave network is basic FCN-8s.

MSNet-FB2: similar to MSNet-B2, but the Master is upgraded to be our baseline FCN-8s-F2. The Slave network is basic FCN-8s.

6.4.3 Optimization Details

MSNet is end-to-end differentiable and trainable using the backpropagation algorithm (we use the stochastic gradient descent (SGD) with momentum). Both the Master and the Slave networks are trained to minimize the cross entropy log-loss averaged over all image patches (we only consider the semantically labeled ones), but they are *not* jointly trained. In the forward pass, the propagation begins in the Slave network. When it reaches the first backward skip connections, it triggers the propagation in the Master

Algorithm	$\sim$ Train (ms)	$\sim$ Test (ms)
FCN-8s	183.4	55.6
FCN-8s-F1	195.4	54.6
FCN-8s-F2	238.4	65.6
MSNet-B1	388.3	108.7
MSNet-B2	411.7	114.7
MSNet-FB1	409.2	113.4
MSNet-FB2	475.4	130.2

Table 6.2: We report training and testing run time on the validation set of ADE20K (averaged over 20 trials for an input image of 384×384 on NVIDIA GeForce GTX Titan X and the companion CPU is Intel Xeon E5-2650 v2 at 2.60GHz).

network to start. The execution order is alternating between both networks to make sure that all the feature maps of the backward skip connections are generated prior to any forward propagation in a lower layer in the Master network. In the backward pass, the execution order is reversed.

Hyperparameters like the momentum is initialized and fixed to 0.9, meanwhile the learning rate is initialized to 10^{-5} , and it decays exponentially with the rate of 10% after 10 epochs, and after 5 epochs when it finish the first 30 epochs. We follow [234] to train our basic FCN-8s models. We initialize all our networks with the VGG ImageNet-pretrained model [231]. We used the publicly available MatConvNet MATLAB implementations [211]. The reported results are based on the model trained for 50 epochs. After we generate the class likelihood per each pixel in the image, we calculate the cross entropy loss for training. Thus, the error signal of each image is averaged across all valid/semantically-labeled pixels (we ignore the contribution of unlabeled pixels in the loss calculation). We also report the learning and inference run time of our baselines and our methods as shown in Table 6.2.

6.4.4 Results

Table 6.1 show the prediction results on the ADE20K, PASCAL-Context and PASCAL VOC 2011 datasets.

Algorithm	Validation			Test
	PA(%)	CA(%)	IU(%)	IU(%)
MSNet-FB1	91.9	77.2	67.7	69.3
MSNet-FB1+CRF	92.4	78.3	68.9	70.3
MSNet-FB2	92.3	80.0	69.2	70.5
MSNet-FB2+CRF	92.8	80.9	70.3	71.4

Table 6.3: Results on the validation and testing sets of PASCAL VOC 2011 [10]. MSNet-*+CRF refers to applying fully-connected CRF on top of the label predictions generated from our MSNet models (post processing step).

Adding Backward Shortcuts helps: our model MSNet-B1 improves over FCN-8s by 3.5%, 2.6% and 2.8% respectively (IU). While MSNet-B2 improves over FCN-8s by 3.9%, 0.6% and 3.1% respectively (IU). The experiments show that the higher layers’ features are indeed important during lower layers’ features extraction. Higher abstractions can help lower layers to learn more informative lower features (as high-level context is explored early).

Backward and Forward Shortcuts are complementary: as shown in Table 6.1, MSNet-FB1 improves over FCN-8s-F1 by 1.1%, 1.4% and 1.7% respectively (IU). While MSNet-FB2 improves over FCN-8s-F2 by 2.1%, 1.1% and 2.3% respectively (IU). The results show that both types of skip connections are complementary and important. We notice the MSNet-FB2 performs better than MSNet-FB1 on PASCAL VOC and ADE20K, but worse on PASCAL Context. This may be due to that MSNet-FB2 is more dense and complex (higher capacity) and thus require more training data to generalize better. For example, ADE20K contains the highest number of training examples and MSNet-FB2 performs well on it.

Backward Shortcuts outperform Forward Shortcuts: interestingly, MSNet-B2 which contains only backward skip connections outperforms both baselines FCN-8s-F1 and FCN-8s-F2 in all datasets. Similarly, MSNet-B1 consistently outperform its counterpart FCN-8s-F1. These findings show that high layers’ features are indeed beneficial for extracting more informative low features.

Figure 6.6: (Best to be zoomed in and viewed in colors) Examples of some qualitative labeling results. For each row, we show the input image, its ground truth, label prediction maps for FCN-8s, FCN-8s-F1, FCN-8s-F2 and our proposed models MSNet-FB1 and MSNet-FB2. The test images are from ADE20K validation set [11] (top) and PASCAL-Context [6] (bottom). The MSNet-FB1 and MSNet-FB2 have extra backward skip connections and improve the quality of the label prediction maps compared with baselines. The FCN-F2 and FCN-F2 baselines improve over FCN-8s as they have more forward skip connections.

MSNets are complementary to CRFs: In order to evaluate the orthogonality of our MSNet with other common methods like CRF, we perform post-processing on top of MSNet using the Fully-connected Conditional Random Fields implementations [212]. Fully-connected CRF is one of the most common contextual modeling methods applied in scene labeling (it is one of the state-of-the-art works that applied in [138, 235]). As shown in Table 6.3, MSNet-FB2+CRF and MSNet-FB1-CRF outperform MSNet-FB2 and MSNet-FB1 by at least 1.0% respectively. This performance gain illustrates that our proposed method is complementary to other existing frameworks.

Some works such as the CNN-CRF model[149] shown in Table 6.4. In their work, they have formulate and trained CRFs and CNN-based pairwise potential functions in order to capture semantic correlations between neighboring patches. They follow a piecewise training strategy to achieve the segmentation task. They report impressive results on the PASCAL VOC 2012 (78% IU). We believe that our work is orthogonal to all other CRF-based methods as in general. Our proposed MS-Net model alone is still faster than typical CRF-based methods since it is solely built on CNN fabrics. Thus, we utilize the GPU resources efficiently and benefit from the parallel implementations. In contrast, CRF-based methods are relatively expensive as they rely and may be restricted to the implementations on the CPU. But through our experimentations with fully-connected CRF models, we find that both of our networks and CRF-based methods can be potentially used for different applications and can also be used together (where CRF can be potentially cascaded after the CNN).

Qualitative Results of MSNets: we show the qualitative accuracy results in Figure 6.6. The quality of the label predictions is significantly improved over FCN-8s as more forward skip connections are engaged (the baselines) and also when backward skip connections are engaged (MSNet models).

Comparison with the State-of-the-art: we compare the performance of our MSNet models with other state-of-the-art methods on image segmentation. Notice that

Algorithm	PA(%)	CA(%)	IU(%)
FCN-8s [234]	65.9	46.5	35.1
FCN-8s [248]	67.5	52.3	39.1
DeepLab [138]	-	-	37.6
DeepLab-CRF [138]	-	-	39.6
HO-CRF [215]	-	-	41.3
CNN-CRF [149]	71.5	53.9	43.3
CRF-RNN [7]	-	-	39.3
ParseNet [216]	67.5	52.3	39.1
ConvPP-8 [217]	-	-	41.0
PixelNet [218]	-	51.5	41.4
O2P [219]	-	-	18.1
CFM [220]	-	-	34.4
BoxSup [221]	-	-	40.5
MSNet-FB1	73.0	53.5	42.1

Table 6.4: Performance comparison on PASCAL-Context [6].

we didn’t engage any additional training data (like from Microsoft Common Objects in Context (MS COCO) dataset [210]). We also didn’t employ any Batch Normalization layers as our main focus is to improve the original design of FCNs. It is also important to mention that there are some methods that use different settings than us like adopting very deep Residual Networks [232] (e.g. [236], [213] and [138]) or employing Dilated convolutions [138]. However, we use VGG-based networks, and note our backward skip techniques can be potentially used as well to enhance their models. The quantitative results comparison on ADE20K, PASCAL-Context and PASCAL VOC 2011 datasets are listed in Table 6.5, 6.4 and 6.6 respectively. Notice that in Table 6.4, we compare on the validation set of ADE20K. For Table 6.6, we evaluate on PASCAL VOC 2011 test set. Our models perform competitively on all datasets to those models which have similar sittings with us.

6.5 Chapter Summary

In this chapter, a new type of skip connection is presented. We upgraded the structure of FCN-8s with two designs: first we add more forward skip connections from lower layers to higher layers (beside its original two skips from *pool3* and *pool4*), and second we propose

Algorithm	PA(%)	CA(%)	IU(%)
SegNet [249]	71.0	31.1	21.6
FCN-8s [234]	71.3	40.3	29.4
DilatedNet [139]	73.6	44.6	32.3
Cascade-SegNet [11]	71.8	37.9	27.5
Cascade-DilatedNet [11]	74.5	45.4	34.9
MSNet-FB2	75.7	43.8	33.8

Table 6.5: Performance comparison on the validation set from ADE20K [11].

Algorithm	IU(%)
BerkeleyRC [226]	39.1
SDS [227]	52.6
R-CNN [228]	47.9
FCN-8s [234]	62.7
FCN-8s [248]	67.5
Zoomout [229]	64.4
CRF-RNN[7]	72.4
MSNet-FB2	70.5

Table 6.6: Average performance comparison on the test set from PASCAL VOC 2011 [10].

new backward skip connections that fuse higher layers feature maps into lower layers. In order to achieve this, we design our model as a pair of Master and Slave networks. They are twin versions of FCN-8s, however, the Master is upgraded with more forward skips. The Slave is only responsible for feeding the Master with backward skip connections from its higher layers to the Master lower layers. The Master is accountable to produce the final prediction maps. Our experiments show that lower layers can adaptively extract more informative and beneficial features if they are aware as early as possible with high layer feature representations.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

This thesis focuses on improving various deep feature learning models. It propose to enhance them to have the ability to share various types of important knowledge across feature classifiers and representations. We address attribute prediction, RGB-D and RGB semantic segmentation tasks. Specifically, we improve CNNs and allow them to share attribute information in a multi-task framework. We improve RNNs and allow them to share the correlated mutual information between multiple data modalities. We also improve CNNs (specifically FCNs) and allow them to explore, share and encode the contextual dependencies among image regions in each local patch representations. We first engaged attention-based models alongside RNNs within an FCN, but then we upgrade the internal design of the FCN itself and allow it to share high-level semantics and contextual cues on its own. We evaluate our proposed deep models on various competitive and popular benchmarks addressing and our models achieve substantial improvements over the baseline methods and competitive performance to (and in many sittings outperform) other state-of-the-art works.

Firstly, we introduce an enhanced deep multi-task learning method to better predict semantic binary attributes. Our proposed multi-task CNN model allows sharing of visual knowledge between different CNNs. We encode semantic group information in our

MTL framework to encourage more sharing between attributes in the same group while discouraging to share is attribute belong to different groups. We introduce to decompose the model parameters into a latent task matrix and a linear combination matrix. The latent task matrix can effectively learn localized feature patterns and it is shared among all the CNN models, where any under-sampled classifier will generalize better through leveraging this sharable latent layer. Our experiments on both attribute benchmark datasets show that our learned multi-task CNN classifiers easily outperform the previous single-task CNN and other classifiers.

Second, we presents a new deep multi-modal method for RGB-D semantic segmentation. We improve the structure of the recurrent neural network to have two crossly-connected quad-directional 2D-RNNs with information transfer layers. Transfer layers are learned to select and extract relevant contextual information between different modalities and help each modality to learn multi-modal context-aware features through the shared information. Each RNN is responsible to process a single data modality, hence, modality-specific contextual information are retained and not lost due to sharing related across-modal features from the other data modalities. Both RNNs are simultaneously optimized, thus, they can process multiple data modalities as efficient as one single RNN.

Third, we focus our study on typical RGB scene labeling. We enhance the current state-of-the-art deep FCN model and generate contextually-aware convolutional features for each patch in the image. In order to effectively leverage the relevant contextual patches to enhance the local classification accuracy, we propose an episodic attention-based contextual memory network. Our model presents a unified framework that mainly consists of FCN convolutional layers, attention-based model, and an RNN to perform context selection, aggregation and refinement. The full model produces context-aware representation for each target patch by aggregating the activated context and its original local representation.

Our experiments show that the proposed method significantly boosts the labeling performance of the original FCN by incorporating the selected context. More importantly,

our method achieves competitive performance on the public PASCAL-Context, SIFT Flow and PASCAL VOC 2011 scene labeling benchmarks when compared with other state-of-the-art work.

Last, we zoom in our focus on the internal design on FCN model and improve it to have the ability to propagate important high-level contextual features. In our work, a new type of skip connections is presented which is the backward skip connections. We upgraded the structure of FCN-8s with two designs: first we add more forward skip connections from lower layers to higher layers (beside its original two skips from *pool3* and *pool4*), and second we propose new backward skip connections that fuse higher layers feature maps into lower layers. Higher layers feature contain high-level semantics that are important and useful to lower layers. In order to achieve this, we design our model as a pair of Master and Slave networks. They are twin versions of FCN-8s, however, the Master is upgraded with more forward skips. The Slave is only responsible for feeding the Master with backward skip connections from its higher layers to the Master lower layers. The Master is accountable to produce the final prediction maps.

Our experiments show that lower layers can adaptively extract more informative and beneficial features if they are aware as early as possible with high layer feature representations.

7.2 Future Works

I would like to discuss some potential research directions which can be explored in the future.

In our first work in Chapter 3, we introduce to decompose the model parameters into a latent task matrix and a linear combination matrix. The latent task matrix is intend to be shared among all of the classifiers. The learned localized feature patterns in the latent matrix allow effective knowledge sharing among all the CNN models, where any under-sampled classifier will generalize better through leveraging this sharable latent

layer. The construction of this latent task matrix is very crucial for effective and useful knowledge sharing. Specifically, we would like to explore other potential ways of the latent task matrix decomposition so that it can be informative enough to generate an efficient description of the input image in terms of either semantic or latent attributes.

In our second work as presented in Chapter 4, we process only two data modalities using our crossly-connected RNNs. RGB-D itself is a big research area with many active researchers. However, the framework can be generalized to problems with multi-modalities and with the need of modeling context. Examples include joint image-text analysis and multi-modal action analysis. Our method is scalable and can be extended to address more than two modalities. In our future work, we will evaluate the effectiveness and the scalability of the transfer layers on more modalities (modalities > 2) and process new types of input data signals like audio and textual data modalities.

In our third work as introduced in Chapter 5, we believe the performance of our episodic-CAMN model can be further improved by enhancing some internal functionalities such as the relevance learning. As presented in the CAN model, to calculate the attention heat map over the image patch locations, the relevance between the target patch and the remaining patches in the image is learned. In our current framework, we calculate the similarity in simple manner, however, we believe it is very important to advance this part and engage more sophisticated operations, so it can potentially capture subtle relationships and interactions between the referenced local patch and its context.

In our last work as introduced in Chapter 6, we propose our backward skip connections mainly in VGG-base FCN models. We will explore other design frameworks like ResNets and Inception. We will upgrade the design of our basic Master and Slave networks to be ResNets (50 layers or more) but advanced with our proposed backward and forwards skip connections (also our introduced densely connected designs are applicable). We will also continue to apply our model on other important visual tasks like image classification and object detection. In these tasks, high-level semantics are deemed to be important

to boost the classifiers performance. Our internal design is orthogonal to almost all recently proposed deep network, and has the ability to allow important high-level semantic information to be propagated from higher layers to lower ones.

Deep networks are efficient and can work because they are trained to learn to extract features in an end-to-end manner. They are data-adaptive models that can automatically learn powerful and discriminative features for the task in hand. Concretely, deep networks are well motivated in the community because of their capability to learn features end-to-end. CNNs have shown extraordinary performance (achieve the state-of-the-art results most of the time) on various tasks. Throughout our proposed models in the chapters, all our methods are basically built upon the recent popular deep learning models which are CNNs, FCNs and RNNs alongside attention-based models. We believe other visual and non-visual recognition tasks that require knowledge sharing can also benefit from all our presented improved deep models.

Author's Publications

- (i) **Abrar H. Abdulnabi**, Gang Wang, Jiwen Lu and Kui Jia. "Mutli-Task CNN Model for Attribute Prediction.", IEEE Transactions on Multimedia (TMM), 2015.
- (ii) **Abrar H. Abdulnabi**, Bing Shuai, Stefan Winkler and Gang Wang. "Episodic CAMN: Contextual Attention-based Memory Networks for Scene Labeling." in proceedings of the IEEE Computer Vision and Pattern Recognition Conference (CVPR), 2017.
- (iii) **Abrar H. Abdulnabi**, Bing Shuai, Zhen Zuo, Lap-Pui Chau and Gang Wang. "Multimodal Recurrent Neural Networks with Information Transfer Layers for Indoor Scene Labeling.", IEEE Transactions on Multimedia (TMM), 2017.
- (iv) **Abrar H. Abdulnabi**, Stefan Winkler and Gang Wang. "Beyond Forward Shortcuts: Fully Convolutional Master-Slave Networks (MSNets) with Backward Skip Connections for Scene Labeling.", arXiv, 2017
- (v) Jinghua Wang, **Abrar H. Abdulnabi**, Gang Wang, Chengde Wan and Tian-Tsong Ng. "Towards Predicting the Likeability of Fashion Images.", arXiv, 2017
- (vi) Huijing Zhan, Yan Wang, **Abrar H. Abdulnabi**, Sheng Li, Dennis Sng, Alex C. Kot and Simon See. "Visual Product Search for Fashion Domain: From Hand-crafted features to Deep Learning.", Princess Maha Chakri Sirindhorn Congress Interdisciplinary Approach to Sustainable Research and Developments (INRIT), 2015.

References

- [1] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in *CVPR*, 2009.
- [2] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgb-d images,” in *ECCV*, 2012.
- [3] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *ECCV*, 2014.
- [4] A. Graves, S. Fernandez, and J. Schmidhuber, “Multi-dimensional recurrent neural networks,” in *ICANN*, 2007.
- [5] H. Chen, A. Gallagher, and B. Girod, “Describing clothing by semantic attributes,” in *ECCV*, 2012.
- [6] R. Mottaghi, X. Chen, X. Liu, N. Cho, S. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in *CVPR*, 2014.
- [7] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, “Conditional random fields as recurrent neural networks,” in *ICCV*, 2015.
- [8] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes challenge 2011 (VOC2011) results.” <http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html>.

REFERENCES

- [9] C. Liu, J. Yuen, and A. Torralba, “Nonparametric scene parsing: Label transfer via dense scene alignment,” in *CVPR*, 2009.
- [10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.” <http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html>.
- [11] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Semantic understanding of scenes through the ADE20K dataset,” *CoRR*, vol. abs/1608.05442, 2016.
- [12] H. Chen, A. Gallagher, and B. Girod, “Describing clothing by semantic attributes,” in *ECCV*, 2012.
- [13] N. Silberman and R. Fergus, “Indoor scene segmentation using a structured light sensor,” in *ICCV Workshops*, 2011.
- [14] B. S. Manjunath and W. Y. Ma, “Texture features for browsing and retrieval of image data,” *TPAMI*, 1996.
- [15] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, 2004.
- [16] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *TPAMI*, 2002.
- [17] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *International Conference on Computer Vision and Pattern Recognition*, 2005.
- [18] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *CVPR*, 2011.

REFERENCES

- [19] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR*, 2006.
- [20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *CVPR*, 2010.
- [21] A. Krizhevsky, S. Ilya, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [22] C. Corinna and V. Vladimir, “Support-vector networks,” *Machine learning*, 1995.
- [23] F. Nir, G. Dan, and G. Moises, “Bayesian network classifiers,” *Machine Learning*, 1997.
- [24] A. Graves, S. Fernandez, and J. Schmidhuber, “Multi-dimensional recurrent neural networks,” in *ICANN*, 2007.
- [25] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: an astounding baseline for recognition,” in *CVPR Workshops*, 2014.
- [26] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.
- [27] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, “PANDA: pose aligned networks for deep attribute modeling,” in *CVPR*, 2014.
- [28] S. R, V. Y, and J. C, “Relative parts: Distinctive parts for learning relative attributes,” in *CVPR*, 2014.
- [29] P. Luo, X. Wang, and X. Tang, “A deep sum-product architecture for robust facial attributes analysis,” in *ICCV*, 2013.
- [30] H. Lee, R. Grosse, R. Ranganath, and A. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *ICML*, 2009.

REFERENCES

- [31] 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,” in *ICML*, 2013.
- [32] J. Read, B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” in *ECML PKDD*, 2009.
- [33] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *Int J Data Warehousing and Mining*, 2007.
- [34] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, “CNN: single-label to multi-label,” *arXiv*, 2014.
- [35] D. Jayaraman, F. Sha, and K. Grauman, “Decorrelating semantic visual attributes by resisting the urge to share,” in *CVPR*, 2014.
- [36] Q. Zhou, G. Wang, K. Jia, and Q. Zhao, “Learning to share latent tasks for action recognition,” in *ICCV*, 2013.
- [37] C. Rich, “Multitask learning,” *Machine Learning*, 1997.
- [38] C. Li, J. Zhu, and J. Chen, “Bayesian max-margin multi-task learning with data augmentation,” in *ICML*, 2014.
- [39] S. Kim and E. P. Xing, “Tree-guided group lasso for multi-task regression with structured sparsity,” in *ICML*, 2010.
- [40] S. J. Hwang, F. Sha, and K. Grauman, “Sharing features between objects and their attributes,” in *CVPR*, 2011.
- [41] A. Argyriou, T. Evgeniou, and M. Pontil, “Multi-task feature learning,” in *NIPS*, 2007.

REFERENCES

- [42] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *ICML*, 2011.
- [43] N. Srivastava and R. Salakhutdinov, “Multimodal learning with deep boltzmann machines,” in *NIPS*, 2012.
- [44] K. Sohn, W. Shang, and H. Lee, “Improved multimodal deep learning with variation of information,” in *NIPS*, 2014.
- [45] H. Lee, C. Ekanadham, and A. Y. NG, “Sparse deep belief net model for visual area v2,” in *NIPS*, 2007.
- [46] G. Heitz and D. Koller, “Learning spatial context: Using stuff to find things,” in *ECCV*, 2008.
- [47] G. Singh and J. Kosecka, “Nonparametric scene parsing with adaptive feature relevance and semantic context,” in *CVPR*, 2013.
- [48] A. Sharma, O. Tuzel, and M.-Y. Liu, “Recursive context propagation network for semantic scene labeling,” in *NIPS*, 2014.
- [49] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *TPAMI*, 2013.
- [50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [51] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the devil in the details: Delving deep into convolutional nets,” in *BMVC*, 2014.
- [52] A. H. Abdulnabi, S. Winkler, and G. Wang, “Beyond forward shortcuts: Fully convolutional master-slave networks (msnets) with backward skip connections for semantic segmentation,” *arXiv*, 2017.

REFERENCES

- [53] A. H. Abdulnabi, B. Shuai, S. Winkler, and G. Wang, “Episodic camn: Contextual attention-based memory networks with iterative feedback for scene labeling,” in *CVPR*, 2017.
- [54] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ICLR*, 2015.
- [55] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe, “Deep convolutional ranking for multilabel image annotation,” *arXiv*, 2013.
- [56] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [57] 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,” in *ICML*, 2013.
- [58] A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” in *NIPS*, 2008.
- [59] O. Irsoy and C. Cardie, “Opinion mining with deep recurrent neural networks,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.
- [60] A. Graves, A. Mohamed, and G. E. Hinton, “Speech recognition with deep recurrent neural networks,” in *ICASSP*, 2013.
- [61] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press, 2001.
- [62] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *MIT Press*, 1997.

REFERENCES

- [63] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural Networks*, 2005.
- [64] A. Graves, S. Fernandez, and J. Schmidhuber, “Bidirectional lstm networks for improved phoneme classification and recognition,” in *ICANN*, 2005.
- [65] B. Dzmitry, C. Kyunghyun, and B. Yoshua, “Neural machine translation by jointly learning to align and translate,” in *ICLR*, 2015.
- [66] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *EMNLP*, 2015.
- [67] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *NIPS*, 2015.
- [68] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *ICML*, 2015.
- [69] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, “Teaching machines to read and comprehend,” in *EMNLP*, 2015.
- [70] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for sentence summarization,” in *EMNLP*, 2015.
- [71] W. Ling, L. Chu-Cheng, Y. Tsvetkov, S. Amir, R. F. Astudillo, C. Dyer, A. W. Black, and I. Trancoso, “Not all contexts are created equal: Better word representations with variable attention,” in *EMNLP*, 2015.
- [72] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, “Ask me anything: Dynamic memory networks for natural language processing,” in *ICML*, 2016.

- [73] C. Xiong, S. Merity, and R. Socher, “Dynamic memory networks for visual and textual question answering,” in *ICML*, 2016.
- [74] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” in *NIPS*, 2014.
- [75] 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 *CVPR*, 2015.
- [76] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. Huang, “Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks,” in *ICCV*, 2015.
- [77] J. Ba, V. Mnih, and K. Kavukcuoglu, “Multiple object recognition with visual attention,” in *ICLR*, 2015.
- [78] I. Sutskever, O. Vinyals, and Q. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014.
- [79] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *CVPR*, 2016.
- [80] J. Weston, S. Chopra, and A. Bordes, “Memory networks,” *arXiv*, 2014.
- [81] A. Graves, G. Wayne, and I. Danihelka, “Neural Turing machines,” *arXiv*, 2014.
- [82] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “Weakly supervised memory networks,” *arXiv*, 2015.
- [83] K. Tran, A. Bisazza, and C. Monz, “Recurrent memory network for language modeling,” *arXiv*, 2016.

REFERENCES

- [84] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” in *NIPS Workshop*, 2014.
- [85] L. Itti and C. Koch, “Computational modelling of visual attention,” *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.
- [86] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, “Learning sentiment-specific word embedding for twitter sentiment classification,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.
- [87] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, “Describing objects by their attributes,” in *CVPR*, 2009.
- [88] R. Sandeep, Y. Verma, and C. Jawahar, “Relative parts: Distinctive parts for learning relative attributes,” in *CVPR*, 2014.
- [89] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by betweenclass attribute transfer,” in *CVPR*, 2009.
- [90] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, R. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. Blaschko, D. Weiss, B. Taskar, K. Simonyan, N. Saphra, and S. Mohamed, “Understanding objects in detail with fine-grained attributes,” in *CVPR*, 2014.
- [91] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang, “Two-dimensional active learning for image classification,” in *CVPR*, 2008.
- [92] D. Jia, D. Nan, J. Yangqing, F. Andrea, M. Kevin, B. Samy, L. Yuan, N. Hartmut, and A. Hartwig, “Large-scale object classification using label relation graphs,” in *ECCV*, 2014.
- [93] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia, “Multi-task cnn model for attribute prediction,” *IEEE TMM*, 2015.

- [94] J. Wang, A. H. Abdulnabi, G. Wang, C. Wan, and T. Ng, “Towards predicting the likeability of fashion images,” *arXiv*, 2015.
- [95] B. Hariharan, L. Zelnik-manor, S. V. N. Vishwanathan, and M. Varma, “Large scale max-margin multi-label classification with priors,” in *ICML*, 2010.
- [96] P. Rai and H. D. Iii, “Infinite predictor subspace models for multitask learning,” in *AISTATS*, 2010.
- [97] M. Yang, Y. Li, and ZhongFei, “Multi-task learning with gaussian matrix generalized inverse gaussian model,” in *ICML*, 2013.
- [98] Y. Zhang and J. Schneider, “Learning multiple tasks with a sparse matrix-normal penalty,” in *NIPS*, 2010.
- [99] G. Pinghua, Y. Jieping, and Z. Changshui, “Robust multi-task feature learning,” in *ACM SIGKDD*, 2012.
- [100] A. Kumar and H. Daume, “Learning task grouping and overlap in multi-task learning,” in *ICML*, 2012.
- [101] S. Gupta, P. Arbelaez, R. Girshick, and J. Malik, “Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation,” *IJCV*, 2015.
- [102] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, “Indoor semantic segmentation using depth information,” in *ICLR*, 2013.
- [103] X. Ren, L. Bo, and D. Fox, “Rgb-d scene labeling: Features and algorithms,” in *CVPR*, 2012.
- [104] A. H. Abdulnabi, B. Shuai, Z. Zuo, L.-P. Chau, and W. Gang, “Multimodal recurrent neural networks with information transfer layers for indoor scene labeling,” *IEEE TMM*, 2017.

REFERENCES

- [105] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *CVPR*, 2015.
- [106] A. Wang, J. Lu, G. Wang, J. Cai, and T. Cham, “Multi-modal unsupervised feature learning for rgb-d scene labeling,” in *ECCV*, 2014.
- [107] S. Tang, X. Wang, X. Lv, T. Han, J. Keller, Z. He, M. Skubic, and S. Lao, “Histogram of oriented normal vectors for object recognition with a depth sensor,” in *ACCV*, 2013.
- [108] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, “Indoor semantic segmentation using depth information,” in *ICLR*, 2013.
- [109] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from RGBD images,” in *ECCV*, 2012.
- [110] X. Ren, L. Bo, and D. Fox, “RGB-D scene labeling: Features and algorithms,” in *CVPR*, 2012.
- [111] O. Kahler and I. Reid, “Efficient 3D scene labeling using fields of trees,” in *ICCV*, 2013.
- [112] D. Zhuo, T. Sinisa, and J. L. Longin, “Semantic segmentation of RGBD images with mutex constraints,” in *ICCV*, 2015.
- [113] X. He, R. Zemel, and M. Carreira-Perpindn, “Multiscale conditional random fields for image labeling,” in *CVPR*, 2004.
- [114] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Scene parsing with multiscale feature learning, purity trees, and optimal covers,” in *ICML*, 2012.
- [115] P. H. O. Pinheiro and R. Collobert, “Recurrent convolutional neural networks for scene labeling,” in *ICML*, 2014.

REFERENCES

- [116] W. Wang and Z. Zhou, “A new analysis of co-training,” in *ICML*, 2010.
- [117] Z. Wang, S. Chen, and T. Sun, “MultiK-MHKS: A novel multiple kernel learning algorithm,” *TPAMI*, 2008.
- [118] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, “Deep canonical correlation analysis,” in *ICML*, 2013.
- [119] C. Xu, D. Tao, and C. Xu, “A survey on multi-view learning,” *arXiv*, 2013.
- [120] N. Dalal, B. Triggs, and C. Schmid, “Human detection using oriented histograms of flow and appearance,” in *ECCV*, 2006.
- [121] M. Rapus, S. Munder, G. Barattoff, and J. Denzler, “Pedestrian recognition using combined low-resolution depth and intensity images,” *Intelligent Vehicles Symposium*, 2008.
- [122] W. V. D. Mark and D. M. Gavrila, “Real-time dense stereo for intelligent vehicles,” *TITS*, 2006.
- [123] M. Rohrbach, M. Enzweiler, and D. M. Gavrila, “High-level fusion of depth and intensity for pedestrian classification,” in *Pattern Recognition: 31st DAGM Symposium*, 2009.
- [124] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” in *IJRR*, 2014.
- [125] D. Gavrila and S. Munder, “Multi-cue pedestrian detection and tracking from a moving vehicle,” *IJCV*, 2007.
- [126] A. Ess, B. Leibe, and L. V. Gool, “Depth and appearance for mobile scene analysis,” in *ICCV*, 2007.

REFERENCES

- [127] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool, “Dynamic 3d scene analysis from a moving vehicle,” in *CVPR*, 2007.
- [128] Z. Hong, X. Mei, D. Prokhorov, and D. Tao, “Tracking via robust multi-task multi-view joint sparse representation,” in *ICCV*, 2013.
- [129] J. He and R. Lawrence, “A graph-based framework for multi-task multi-view learning,” in *ICML*, 2011.
- [130] S. Kim and E. P. Xing, “Tree-guided group lasso for multi-task regression with structured sparsity,” in *ICML*, 2010.
- [131] H. Hotelling, “Relations between two sets of variates,” *Biometrika*, 1936.
- [132] H. Wolfgang and S. Leopold, “Canonical correlation analysis,” *Applied Multivariate Statistical Analysis*, 2007.
- [133] S. Gupta, J. Hoffman, and J. Malik, “Cross modal distillation for supervision transfer,” in *CVPR*, 2016.
- [134] Z. Yan, H. Zhang, Y. Jia, T. Breuel, and Y. Yu, “Combining the best of convolutional layers and recurrent layers: A hybrid network for semantic segmentation,” *arXiv*, 2016.
- [135] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.
- [136] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *TPAMI*, 2013.
- [137] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected CRFs,” in *ICLR*, 2015.

REFERENCES

- [138] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *arXiv*, 2016.
- [139] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *ICLR*, 2016.
- [140] F. Visin, A. Romero, K. Cho, M. Matteucci, M. Ciccone, K. Kastner, Y. Bengio, and A. Courville, “ReSeg: A recurrent neural network for object segmentation,” *arXiv*, 2016.
- [141] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki, “Scene labeling with lstm recurrent neural networks,” in *CVPR*, 2015.
- [142] B. Shuai, Z. Zuo, G. Wang, and B. Wang, “DAG-Recurrent neural networks for scene labeling,” in *CVPR*, 2016.
- [143] A. Sharma, O. Tuzel, and D. Jacobs, “Deep hierarchical parsing for semantic segmentation,” in *CVPR*, 2015.
- [144] G. Heitz and D. Koller, “Learning spatial context: Using stuff to find things,” in *ECCV*, 2008.
- [145] G. Singh and J. Kosecka, “Nonparametric scene parsing with adaptive feature relevance and semantic context,” in *CVPR*, 2013.
- [146] J. Yang, B. Price, S. Cohen, and M.-H. Yang, “Context driven scene parsing with attention to rare classes,” in *CVPR*, 2014.
- [147] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in *CVPR*, 2014.

REFERENCES

- [148] F. Tung and J. Little, “CollageParsing: Nonparametric scene parsing by adaptive overlapping windows,” in *ECCV*, 2014.
- [149] G. Lin, C. Shen, I. D. Reid, and A. V. D. Hengel, “Efficient piecewise training of deep structured models for semantic segmentation,” in *CVPR*, 2016.
- [150] Z. Liu, X. Li, P. L. C. C. Loy, and X. Tang, “Semantic image segmentation via deep parsing network,” in *ICCV*, 2015.
- [151] J. L. K. J. Abrar H. Abdulnabi, Gang Wang, “Multi-task cnn model for attribute prediction,” *TMM*, 2015.
- [152] B. Siddiquie, R. Feris, and L. Davis, “Image ranking and retrieval based on multi-attribute queries,” in *CVPR*, 2011.
- [153] D. Jayaraman and K. Grauman, “Zero shot recognition with unreliable attributes,” *arXiv*, 2014.
- [154] V. Ferrari and A. Zisserman, “Learning visual attributes,” in *NIPS*, 2007.
- [155] O. Russakovsky and L. Fei-fei, “Attribute learning in large-scale datasets,” in *ECCV*, 2010.
- [156] G.-J. Qi, C. Aggarwal, Q. Tian, H. Ji, and T. Huang, “Exploring context and content links in social media: A latent space method,” *T-PAMI*, 2012.
- [157] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, “Correlative multi-label video annotation,” in *ACM MM*, 2007.
- [158] G.-J. Qi, X.-S. Hua, and H.-J. Zhang, “Learning semantic distance from community-tagged media collection,” in *ACM MM*, 2009.
- [159] S. Chang, G.-J. Qi, J. Tang, Q. Tian, Y. Rui, and T. S. Huang, “Multimedia lego: Learning structured model by probabilistic logic ontology tree,” in *ICDM*, 2013.

REFERENCES

- [160] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, “Attribute and simile classifiers for face verification,” in *ICCV*, 2009.
- [161] S. Ma, S. Sclaroff, and N. Ikizler-Cinbis, “Unsupervised learning of discriminative relative visual attributes,” in *ECCV Workshops*, 2012.
- [162] D. Parikh and K. Grauman, “Relative attributes,” in *ICCV*, 2011.
- [163] W. Scheirer, N. Kumar, P. Belhumeur, and T. Boult, “Multi-attribute spaces: Calibration for attribute fusion and similarity search,” in *CVPR*, 2012.
- [164] G. Wang and D. Forsyth, “Joint learning of visual attributes, object classes and visual saliency,” in *CVPR*, 2009.
- [165] R. Salakhutdinov, A. Torralba, and J. Tenenbaum, “Learning to share visual appearance for multiclass object detection,” in *CVPR*, 2011.
- [166] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr, “BING: Binarized normed gradients for objectness estimation at 300fps,” in *CVPR*, 2014.
- [167] S. Chen and Y. Wang, “Convolutional neural network and convex optimization,” *University of California San Diego*, 2014.
- [168] Y. Tang, “Deep learning using linear support vector machines,” in *ICML*, 2015.
- [169] X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. Xing, “Smoothing proximal gradient method for general structured sparse learning,” *The Annals of Applied Statistics*, 2012.
- [170] S. Kim and E. Xing, “Tree-guided group lasso for multi-task regression with structured sparse,” in *ICML*, 2010.
- [171] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *RSS*, 1996.

REFERENCES

- [172] E. P. X. Seyoung Kim, “Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping,” in *ICML*, 2012.
- [173] F. Bach, “Consistency of the group lasso and multiple kernel learning,” *JMLR*, 2008.
- [174] P. Tseng, “On accelerated proximal gradient methods for convex-concave optimization,” in *ICML*, 2008.
- [175] C. Lampert, H. Nickisch, and S. Harmeling, “Attribute-based classification for zero-shot visual object categorization,” *TPAMI*, 2014.
- [176] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [177] A. Gallagher and T. Chen, “Clothing cosegmentation for recognizing people,” in *CVPR*, 2008.
- [178] M. Ozeki and T. Okatani, “Understanding convolutional neural networks in terms of category-level attributes,” in *ACCV*, 2014.
- [179] D. Wang, P. Cui, M. Ou, and W. Zhu, “Learning compact hash codes for multimodal representations using orthogonal deep structure,” *IEEE Transactions on Multimedia*, vol. 17, pp. 1404–1416, Sept 2015.
- [180] C. Ding and D. Tao, “Robust face recognition via multimodal deep face representation,” *IEEE Transactions on Multimedia*, vol. 17, pp. 2049–2058, Nov 2015.
- [181] A. Wang, J. Lu, J. Cai, T. J. Cham, and G. Wang, “Large-margin multi-modal deep learning for rgb-d object recognition,” *IEEE Transactions on Multimedia*, vol. 17, pp. 1887–1898, Nov 2015.

REFERENCES

- [182] S. C. H. Hoi and M. R. Lyu, “A multimodal and multilevel ranking scheme for large-scale video retrieval,” *IEEE Transactions on Multimedia*, vol. 10, pp. 607–619, June 2008.
- [183] S. S. Mukherjee and N. M. Robertson, “Deep head pose: Gaze-direction estimation in multimodal video,” *IEEE Transactions on Multimedia*, vol. 17, pp. 2094–2107, Nov 2015.
- [184] H. Pirsiavash and D. Ramanan, “Detecting activities of daily living in first-person camera views,” in *CVPR*, 2012.
- [185] Y. J. Lee, J. Ghosh, and K. Grauman, “Discovering important people and objects for egocentric video summarization,” in *CVPR*, 2012.
- [186] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, “Convolutional recurrent neural networks: learning spatial dependencies for image representation,” in *CVPR Workshops*, 2015.
- [187] B. Shuai, Z. Zuo, and G. Wang, “Quaddirectional 2d-recurrent neural networks for image labeling,” in *Signal Processing Letters*, 2015.
- [188] A. Krizhevsky, Sutskever, Ilya, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.
- [189] J. L. Elman, “Finding structure in time,” in *Cognitive Science*, 1990.
- [190] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, “Advances in optimizing recurrent networks,” in *ICASSP*, 2013.
- [191] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *arXiv preprint*, 2012.

REFERENCES

- [192] H. Fan, X. Mei, D. Prokhorov, and H. Ling, “Rgb-d scene labeling with multimodal recurrent neural networks,” in *CVPR Workshops*, 2017.
- [193] D. Pei, H. Liu, Y. Liu, and F. Sun, “Unsupervised multimodal feature learning for semantic image segmentation,” *IJCNN*, 2013.
- [194] J. Stuckler, B. Waldvogel, H. Schulz, and S. Behnke, “Dense real-time mapping of object-class semantics from rgb-d video,” *Journal Real-Time Image Processing*, 2014.
- [195] S. H. Khan, M. B. andFerdous Sohel, and R. Togneri, “Geometry driven semantic labeling of indoor scenes,” in *ECCV*, 2014.
- [196] A. C. Muller and S. Behnke, “Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images,” in *ICRA*, 2014.
- [197] S. Gupta, P. Arbelaez, and J. Malik, “Perceptual organization and recognition of indoor scenes from rgb-d images,” in *CVPR*, 2013.
- [198] C. Cadena and J. Kosecka, “Semantic parsing for priming object detection in RGB-D scenes,” *Workshop on Semantic Perception Mapping and Exploration*, 2013.
- [199] A. Hermans, G. Floros, and B. Leibe, “Dense 3D semantic mapping of indoor scenes from RGB-D images,” in *ICRA*, 2014.
- [200] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.
- [201] S. R. Kaiming He, Xiangyu Zhang and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [202] E. S. Jonathan Long and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.

REFERENCES

- [203] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *ICLR*, 2016.
- [204] J. Elman, “Finding structure in time,” *Cognitive Science*, 1990.
- [205] T. Zhuowen, “Auto-context and its application to high-level vision tasks,” in *CVPR*, 2008.
- [206] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, “Human pose estimation with iterative error feedback,” in *CVPR*, 2016.
- [207] B. N. Lundstrom, M. H. Higgs, W. J. Spain, and A. L. Fairhall, “Fractional differentiation by neocortical pyramidal neurons,” *Nature Neuroscience*, 2008.
- [208] M. London and M. Hausser, “Dendritic computation,” *Annual Review of Neuroscience*, 2005.
- [209] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *CVPR*, 2009.
- [210] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, and L. Z. P. Dollar, “Microsoft COCO: Common objects in context,” *arXiv*, 2014.
- [211] A. Vedaldi and K. Lenc, “MatConvNet – convolutional neural networks for matlab,” in *ACM MM*, 2015.
- [212] P. Krahenbuhl and V. Koltun, “Efficient inference in fully connected CRFs with Gaussian edge potentials,” in *NIPS*, 2011.
- [213] Z. Wu, C. Shen, and A. Hengel, “Bridging category-level and instance-level semantic image segmentation,” *arXiv*, 2016.

REFERENCES

- [214] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *TPAMI*, 2016.
- [215] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr, “Higher order conditional random fields in deep neural networks,” in *ECCV*, 2016.
- [216] W. Liu, A. Rabinovich, and A. Berg, “ParseNet: Looking wider to see better,” in *ICLR Workshop*, 2016.
- [217] S. Xie, X. Huang, and Z. Tu, “Top-down learning for structured labeling with convolutional pseudoprior,” in *ECCV*, 2016.
- [218] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan, “PixelNet: Towards a general pixel-level architecture,” *arXiv*, 2016.
- [219] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic segmentation with second-order pooling,” in *ECCV*, 2012.
- [220] J. Dai, K. He, and J. Sun, “Convolutional feature masking for joint object and stuff segmentation,” in *CVPR*, 2015.
- [221] J. Dai, K. He, and J. Sun, “BoxSup: exploiting bounding boxes to supervise convolutional networks for semantic segmentation,” in *ICCV*, 2015.
- [222] C. Liu, J. Yuen, and A. Torralba, “SIFT Flow: Dense correspondence across scenes and its applications,” *TPAMI*, 2011.
- [223] J. Tighe and S. Lazebnik, “Finding Things: Image parsing with regions and per-exemplar detectors,” in *CVPR*, 2013.
- [224] M. Liang, X. Hu, and B. Zhang, “Convolutional neural networks with intra-layer recurrent connections for scene labeling,” in *NIPS*, 2015.

REFERENCES

- [225] H. Caesar, J. R. R. Uijlings, and V. Ferrari, “Joint calibration for semantic segmentation,” in *BMVC*, 2015.
- [226] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” *TPAMI*, 2011.
- [227] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, “Simultaneous detection and segmentation,” in *ECCV*, 2014.
- [228] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *CVPR*, 2014.
- [229] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, “Feedforward semantic segmentation with zoom-out features,” in *CVPR*, 2015.
- [230] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [231] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.
- [232] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015.
- [233] S. Ren, K. He, R. B. Girshick, and J. Sun, “Faster R-CNN: towards real-time object detection with region proposal networks,” *CoRR*, vol. abs/1506.01497, 2015.
- [234] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.
- [235] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” in *ICLR*, 2015.

REFERENCES

- [236] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” *CoRR*, vol. abs/1612.01105, 2016.
- [237] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *ECCV*, 2014.
- [238] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *CoRR*, vol. abs/1505.00387, 2015.
- [239] M. Drozdal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, “The importance of skip connections in biomedical image segmentation,” *CoRR*, vol. abs/1608.04117, 2016.
- [240] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” *CoRR*, vol. abs/1603.09382, 2016.
- [241] S. Singh, D. Hoiem, and D. A. Forsyth, “Swapout: Learning an ensemble of deep architectures,” *CoRR*, vol. abs/1605.06465, 2016.
- [242] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” *CoRR*, vol. abs/1608.06993, 2016.
- [243] S. Jegou, M. Drozdal, D. Vazquez, A. Romero, and Y. Bengio, “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation,” *CoRR*, vol. abs/1611.09326, 2016.
- [244] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *CoRR*, vol. abs/1605.07146, 2016.
- [245] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *CoRR*, vol. abs/1603.08029, 2016.

REFERENCES

- [246] P. Dayan and L. F. Abbott, “Theoretical neuroscience,” *The MIT Press*, vol. 806, 2001.
- [247] G. Deco and T. S. Lee, “The role of early visual cortex in visual integration: a neural model of recurrent interaction,” *European Journal of Neuroscience*, 2004.
- [248] L. Jonathan, S. Evan, and D. Trevor, “Fully convolutional networks for semantic segmentation,” *CoRR*, 2016.
- [249] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *CoRR*, vol. abs/1511.00561, 2015.