Weidi Xie
Wolfson College

D.Phil Thesis
Michaelmas 2017

Deep Neural Networks in Computer Vision and Biomedical Image Analysis

Abstract

This thesis proposes different models for a variety of applications, such as semantic segmentation, in-the-wild face recognition, microscopy cell counting and detection, standardized re-orientation of 3D ultrasound fetal brain and Magnetic Resonance (MR) cardiac video segmentation. Our approach is to employ the large-scale machine learning models, in particular deep neural networks. Expert knowledge is either mathematically modelled as a differentiable hidden layer in the Artificial Neural Networks, or we tried to break the complex tasks into several small and easy-to-solve tasks.

Multi-scale contextual information plays an important role in pixel-wise prediction, e.g. semantic segmentation. To capture the spatial contextual information, we present a new block for learning receptive field adaptively by within-layer recurrence. While interleaving with the convolutional layers, receptive fields are effectively enlarged, reaching across the entire feature map or image. The new block can be initialized as identity and inserted into any pre-trained networks, therefore taking benefit from the "pre-train and fine-tuning" paradigm.

Current face recognition systems are mostly driven by the success of image classification, where the models are trained to by identity classification. We propose a multi-column deep comparator networks for face recognition. The architecture takes two sets (each contains an arbitrary number of faces) of images or frames as inputs, facial part-based (e.g. eyes, noses) representations of each set are pooled out, dynamically calibrated based on the quality of input images, and further compared with local "experts" in a pairwise way.

Unlike the computer vision applications, collecting data and annotation is usually more expensive in biomedical image analysis. Therefore, the models that can be trained with fewer data and weaker annotations are of great importance. We approach the microscopy cell counting and detection based on density estimation, where only central dot annotations are needed. The proposed fully convolutional regression networks are first trained on a synthetic dataset of cell nuclei, later fine-tuned and shown to generalize to real data.

In 3D fetal ultrasound neurosonography, establishing a coordinate system over the fetal brain serves as a precursor for subsequent tasks, e.g. localization of anatomical landmarks, extraction of standard clinical planes for biometric assessment of fetal growth, etc. To align brain volumes into a common reference coordinate system, we decompose the complex transformation into several simple ones, which can be easily tackled with Convolutional Neural Networks. The model is therefore designed to leverage the closely related tasks by sharing low-level features, and the task-specific predictions are then combined to reproduce the transformation matrix as the desired output.

Finally, we address the problem of MR cardiac video analysis, in which we are interested in assisting clinical diagnosis based on the fine-grained segmentation. To facilitate segmentation, we present one end-to-end trainable model that achieves multi-view structure detection, alignment (standardized re-orientation), and fine-grained segmentation simultaneously. This is motivated by the fact that the CNNs in essence is not rotation equivariance or invariance, therefore, adding the pre-alignment into the end-to-end trainable pipeline can effectively decrease the complexity of segmentation for later stages of the model.

Deep Neural Networks in Computer Vision and Biomedical Image Analysis

D.Phil Thesis

Supervised by

Professor Alison Noble
Professor Andrew Zisserman

Weidi Xie
Wolfson College
University of Oxford
Michaelmas 2017

Acknowledgements

Firstly, I am extremely grateful to my wonderful supervisors Professor Alison Noble and Professor Andrew Zisserman for their continuous support, for their patience, motivation, and immense knowledge, for their guidance to explore the unknowns and become a critical thinker. I could not have imagined having better advisors and mentors for my DPhil study.

I thank my fellow labmates and collaborators from both BioMedia Lab and Visual Geometry Group (VGG), for all the stimulating discussions, for the days we were working together, and for all the fun we have had in the past three years.

I want to thank my funding body, China Oxford Scholarship Funding (COSF) and Google DeepMind, without them, it would not be possible for me to conduct researches without economical concerns.

Last but not the least, I would also like to thank my family for supporting me spiritually throughout my D.Phil life in general.

Declaration

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research. – Weidi Xie

Contents

List of Abbreviations

SVMs . . . . . Support Vector Machines.

SGDs . . . . . Stochastic Gradient Descents.

ANNs . . . . . Artificial Neural Networks.

DNNs . . . . . Deep Neural Networks.

MLPs . . . . . Multi-Layer Perceptrons.

CNNs . . . . . Convolutional Neural Networks.

ResNets . . . . Residual Neural Networks.

FCNs . . . . . Fully Convolutional Neural Networks.

FCRNs . . . . Fully Convolutional Regression Networks.

STNs . . . . . Spatial Transformer Networks.

RNNs . . . . . Recurrent Neural Networks.

LSTM . . . . . Long-Short Term Memory.

GRUs . . . . . Gated Recurrent Units.

L-RNNs . . . . Layer Recurrent Neural Networks.

DCNs . . . . . Deep Comparator Networks.

MR . . . . . Magnetic Resonance.

SSFP . . . . . Steady State Free Precession.

1 Introduction

1.1 Objective

In this thesis, we propose various machine learning models for a variety of applications, ranging from computer vision to biomedical image analysis. By that we aim to model expert knowledge in a mathematical way, and to incorporate it into the deep neural networks (DNNs) architectures as a differentiable layer, or to break the complex tasks into several small, easily-solvable tasks.

The resurgence of DNNs can be traced back to the year of 2012, when the AlexNets [1] was proposed for a winning entry to the ImageNet Challenge [2], and beat the second entry by a large margin (over 10% on top-5 error rate). Since then, researchers from the machine learning and computer vision communities have continued developing new architectures and training techniques to improve the power of deep networks.

Theoretically, DNNs have been proven to be a general function approximator [3], indeed, given enough training data, the deep Convolutional Neural Networks (CNNs) have shown tremendous success in lots of computer vision applications, for instance, large-scale image classification (ImageNet [2]), and semantic segmentation (Cityscapes [4], COCO [5], Pascal VOC [6]). However, in other scenarios, such as biomedical image analysis, it is usually a laborious and expensive process to collect

the data and annotation. In this thesis, we descibe several different methods to build models that reflect human expert knowledge and intuition, hoping this can effectively reduce the searching space of model parameters during training.

1.2 Motivation and Application

Chapter 3 - Semantic Segmentation In computer vision, semantic segmentation is considered as an important step towards image understanding. The task is to assign labels to each individual pixel, describing their categories (Figure 1.1). It acts as a fundamental role in many exciting applications, such as enabling autonomous vehicles to navigate the streets and watch out for pedestrians, diagnosing the heart diseases, or improving automated image annotation systems.

In semantic segmentation, multi-scale contextual information plays an important role. Conventionally, this is usually achieved with a large spatial footprint by the composition of filters through the convolutional and pooling layers. We address this problem by introducing a new block that is able to learn spatial contextual information by within-layer recurrence. During training, the block is able to learn an effective receptive field adaptively, reaching across the entire feature map or image, if that is required for the task. Moreover, given the annotation efforts for semantic segmentation, the new block can be initialized as an identity layer and inserted into any pre-trained CNNs exactly, and taking benefit from the "pre-train and fine-tuning" paradigm.

Chapter 4 - Face Recognition Human face recognition is undoubtedly one of the most widely studied area in computer vision. It finds major applications in face retrieval, surveillance, etc. According to a report by Tencent YouTu Lab (Chinese Tech Company), their face recognition system has helped nearly 124 families to find missing members since it was launched. Despite the tremendous success of current algorithms, it remains challenging for senarios where variations in facial attributes exist, such as age, pose, illumination, accessories (glasses), expressions. Generally,

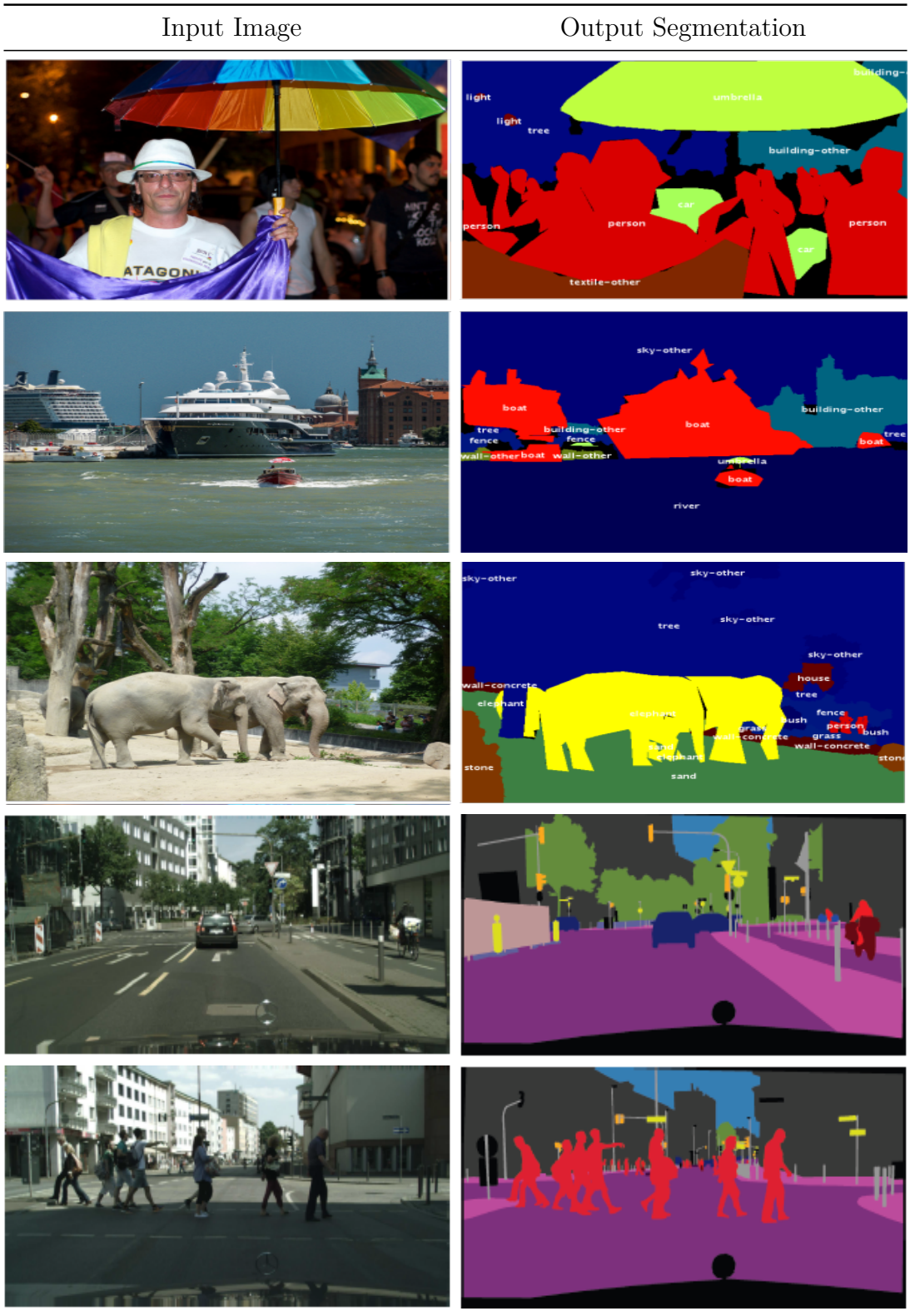


Figure 1.1: Semantic segmentation. The goal is to assign labels to each pixel.

face recognition is usually categorized into two different sub-tasks, namely face identification, face verification.

- The goal of face identification is to assign an identity to the faces in the images or video streams (Figure 1.2).
- Face verification is posed as an open-set problem, which means that the training and testing sets do not have any common identity. In this case, images or videos are always presented in pairs during testing, where each can be composed of a variable number of images or frames. The task is therefore to verify if they belong to the same or different persons (Figure 1.3).

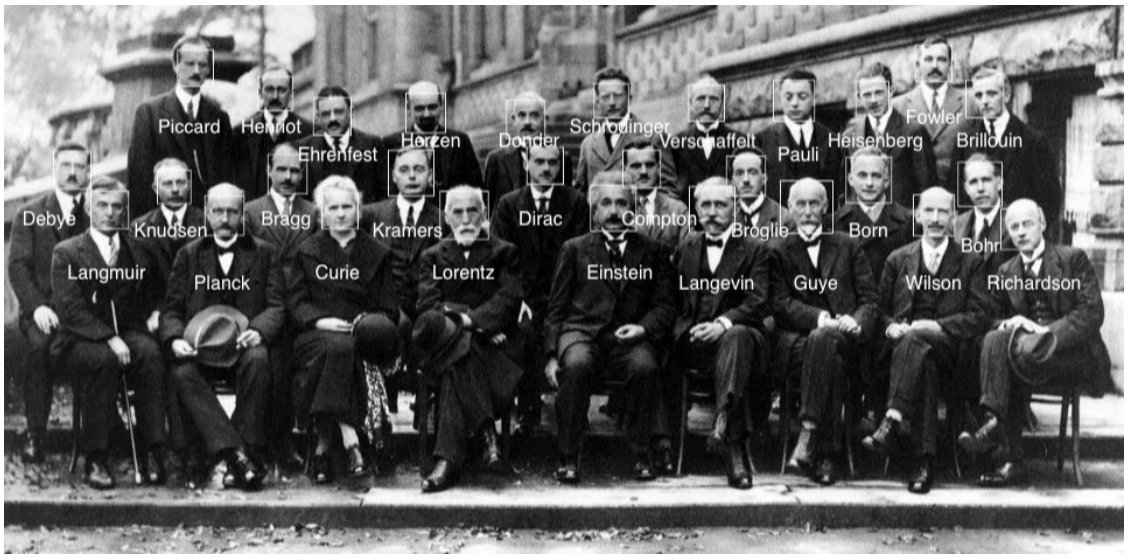


Figure 1.2: Face identification.
The goal here is to label faces with the corresponding identity labels.

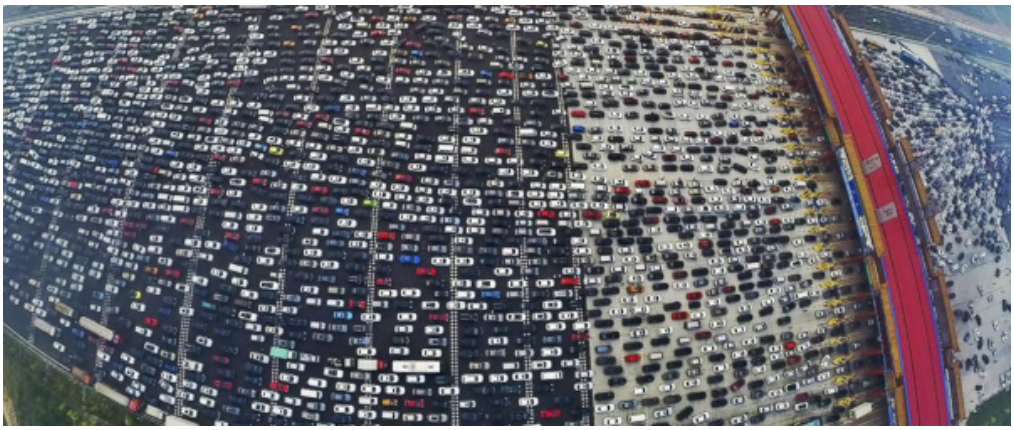
Driven by the recent success on image classification, deep networks have also been trained on large-scale face datasets. While deploying to real scenarios, feature representations for images are computed and directly compared with certain similarity metric, e.g. cosine similarity. In this thesis, we first explore in-plane face alignments by using only identity labels. Further, the face recognition is treated as a fine-grained problem, and addressed with a multi-column deep comparator network. The architecture was designed to take two sets (each containing an arbitrary number of faces) of images or frames as inputs. Feature descriptors for

Figure 1.3: Face verification. Given a pair of images or videos, where each set contains variable number of images, e.g. {(a1)} vs {(a2)} or {(e1)} vs {(e2),(e3),(e4)}, the goal here is to confirm whether the sets belong to same identity of person.

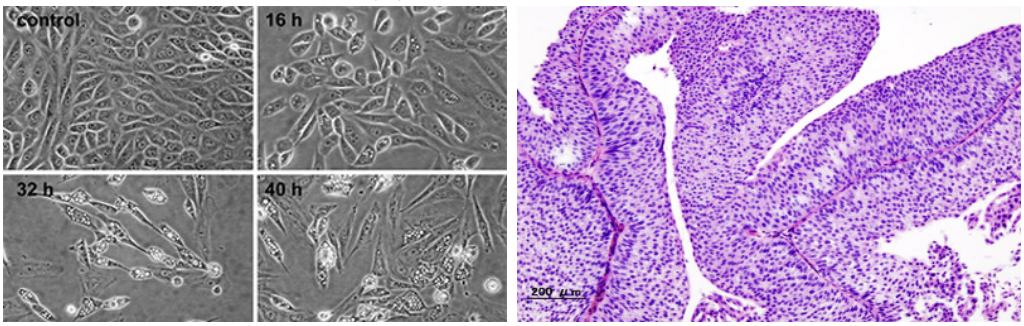
different facial parts are pooled out by attending to the corresponding spatial locations, and further compared with local "experts" in a pairwise way to predict if the two sets belong to same the person or not.

Chapter 5 - Objects Counting and Detection Counting objects in crowded images or videos is an extremely tedious and time-consuming task encountered in many real-world applications (Figure 1.4), e.g. traffic monitoring, cancer research. In this thesis, we approach microscopy cell counting and detection based on density estimation. Unlike computer vision applications, where data is usually adequate, it

(a) Human Crowds



(b) Traffic Monitoring



(c) Biology Cells

Figure 1.4: Counting applications. The goal is to monitor the objects number change.

is required to train models with very few data and weak annotation in biomedical image analysis. Therefore, the proposed fully convolutional regression networks are trained on a synthetic dataset of cell nuclei, where only central dot annotations are provided during supervised training. We show state-of-the-art performance on

the synthetic dataset, and later, the model is re-calibrated (fine-tuned) and has shown strong generalizations on real microscopy images.

Chapter 6 - Standardized Re-orientation of 3D Ultrasound Fetal Brain In fetal ultrasound neurosonography, due to the variability in the location of the fetal brain relative to the probe, establishing a common coordinate system for the fetal brain always serves as a precursor for subsequent tasks, e.g. localization of anatomical structures, extraction of standard clinical planes for biometric assessment of fetal growth, etc. To align brain volumes into a common reference coordinate system, the transformation is usually highly abstract from the data itself (Figure 1.5). Therefore, we address this problem by decomposing the complex transformation into several simple ones, which can be easily tackled with Convolutional Neural Networks, e.g. skull segmentation, facing inferior or superior, posterior or anterior, etc. The model is designed to leverage the closely related tasks by sharing low-level features, and the task-specific predictions are then combined to reproduce the transformation matrix as the desired output.

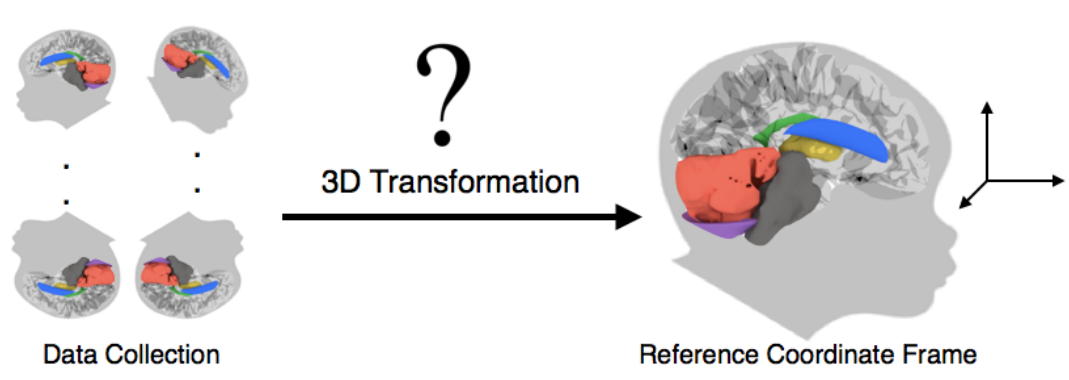


Figure 1.5: Standardized re-orientation of 3D ultrasound fetal brain. The goal is to transform the collected 3D ultrasound fetal brain to a common coordinate.

Chapter 7 - Magnetic Resonance (MR) Cardiac Video Segmentation In MR cardiac image analysis, pixel-wise segmentation of the ventricular myocardium and the four cardiac chambers is an essential preprocessing step for volume estimation. However, the automatic cardiac segmentation remains a notoriously

difficult problem (Figure 1.6). To address this, we present an end-to-end trainable model that achieves multiview structure detection, alignment (standardized re-orientation), and fine-grained segmentation simultaneously. This is motivated by two facts: first, CNNs is not well-understood by the community, by decomposing the complicated task into several tractable sub-tasks, we are able to visualize the intermediate steps and therefore act as a step towards more explainabe models. second, CNNs in essence, is not rotation equivariance or invariance. Therefore, adding the pre-alignment to the end-to-end trainable pipeline can effectively decrease the complexity of segmentation for later stages of the model.

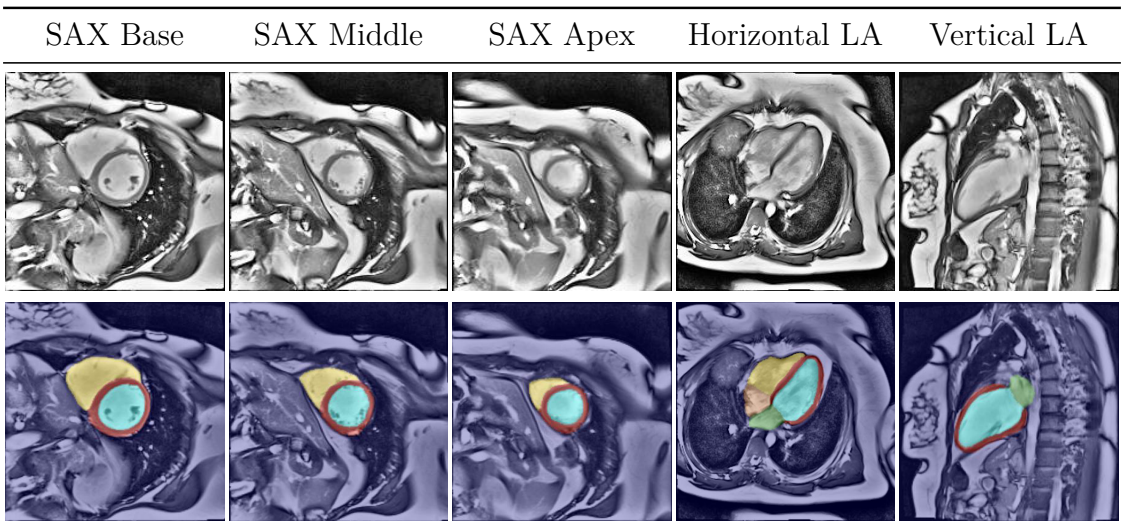


Figure 1.6: Magnetic Resonance (MR) cardiac video segmentation (For viewing convinience, images have been preprocessed by histogram equalization).
Notation: SAX: Short Axis, LA: Long Axis

1.3 Publication

The subsequent chapters in this thesis will describe the works that have been published (or submitted to) on the following conferences and journals [7–18]:

1.3.1 Publications Related to This Thesis.

Chapter 3 Weidi Xie, J. Alison Noble and Andrew Zisserman, "Layer Recurrent Neural Networks", (Technique Report, 2016). `https://openreview.net/forum?id=rJJRDvcex`

Chapter 4 Weidi Xie, Li Shen and Andrew Zisserman, "Comparator Networks", In: *European Conference on Computer Vision (ECCV)*, 2018.

Chapter 5 Weidi Xie, J. Alison Noble and Andrew Zisserman, "Microscopy Cell Counting with Fully Convolutional Regression Networks", In: *MICCAI Workshop on Deep Learning in Medical Image Analysis*, 2015.

Chapter 5 Weidi Xie, J. Alison Noble and Andrew Zisserman, "Microscopy Cell Counting and Detection with Fully Convolutional Regression Networks", In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1-10, 2016.

Chapter 6 Weidi Xie[1], Ana Namburete[1], Mohammad Yaqub, Andrew Zisserman and J. Alison Noble, "Fully-Automated Standardized Reorientation of 3D Fetal Neurosonography Images using Multi-Task FCNs", In: *Medical Image Analysis*, Volume 46, May 2018, Pages 1-14.

Chapter 7 Weidi Xie[1], Davis M. Vigneault[1], Carolyn Ho, David A. Bluemke and J. Alison Noble, "Ω-Net: Fully Automatic, Multi-View Cardiac MR Detection, Orientation Alignment, and Segmentation with Deep Neural Networks", In: *Medical Image Analysis*, Volume 48, August 2018, Pages 95-106.

1.3.2 Publications Not Covered in This Thesis.

[1] **Weidi Xie** and Andrew Zisserman, "Multicolumn Networks on Face Recognition". In: *British Machine Vision Conference (BMVC)*, 2018.

[2] Qiong Cao, Li Shen, **Weidi Xie**, Omkar M. Parkhi and Andrew Zisserman, "VGGFace2: A dataset for recognising faces across pose and age", In: *IEEE Conference on Automatic Face and Gesture Recognition (F&G)*, 2018.

[3] Ruobing Huang, **Weidi Xie** and J. Alison Noble, "VP-Nets : Efficient Automatic Localization of Key Brain Structures in 3D Fetal Neurosonography", In: *Medical Image Analysis*, Volume 47, July 2018, Pages 127–139.

[1]Equal Contribution, joint first author

[4] Mohammad Ali Maraci, **Weidi Xie**, and J. Alison Noble, "Can Dilated Convolutions Capture Ultrasound Video Dynamics?". In: *9th International Conference on Machine Learning in Medical Imaging (MLMI), 2018.*

[5] Ana Namburete, **Weidi Xie** and J. Alison Noble, "Robust Regression of Brain Maturation from 3D Fetal Neurosonography using CRNs", In: *MICCAI Workshop on Fetal and InFant Image analysis* (FIFI 2017), Best Paper Award.

[6] Davis M. Vigneault, **Weidi Xie**, David A. Bluemke and J. Alison Noble, "Feature Tracking Cardiac Magnetic Resonance via Deep Learning and Spline Optimization", In: *Functional Imaging and Modelling of the Heart* (FIMH 2017), Best Poster Award.

2
Literature Review

In this chapter, we review the relevant literature on deep neural networks, also known as Deep Learning. Throughout the thesis, we will only focus on models trained with supervised learning (§2.1), which are optimized with first-order gradients (§2.2).

As for Artificial Neural Networks (ANNs) architectures, we start with the fundamental Multi-Layer Perceptrons (MLPs) in §2.3. Convolutional Neural Networks (CNNs), which are more suitable for dealing with vision problems are described in §2.4. In §2.5, we provide an overview of the vanilla Recurrent Neural Networks (RNNs), and other popular variants, e.g. Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs).

2.1 Supervised Learning

Recent successes in computer vision and machine learning have been primarily driven by supervised learning, which aims to fit a function mapping from the input space X to the output space Y. In general, we assume that $(x, y) \sim D \in X \times Y$ represents the data pair sampled from the set D, where x refers to the input object (typically a vector calculated from images or audio signal), and y is the desired (possibly noisy) output (also called the supervisory signal). The goal of supervised learning is therefore to find the weights θ of a parametrized function $y = g(x; \theta)$ that achieves

minimum prediction error on the samples of an unseen test set D_{test}. This is fundamentally impossible to achieve directly. Instead, we approximate the test error by errors on a training set D_{train}, based on the assumption that the samples in the training set follow a similar distribution as the test set.

In practise, given a training set of N independently and identically distributed (*i.i.d.*) sample points, a loss (cost) $L(y_i, \hat{y}_i)$ is defined to measure how well a function $g(\cdot, \theta)$ fits the training data, where y_i and $\hat{y}_i$ refer to the groundtruth and prediction respectively. The empirical risk $R(g)$ of function g is defined as the expected loss of g over the entire dataset:

$$R_{emp}(g) = \frac{1}{N} \sum_i L(y_i, g(x_i; \theta)) \tag{2.1}$$

and the objective is to minimize the empirical risk:

$$J(\theta; D) = R(g) + \alpha\Omega(\theta) \tag{2.2}$$

where $\Omega(\theta)$ refers to the regularizer that penalizes certain values of θ and α is the regularization parameter. With each set of weights ($\theta's$) being randomly initialized, minimizing the expected loss therefore becomes an optimization problem.

2.2 Gradient-based Optimization

Once the objective function is defined, it can be effectively optimized with a gradient-based approach by back-propagation. In the deep learning literature, first-order gradient descent [19] is the most fundamental and effective approach to optimizing the objective function $J(\theta; D)$, where θ, D refer to the model pramaters and data respectively. Following $-\nabla_\theta J(\theta; D)$ direction, the parameters are updated by:

$$\theta := \theta - \alpha \nabla_\theta J(\theta; D) \tag{2.3}$$

where α is called the learning rate (Figure 2.1).

Empirical, although vanilla gradient descent can always lead to good convergence, it also suffers from a few challenges:

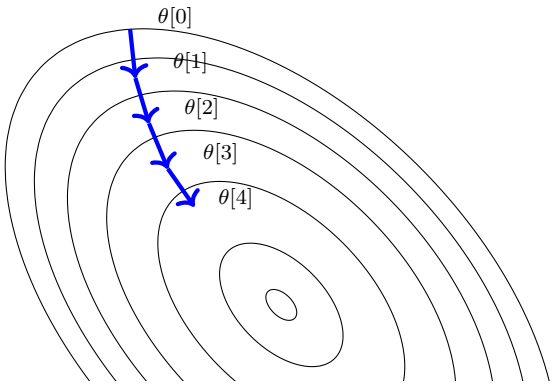


Figure 2.1: Gradient descent optimization for a quadratic function. Gradient descent falls into the first-order optimization, which aims to update the model parameters by the negative gradient, as it describes the direction of the steepest descent. The learning rate α describes the step size taken in each iteration step.

- The calculation of gradient needs to consider the entire dataset, making it very time-consuming and memory inefficient.
- Selecting a proper learning rate can be difficult, small learning rate leads to very slow convergence, while a large learning rate may lead to a gradient explosion and divergence.
- Updating each parameter with the same learning rate is often not ideal.
- For naïve gradient descent, the local optima and saddle points are often difficult to escape.

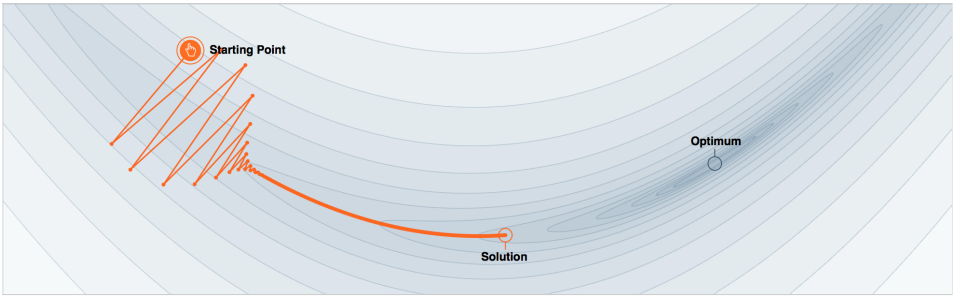
In practise, mini-batch stochastic gradient descent (SGD) with momentum is widely used to accelerate the convergence.

Momentum Momentum [20] is a method that helps to accelerate SGD in the relevant direction and dampens the oscillations. It does this by adding a fraction γ of the update vector from the past time step to the current update vector, therefore builds up the speed in directions with a gentle but consistent gradient.

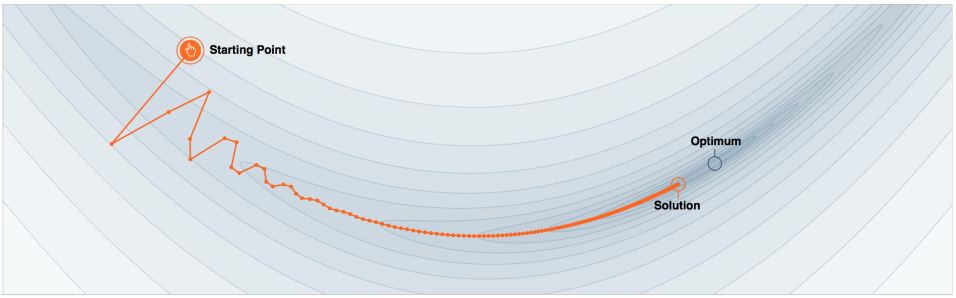
$$\begin{aligned} v_t &= \gamma v_{t-1} + \epsilon \nabla_\theta J(\theta) \\ \theta &:= \theta - v_t \end{aligned} \tag{2.4}$$

As shown in Figure 2.2, by considering gradients from the previous step, momentum can effectively reduce updates for dimensions whose gradients suddenly change directions, and increase updates for dimensions whose gradients point to the same directions (Figure 2.2). More recently, to overcome the above challenges,

even more advanced optimization techniques have also been proposed, such as Adagrad [21], Adadelta [22], RMSProp [23], Adam [24].



(a) SGD without Momentum



(b) SGD with Momentum

Figure 2.2: Momentum in stochastic gradient descent. (Figures are from [25])

2.3 Multi-Layer Perceptrons

Multi-Layer Perceptrons (MLPs) are among the most fundamental building blocks in Artificial Neural Networks (ANNs). It refers to a set of computational models that are loosely inspired by the human brain. In general, they consist of two important elements, namely, artificial neurons (nodes) and synapses (weights) that connect them (Figure 2.3) .

Inspired by the natural world, multi-layer MLPs are designed to learn feature representations in an hierarchical manner, making them so powerful that in theory they can fit an arbitrary function to arbitrary accuracy [3]. One way of thinking about feedforward neural networks is as a set of nonlinear transformations, where hidden layers are stacked on top of previous outputs. The hidden layer output h can be computed as:

$$h = f(\mathbf{W} \cdot x) \tag{2.5}$$

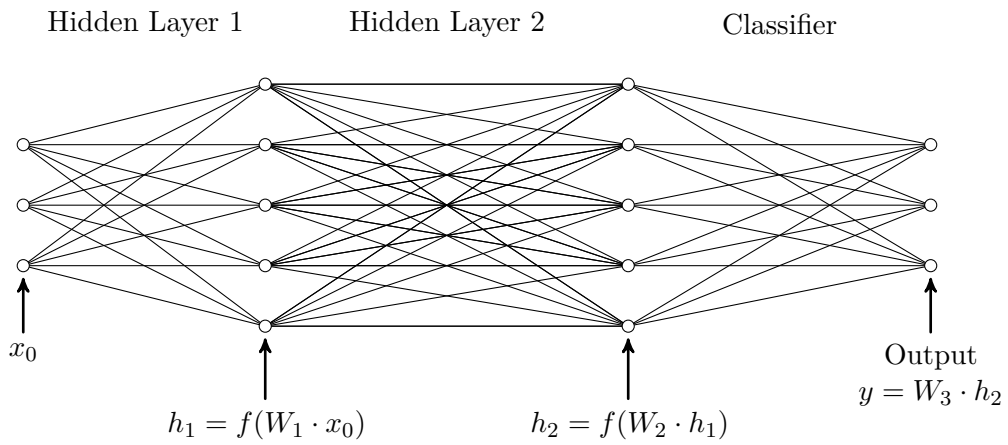


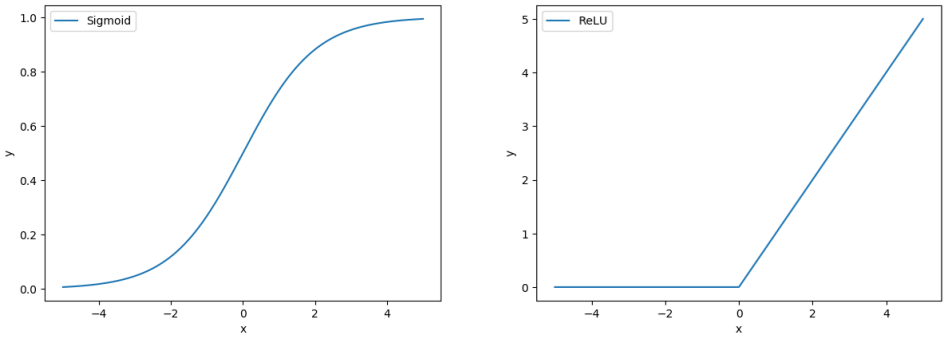
Figure 2.3: Simple Multi-Layer Perceptrons. Each small circle corresponds to a neuron, taking a number of inputs and producing a single output by modulating the weighted sum with a nonlinear function. Connections between the neurons thus form the network, and can take any arbitrary asyclic topology.

where x refers to the inputs to the hidden layer, and $f(\cdot)$ is the nonlinear activation function. Traditionally, a sigmoid (Figure 2.4a) is widely used for f:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

More recent researches start using the Rectified Linear Units (Figure 2.4b):

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$



(a) Sigmoid Function **(b)** Rectified Linear Units (ReLU)

Figure 2.4: Activation functions.

2.4 Convolutional Neural Networks (CNNs)

In computer vision, one of the earliest neural networks dates back to "Neocognition" in the 1980s [26]. Neocognition is a hierarchical network consisting of many layers, and variable connections between nodes in adjacent layers. Local features of the input are initially extracted by the lower level nodes, and gradually integrated into more global features. With the pooling layers, the networks can tolerate slight distortion at each stage, and eventually get robust to deformations, scales, and translations in the position of the inputs. After training, the networks have equipped the ability to perform simple pattern recognition.

Later on, inspired by the "Neocognition", CNNs were introduced and successfully trained with back-propagation [27, 28]. With the help of large-scale datasets and high-performance Graphic Processing Units (GPUs), CNNs started taking off in 2012. Figure 2.5, 2.6 shows the recent development of CNNs architectures on the ImageNet image classification benchmark [29].

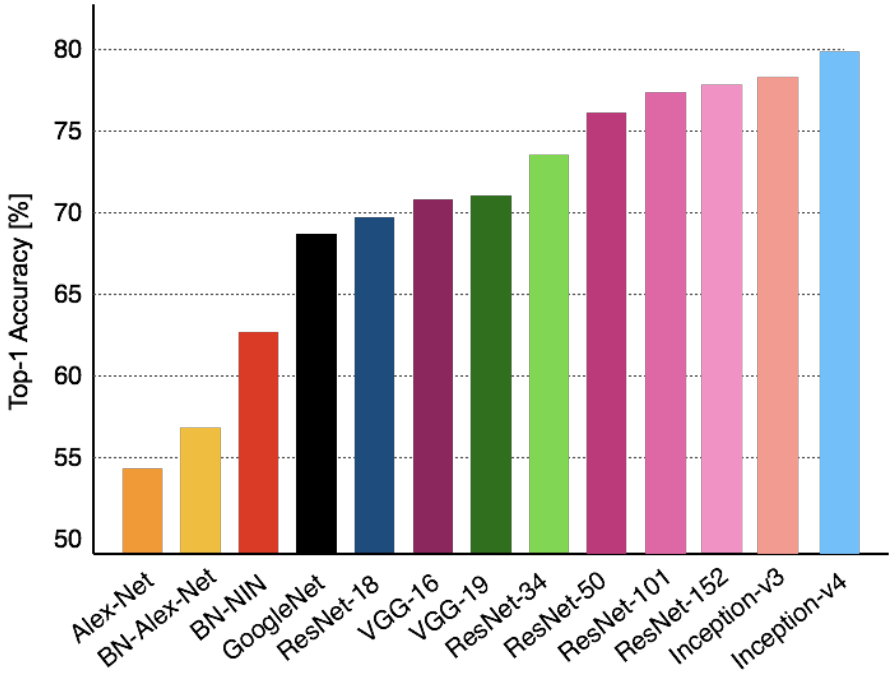


Figure 2.5: Architecture comparisons (top1-acc) on ImageNet image classification benchmark. This figure is drawn based on [29]..

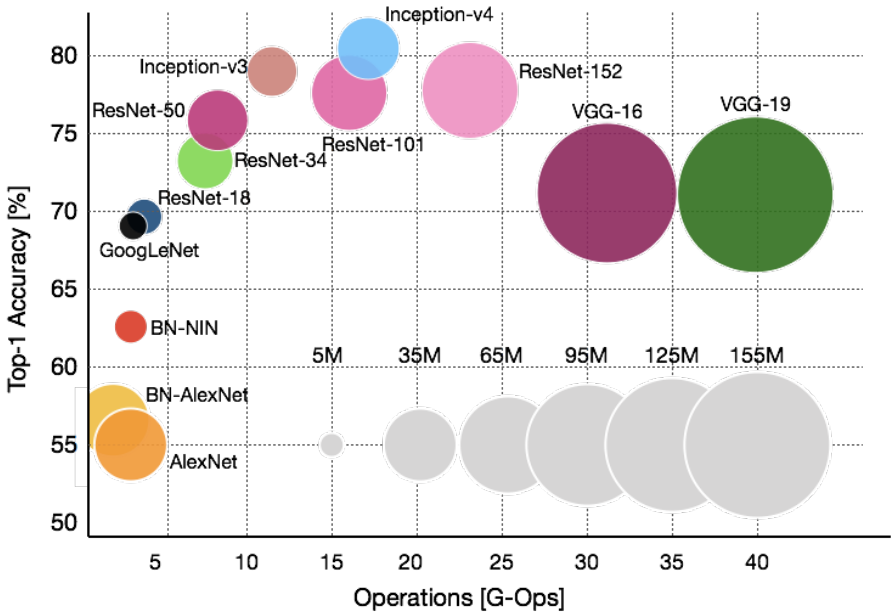


Figure 2.6: Architecture comparisons (operations) on ImageNet image classification benchmark. This figure is drawn based on [29]..

2.4.1 LeNets

LeNets represent the first-generation CNNs, which was initially developed to recognize handwritten digits. Through the architecture, convolutional layers with sigmoid units alter with average pooling layers, where tiny invariance is added and computational burden reduced. At the end of architecture, classifier (fully connected layer) was appended and trained with the classification losses.

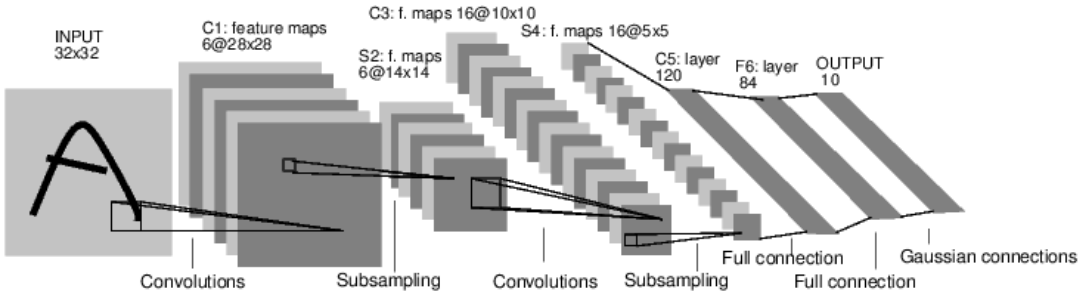


Figure 2.7: LeNets. The figure is taken from the original paper [28].

Although LeNets have achieved the state-of-the-art result on the MNIST dataset, it was found difficult to generalize to real-world vision problems in the 1990s, mainly for two reasons. First, to train the large-scale CNNs, it usually took thousands of

iterations to converge with stochastic gradient descent (SGD), however, there was not enough computation power at that time. Second, even by sharing convolutional kernels spatially, a typical CNNs still contains millions of paramters that leads to overfiting the dataset (MNIST) easily.

2.4.2 AlexNets

The first breakthrough of CNNs in computer vision happened in the year of 2012, AlexNets (Figure 2.8) was successfully trained to win the ImageNet Challenge [2]. With the help of a large amount of labelled data and high-performance GPUs, the large-scale CNNs were brought back to the stage.

More specifically, AlexNets includes 8 convolutional layers with rectified linear units (ReLUs), max-poolings are used to downsample the feature maps and gradually add tiny invariance. To avoid over-fitting, dropout [30] and data augmentation are applied during training, e.g. color jitterings, random croppings, rotations.

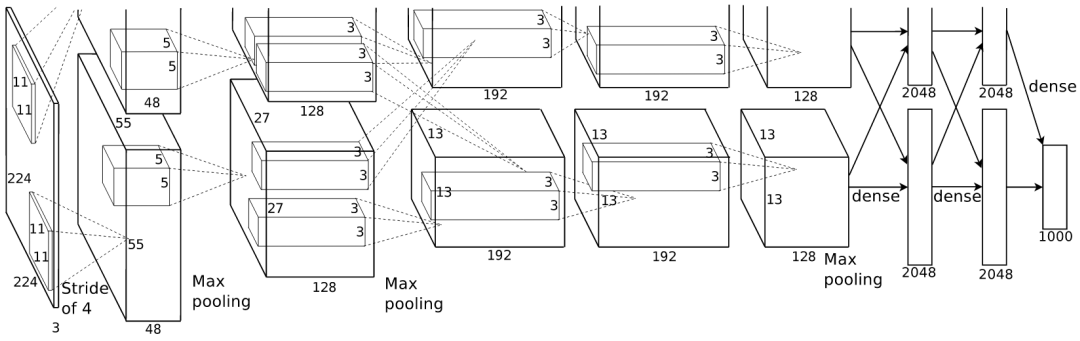


Figure 2.8: AlexNet. The figure is taken from the original paper [1].

2.4.3 VGGNets & GoogLeNets

Since then, a number of works started to further advance image classification performance with CNNs. In 2014, both GoogleNets [31] and VGGNets [32] achieved significant performance boost by using deeper architectures. Specifically, the VGGNets (Figure 2.9) is constructed with 19 convolutional layers of 3×3 kernels, maxpooling layers are used after every two or three convolutional layers, at the end,

two fully connected layers (4096 nodes) are attached to summarize the global information.

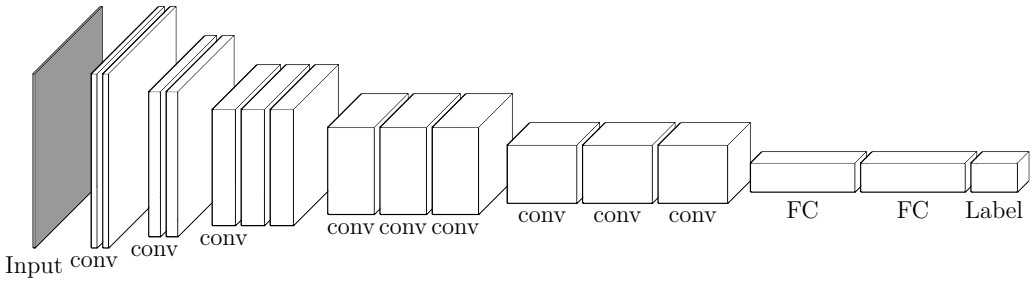


Figure 2.9: VGG-16 Network [32].
In the network, all filter kernels have size of 3x3, max poolings are placed after each 2 or 3 convolutions and the number of filters are doubled after each max-pooling.

Similar to the idea of Network in Network(NIN) [33], GoogLeNets (Figure 2.10) propose to use the Inception Blocks, where several 3×3 and 1×1 kernels are used within one module. Similar to [34], auxiliary supervisions are applied on several intermediate layers to overcome the gradient vanishing issue.

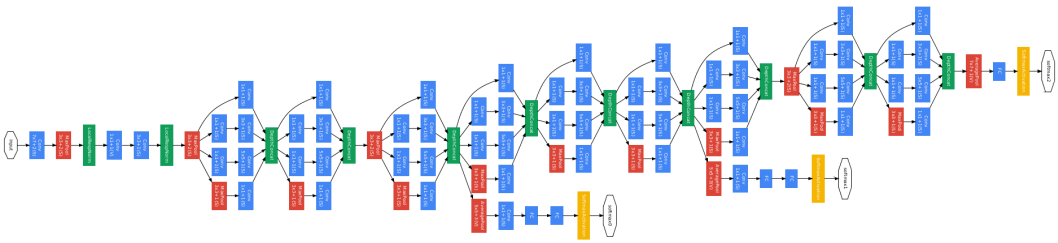


Figure 2.10: GoogLeNet. The figure is taken from the original paper [31].
The network uses combinations of inception modules, each including some pooling, convolutions at different scales and concatenation operations. It also uses 1x1 feature convolutions that work like feature selectors.

2.4.4 ResNets & Highway Networks

Although there seems to be a strong and positive correlation between network depth and image classification performance, researchers found that it was not straightforward to train deeper networks without harming the performance. Until recently,

Residual Networks [35](ResNets) with over hundreds of layers are successfully trained with skip connections, each "Residual Unit" can be expresesed in a general form:

$$\begin{aligned} y_l &= h(x_l) + F(x_l, W_l) \\ x_{l+1} &= f(y_l) \end{aligned} \tag{2.8}$$

where x_l and x_{l+1} are the input and output of the lth unit respectively, and F is a residual function. It is generally conjectured that gradients are more easily back-propagated with the help of residual units. Almost concurrent to ResNets, Highway Networks [36] was proposed to train the very deep CNNs with gating mechanisms.

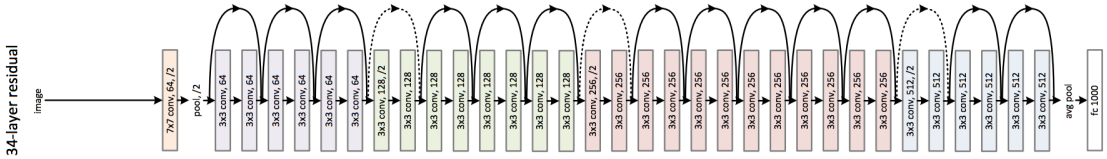


Figure 2.11: ResNets. The figure is taken from the original paper [35].
Residual units are used extensively in the networks, at the end, average pooling is used to aggregate global feature representations.

2.4.5 Fully Convolutional Networks (FCNs)

Beyond the general image-level classifications, pixel-level predictions natually pose more challenges on algorithm accuracy, efficiency, and robustness.

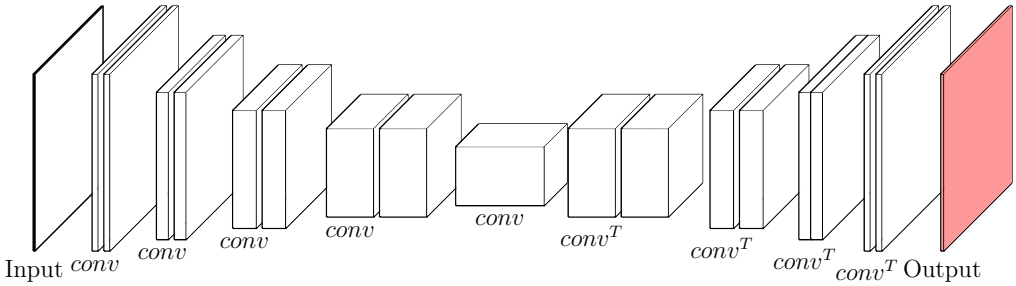


Figure 2.12: Fully Convolutional Networks (FCNs).
The fully convolutional networks (FCNs) consists of a down-sampling path (left), followed by an up-sampling (right) path to restore the original spatial resolution.

To overcome these challenges, modern CNNs have been optimized by replacing fully connected layers with convolutions of 1×1 kernel. The entire architecture (Figure 2.12) consists of two parts. The downsampling path resembles the canonical

classification CNNs, while in the upsampling path, the spatial resolution is restored by training convolution "transpose" kernels. Therefore, given the fully annotated pixel-level labels, image-to-image translation can be trained in an end-to-end way, for instance, to achieve semantic segmentation [37], image colorization [38], and image inpainting [39].

2.4.6 Spatial Transformer Networks (STNs)

Unlike the previous networks, where invariances, e.g. scale, rotation, are learnt from aggresive data augmentations, Jaderberg *et al.* [40] proposes a module that can *explicitly* learn to transform the image or feature maps for the benefit of subsequent tasks, e.g. image classification.

A complete STNs is composed of three parts, namely a localization network, a grid generator, and a sampler. The localization network predicts transformation parameters that transforms the original grid to a new one, and the sampler then accepts this transformed grid to re-sample the input image or feature maps.

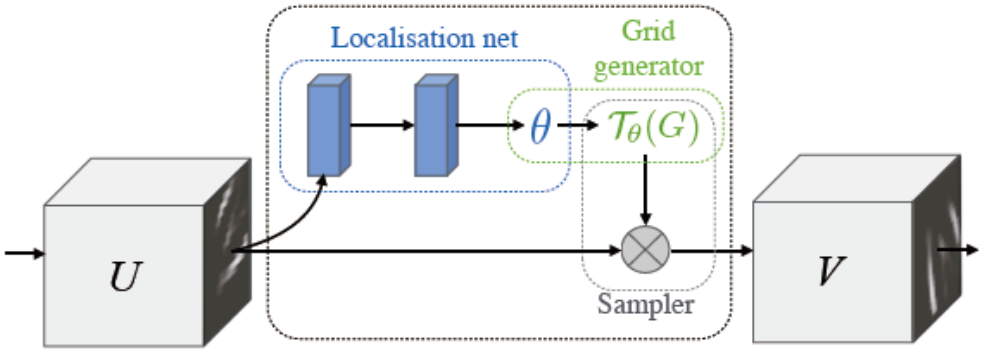


Figure 2.13: Spatial Transformer Networks (STNs). The figure is adapted from the original paper [40]. The entire module consists of three sub-modules, namely localization net, grid generator and sampler. It can be inserted into a network at any level to explicitly transform the feature maps.

Localization Network (LocNet) In a standard STN module, the localization network takes images or feature maps as input, and outputs a complete transformation matrix.

$$T = f(X; W) \tag{2.9}$$

where T refers to the predicted transformation matrix, and $f(\cdot, W)$ is a parameterized function, usually represented as a CNN.

Grid Generation & Sampling Generically, a 2-D "grid generator" takes a (typically uniform) sampling of points $G \in \mathbb{R}^{2\times H'\times W'}$ and transforms them according to the parameters predicted by a LocNet. The grids are usually spaced over the extent of the image ($x \in [-1, 1]$, $y \in [-1, 1]$); these grids are then transformed by the matrix T (predicted by the LocNet) to determine which points to sample from the input image.

The "sampler" takes the input image or feature maps $X \in \mathbb{R}^{H\times W\times C}$ and the transformed grid G' as arguments, and produces a resampled image $X' \in \mathbb{R}^{H'\times W'\times C}$. For each channel $c \in [1 \,..\, C]$, the output $I'_{h',w',c}$ at the location (h', w') is a weighted sum of the input values $I_{h,w,c}$ in the neighborhood of location $(G'_{1,h',y'}, G'_{2,h',w'})$,

$$\begin{aligned} I'_{h',w',c} = \sum_{h=1}^{H}\sum_{w=1}^{W} I_{h,w,c} \\ \cdot \max\left(0, 1 - |\alpha_v G'_{1,h',w'} + \beta_v - h|\right) \\ \cdot \max\left(0, 1 - |\alpha_u G'_{2,h',w'} + \beta_u - w|\right), \end{aligned}$$

where

$$\begin{aligned} \alpha_v &= +\frac{H-1}{2}, \\ \beta_v &= -\frac{H+1}{2}, \\ \alpha_u &= +\frac{W-1}{2}, \text{and} \\ \beta_u &= -\frac{W+1}{2}. \end{aligned}$$

Given every step here is differentiable (either a gradient or sub-gradient is defined), the module can be inserted into any standard CNN architectures and trained end-to-end.

2.5 Recurrent Neural Networks (RNNs)

Alghough CNNs have shown tremendous success in solving vision-related problems, they are mainly designed for processing images, where each sample is treated independently. However, humans do not start their thinking from scratch every second, our brain tends to encode and memorize the important events. For example, imagine we want to classify what kind of event is happening at every point in a movie. Humans have the ability to reason about the previous events in a movie and infer the later frames. To overcome the limitation from memoryless models, Recurrent Neural Networks (RNNs) is proposed.

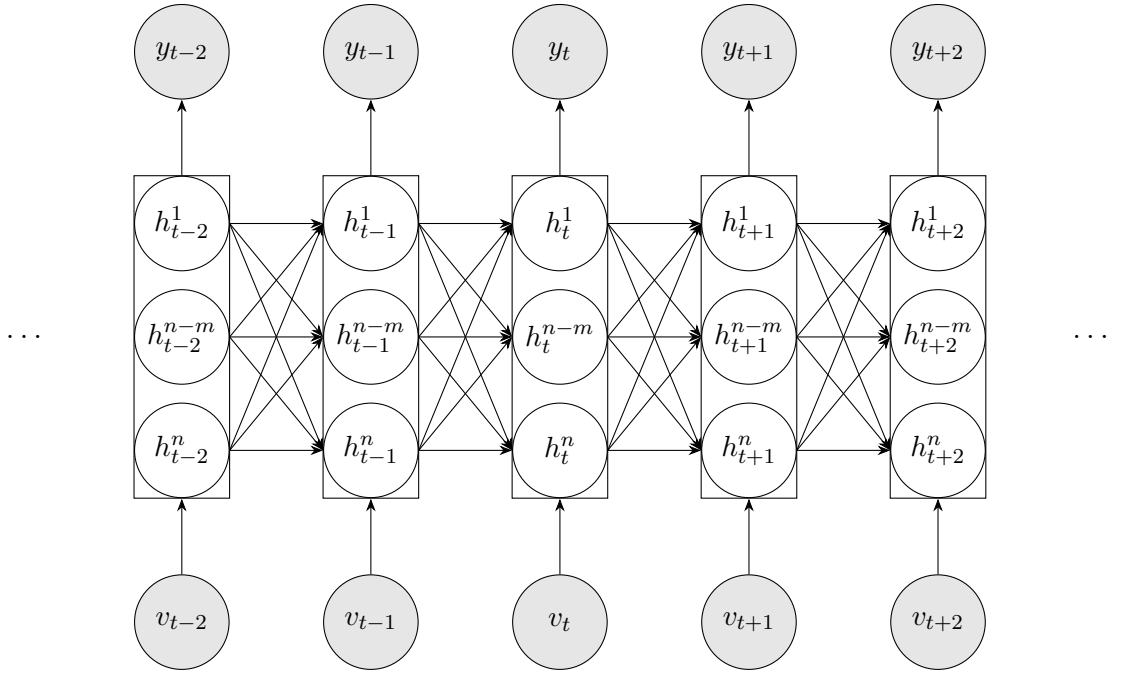


Figure 2.14: Vanilla Recurrent Neural Networks (RNNs).
v_t, h_t, y_t refer to the inputs, hidden units and outputs respectively.

RNNs are a superset of feedforward neural networks, augmented with the ability to pass information across time steps. By sharing parameters in the temporal dimension, RNNs are able to process temporal data of variable length. A vanilla RNNs (Figure 2.14) can be expressed as :

$$\begin{aligned} h_t &= f(Uv_t + Vh_{t-1}) \\ y_t &= g(Wh_t) \end{aligned} \tag{2.10}$$

where v_t, h_t refer to input and hidden state at time step t, f is the nonlinear function and g as an arbitrary differentiable classifier.

The relationships between CNNs and RNNs are similar to FIR and IIR filters in digital signal processing. Besides considering local correlations like CNNs, feedback (recursive) units are also incorporated into RNNs, adding temporal correlations into consideration.

2.5.1 Long Short Term Memory (LSTM)

Over the years of research on RNNs, it is found that training vanilla RNNs always suffers from gradient vanishing as the sequences get long. To tackle this problem, Long Short Term Memory [41] was proposed to explicitly guarantee valid gradients during back-propagation (Figure 2.15).

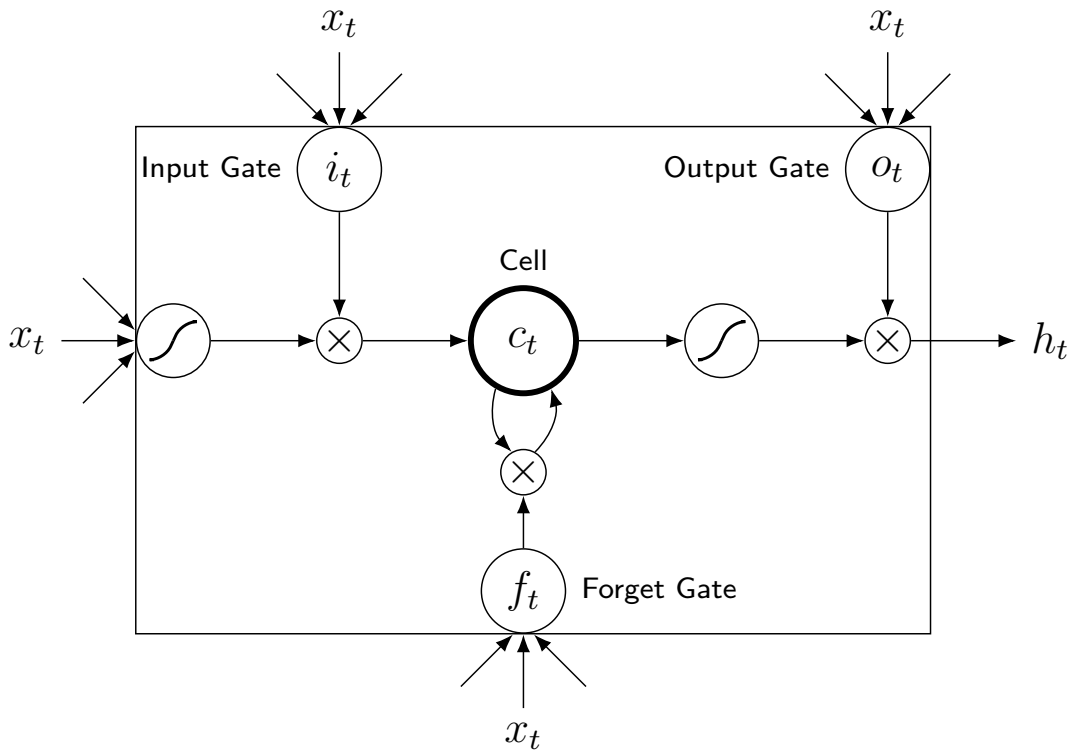


Figure 2.15: Long-Short Term Memory (LSTMs) Unit [41].

The computation for hidden states can be expressed as :

$$\begin{aligned} i &= \sigma(x_t U^i + h_{t-1} W^i) \\ f &= \sigma(x_t U^f + h_{t-1} W^f) \\ o &= \sigma(x_t U^o + h_{t-1} W^o) \\ g &= tanh(x_t U^g + h_{t-1} W^g) \\ c_t &= c_{t-1} \circ f + g \circ i \\ h_t &= tanh(c_t) \circ o \end{aligned} \tag{2.11}$$

where i, f, o refer to the input, forget and output gate respectively, and U^i, U^f, U^o, U^g, W^i, W^f, W^o, W^g are trainable parameters.

The key to LSTM is the cell state (c_t) that is represented as a linear combination of the current inputs and previous cell states (c_{t-1}), making it possible for information to just flow along it unchanged. In addition to the normal computation of hidden states (g) as traditional RNNs, the current inputs (x_t) and previous hidden states (h_{t-1}) are also passed to several gates (i, f, o). The gates are used to optionally control how much information can go through, with sigmoid non-linearity. The output from each dimension of the gates is a number between 0 and 1. A 1 represents "completely keep this" while a 0 represents "completely get rid of this".

2.5.2 Gated Recurrent Units (GRUs)

To simplify the computations in the original LSTM, Gated Recurrent Units (GRUs) [42] was proposed with similar gating ideas. A GRU can be expressed as :

$$\begin{aligned} z &= \sigma(x_t U^z + h_{t-1} W^z) \\ r &= \sigma(x_t U^r + h_{t-1} W^r) \\ i &= tanh(x_t U^i + (h_{t-1} \circ r) W^i) \\ h_t &= (1 - z) \circ i + z \circ h_{t-1} \end{aligned} \tag{2.12}$$

where r, z refer to reset and update gate. Intuitively, the reset gate determines how to combine the current input with previous hidden states, and the update gate defines how much of the previous memory to keep. If we set the reset to all 1's and update gate to all 0's, we are again back to the original vanilla RNNs.

3
Layer Recurrent Neural Networks

3.1 Introduction

In the computer vision tasks, multi-scale contextual information plays a very important role in achieving high performance. The original architectures for these tasks (e.g. AlexNets [1], GoogLeNets [31], VGGNets [32], ResNets [35], FCNs [43], UNets [44]) are able to obtain the multi-scale contexts with a large spatial footprint by the composition of filters through the layers of the network, so that a large receptive field is effectively built up. In fact, the final layers of these networks use global average pooling or fully connected layers, so that the receptive field covers the entire input image patch. More recent architectures for pixel-wise prediction have used dilated convolutions [45, 46], which are able to aggregate multi-scale contextual information without losing resolution (due to the spatial pooling and strides in the original architectures), and without incurring the penalty of learning many parameters for convolutions with very large kernels. Nevertheless, these networks still rely on the hand-specified kernels limiting the architecture to local contexts.

Taking advantages of the topological structure of Recurrent Neural Networks (§ 2.5 on RNNs), in this chapter, we introduce an alternative 'module' for learning spatial contextual information by using recurrence *within* layers, termed as a Layer Recurrent Neural Network (L-RNN). A L-RNN module is composed of several

1D spatial RNNs that aims to learn contextual information adaptively, with the effective receptive field being able to reach across the entire feature map or image, if that is required for the task. The proposed hybrid networks try to combine the best of both worlds: canonical convolutional layers with filters that are efficient in capturing features in a *local* region, whilst the L-RNNs are able to learn *long-range* dependencies across a layer efficiently with only a small number of parameters. In order to accelerate the training speed, we show that the proposed L-RNN module can be initialized as identity function, making it exactly equivalent to any pre-trained CNNs layers, and long-range dependencies can be further learnt by fine-tuning, if it is necessary.

It is worth noting that (broadly) recurrence can be used in feed-forward CNNs architectures in two ways: *between* layers, and *within* layers. For example, between-layer recurrence was used for scene labelling in [47, 48] with convolutions applied recursively on top of the feature maps from different layers or raw input images. In CRF-RNN [49], spatial dependencies are modelled explicitly for semantic segmentation with densely connected Gaussian CRFs by iterated application of bilateral filtering using the between-layer recurrence.

Our proposed Layer-RNN architecture falls into the second category, where *within-layer* recurrence is used to capture the dependencies. Others have learnt contextual information from within-layer recurrence for tasks such as object detection (Inside-Outside Networks [50]), and the low-level vision problems, such as de-noising, colourization and smoothing [51].

In this chapter, we describe the basic L-RNN module in § 3.2.1. § 3.2.2 explains how the L-RNN modules can be inserted into any pre-trained convolutional layer seamlessly. In § 3.2.3, detailed derivation of back-propagation during fine-tuning is provided, by that we mean, the entire network does not have to be trained from scratch, and the added L-RNNs can be finetuned together with the pre-trained networks. We finetune a truncated VGG-16 FCNs base net for semantic segmentation on the Pascal VOC 2012 dataset, and the experiments show that this addition can *always* boost the performance.

3.2 Method

The proposed architecture (Figure 3.1) is composed of two parts, local features are first calculated by the low-level CNNs module, and the Layer-RNN (L-RNN) module, consisting of several 1D spatial RNNs is further applied to capture the spatial dependencies. By scanning across the feature maps in different directions (left-to-right, right-to-left, top-to-bottom and bottm-to-top), the complete L-RNN is able to learn the receptive field in an adaptive way, up to the size of the entire image.

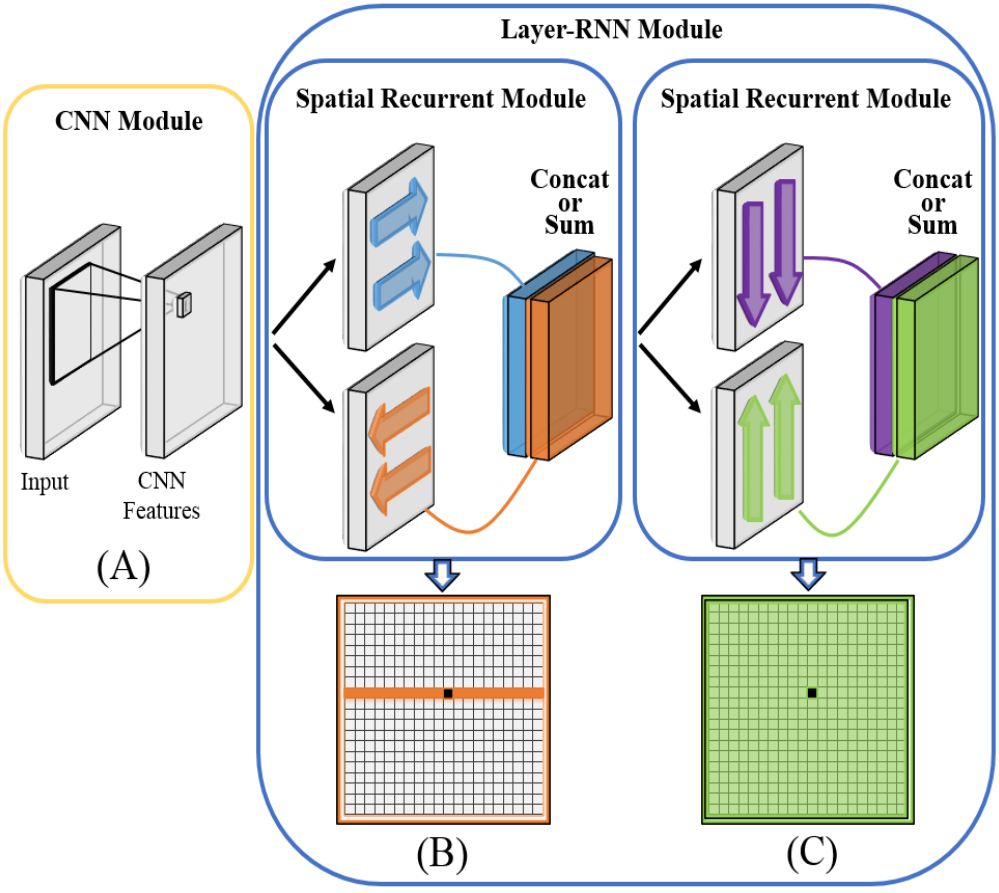


Figure 3.1: Hybrid Networks with CNNs and L-RNNs.
Given the input image, local features are calculated by the CNN module (A).
In (B), two 1D spatial RNNs are applied to scan along each row independently from different directions, hidden states are calculated at every spatial step, and the output feature maps can either be concatenated or summed up. The receptive field for the black pixel in (B) is labelled in orange;
In (C), two 1D spatial RNNs are applied to scan along each column from two directions, the receptive field for the black pixel in (C) is able to cover the whole image of input. The combination of (B) and (C) can define the L-RNN to approximate 2D kernels.

3.2.1 Layer-RNN Module (LRNN)

As shown in Figure 3.1, the Layer-RNN (L-RNN) module is a combination of the 1D spatial recurrent modules (B) and (C). In each module, there are two 1D RNNs scanning across the feature maps horizontally or vertically from two directions, and their hidden states are updated at every spatial step. Consequently, for each of the horizontal and vertical directions, two output feature maps are obtained with the same width and height as the input feature maps, for simplicity, these output feature maps are summed up (an alternative is to concatenate the output feature map, but that would increase the number of parameters).

More formally, assume the feature maps (layer L) coming into the L-RNN module are $X \in \mathbb{R}^{m \times n \times d}$ and output H (layer $L+1$), where m, n, d refers to the width, height, and the number of channels respectively for the input feature map. For simplicity, we take the left-to-right direction as an example for the following discussion, other directions are processed independently in the same manner. While scanning from left to right, each row on the feature maps is then treated as one sequence. the feature responses for location $t+1$ can be calculated as:

$$h_t = f(Ux_t + Vh_{t-1} + b) \qquad \text{left to right} \tag{3.1}$$

Where $h_0 = 0$, $x_t \in \mathbb{R}^{d \times 1}$, $h_t, h_{t-1} \in \mathbb{R}^{D \times 1}$, $U \in \mathbb{R}^{D \times d}$, $V \in \mathbb{R}^{D \times D}$, $b \in \mathbb{R}^{D \times 1}$, D denotes the number of nodes used in the 1D spatial RNNs, and f refers to the non-linearity function. Notice that, the first term of Eq 3.1 encodes local information independently, resembling the normal convolutional layer, and the second term characterizes the *within-layer* recurrence. We will make use of this observation in § 3.2.2.

3.2.2 Adding L-RNN to A Pre-trained CNNs

In this section, we will describe how a Layer-RNN module can be seamlessly inserted into a pre-trained CNNs. In a typical scenario, we make use of the CNNs that is pre-trained for classification on ImageNet (where there are copious annotations), after inserting L-RNN, the hybrid L-RNN network can then be re-purposed with

the available annotation and trained end-to-end for the new task, such as pixel-wise prediction, e.g. semantic segmentation (where the annotated data is usually more limited). This naturally allows multi-scale contexts to be incorporated effortlessly, whilst still benefiting from the earlier classification training (Figure 3.2).

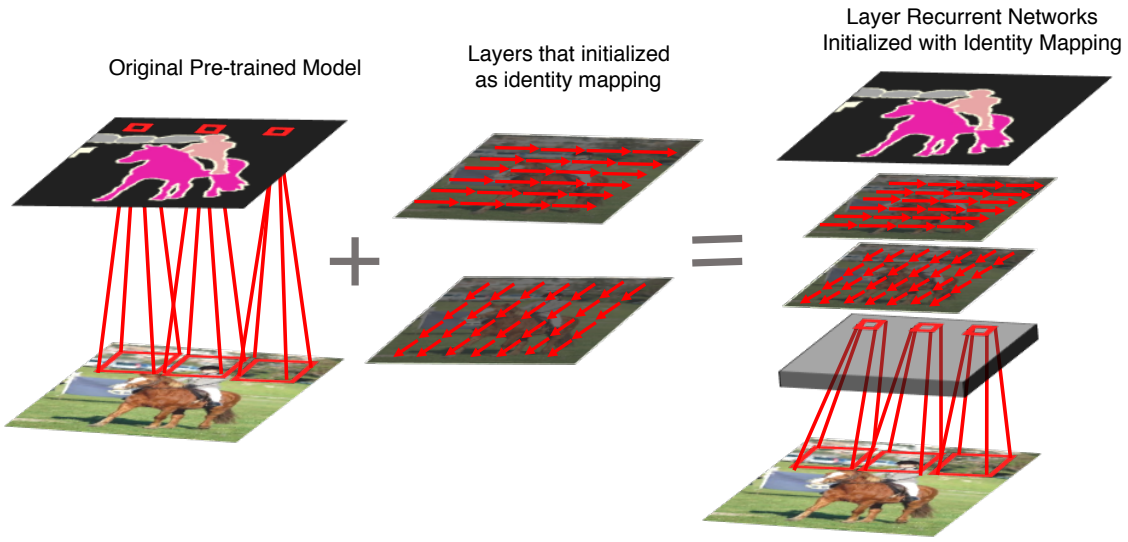


Figure 3.2: Fine-tuning the long-range dependency with identity initialized layer.

The idea is illustrated using 1D convolution, but same principle holds for the entire L-RNN module. Canonical CNNs architecture for a 1D convolution can be denoted as:

$$h = f(W * x + b) \tag{3.2}$$

where $*$ refers to convolution, W and b are the parameters of the CNN, x, h refer to L and $L + 1$ layer respectively.

While the 1D spatial RNNs can be written as :

$$h_t = f(U * x_t + V h_{t-1} + b) \tag{3.3}$$

where U, V, b refer to the parameters that are shared across the whole scan-line.

Notice that, the 1D spatial RNNs are designed to incorporate two terms, projections from local region (input-to-hidden) and recurrence term from previous hidden unit (hidden-to-hidden). In fact, it is the presence of non-zero recurrence

matrix V that characterizes the 1D spatial RNNs, therefore standard spatial RNNs can be calculated in a two-step way as :

$$y = U * x \quad \text{(Convolution)} \tag{3.4}$$

$$h_t = \begin{cases} f(y + b) & (\text{ t = 1, zero initial states }) \\ f(y + Vh_{t-1} + b) & (t > 1) \end{cases} \tag{3.5}$$

By re-interpreting the recurrence in this way, 1D spatial RNNs can be constructed by inserting recurrence directly into any CNNs layer right after the convolution. If the recurrence matrix V is initialized as zero, and ReLU is used as the activation function, then the 1D spatial RNNs have essentially been initialized as the pre-trained CNNs exactly. The complete L-RNN can be constructed by using two 1D spatial RNNs in subsequent layers of the pre-trained CNNs.

3.2.3 Fine-tuning L-RNN with Zero Recurrence Matrix

In this section, we will derive the *within-layer* gradient flow for recurrence matrix in backpropagation, when it is initialized as *zeros*. Again, we will only consider 1D scan-lines of the spatial RNNs, and therefore simplify the derivation to a 1D sequence. Consider the fully connected layer for simplicity, X, H denote $L, L+1$ layer, t refers to the index of input, f refers to ReLU, U, V refer to the input-hidden matrix and recurrence matrix respectively.

$$s_t = U * x + Vh_{t-1} + b$$

$$h_t = f(s_t)$$

Assume E denotes the loss function for a specific task. Since V is shared for the whole 1D sequence (length denoted by T), the back-propagation *within* the layer $L + 1$ can then be derived as:

$$\frac{\partial E}{\partial V} = \sum_T \sum_{t \leq T} \frac{\partial E}{\partial h_T} \cdot \frac{\partial h_T}{\partial h_t} \cdot \frac{\partial h_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial V} \tag{3.6}$$

where $\frac{\partial E}{\partial h_T}$ refers to the gradients of loss layer w.r.t. the layer output,

$$\frac{\partial h_T}{\partial h_t} = \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial h_{T-2}} \cdots \frac{\partial h_{t+1}}{\partial h_t} \quad \text{and} \quad \frac{\partial h_{t+1}}{\partial h_t} = V^T \cdot diag(f') \tag{3.7}$$

Each Jacobian $\frac{\partial h_{t+1}}{\partial h_t}$ is a product of two matrices: (a) the recurrence weight matrix V, and (b) the diagonal matrix composed of the derivative of ReLU (f'). Therefore, when V is initialized as zeros at the starting point, the L-RNN layer is initialized as a pre-trained convolutional layer exactly, and no long-range dependencies are considered.

During the first parameter update, $\frac{\partial h_{t+1}}{\partial h_t}$ are zeros for $t < T$, while for $t = T$:

$$\frac{\partial E}{\partial V} = \sum_T \frac{\partial E}{\partial h_T} \cdot \frac{\partial h_T}{\partial s_T} \cdot \frac{\partial s_T}{\partial V} \tag{3.8}$$

where $\frac{\partial h_T}{\partial s_T} = f'$ and $\frac{\partial s_T}{\partial V} = h_{T-1}$. Therefore, by gradient descent, we have:

$$V_1 = V_0 - \alpha \frac{\partial E}{\partial V} \tag{3.9}$$

Since V_0 has been initialized as zeros, V_1 therefore becomes $-\alpha \frac{\partial E}{\partial V}$ after the very first training iteration. In other words, instead of initializing the recurrence matrix V randomly or to be identity matrix, we actually initialize it based on the features in a local neighbourhood (Eq 3.8). During the back-propagation of the spatial RNNs, gradients will flow *within* layers first, and then $\frac{\partial E}{\partial U}$ (*between layers*) is calculated in the same way as normal convolutional layers.

3.3 Experiment

In this section, we evaluate the proposed Layer-RNN on PASCAL VOC 2012 segmentation task.

Dataset & Evaluation The training set consists of the original VOC 2012 training data (1464 images provided by the challenge organizers), and is augmented with training and validation data from [52], which further extends the training set to a total of $11,685$ images with pixel-level annotation. After removing the overlapping images between the original VOC 2012 validation data and this training

set, we are left with 346 images for validation set. In all the following experiments, the input images are of size 384×384 pixels, and only horizontal flipping is used for data augmentation. The performance is measured in terms of pixel intersection-over-union (IOU) averaged across the 21 classes.

3.4 Result

In this section, we insert L-RNN modules into the VGG-16 networks (pre-trained on ImageNet [2]), and fine-tune the entire network for the PASCAL VOC 2012 segmentation task. The objective is to boost the segmentation performance by providing contextual information via the L-RNNs. In particular, we consider the two FCNs segmentation architectures originally introduced by [43], FCN-32s and FCN-8s; these are described below.

We proceed in three steps: first, we establish strong baselines by training our own FCN-32s and FCN-8s, and comparing their performance to those of [43]. We also investigate the loss in performance as the fully connected (FC) layer is gradually reduced from 4096 to 512 channels. The reason for doing this is that when we insert the L-RNN module, its complexity (dimension of the hidden units) depends on this number of channels, so the overall complexity can be varied. In the second step, we insert L-RNNs into the FCN-32s architecture and evaluate the change in performance. Finally, we insert L-RNNs into the FCN-8s architecture and compare with previous published methods.

3.4.1 Baseline Architectures & Training

Architecture & Training In the FCN-32s, input images are passed through the whole networks, and end up with predictions of $12 \times 12 \times 21$, then, up-sampling layers are directly used to map the predictions back to 384×384 (32 times). In the FCN-16s, instead of directly up-sampling 32 times, the predictions are first up-sampled by 2, and summed up with predictions from pool4 (named after VGG16), then up-sampled by 16 times. In the FCN-8s, the stream predictions from pool3 are further added to the results from FCN-16s, thus, up-sampling layers with only factor 8 is needed.

For all the architectures, the base net (VGG16) is pre-trained on ImageNet [2], and we further train the proposed architecture on Pascal VOC 2012 for 50 epochs. The 4096 channel architectures are trained first, and then the number of channels is gradually reduced in the FC layer by randomly pruning them (e.g. from 4096 to 2048), and re-training the networks.

Result & Discussion Table 3.1 shows the performance of the six baselines: FCN-32s and FCN-8s with the number of channels varying from 512 to 4096. We observe that reducing the nodes in the FC layers does produce a performance drop (from 4096 to 1024 nodes, 1% mean IOU) in both FCN-32s and FCN-8s. Although from 1024 to 4096 nodes, the improvement is tiny, the difference in the number of parameters is over 64 million. Consequently, in the following experiments we choose to perform experiments based on networks with $512, 1024$ or 2048 channels only (i.e. not 4096). In comparison to the original performance for the FCN-8s architecture in [43], we exceed this (by 64.4 to 61.3 mean IOU) in our training.

3.4.2 FCN-32s with L-RNN Modules

Architecture & Training The FCN-32s (L-RNN) architecture is shown in Figure 3.3, where the convolutional part of the architecture is initialized as the pre-trained FCN-32s (2048 channels in FC layer) baseline, then, two 1D spatial RNNs are inserted into the fc1 layer in the horizontal direction, and two 1D spatial RNNs are inserted into the fc2 layer in the vertical direction. The convolution activations of fc1 are shared for both left-right and right-left scanning, as the first step of spatial RNNs. Similarly, for fc2, the convolution activations are shared for top-down and bottom-up column-wise scanning. Therefore, the fc1 and fc2 layers together with the added 1D spatial RNNs form a complete L-RNN module.

During training, as described in § 3.2.2 (initializing L-RNN as identity), the 1D spatial RNNs are initialized with zero recurrence matrices. The entire network is fine-tuned on the PASCAL VOC 2012 data end-to-end. We adopt RMS-prop [23]

for 30 epochs with hyper-parameters $lr = 10^{-4}$, $\rho = 0.9, \epsilon = 10^{-8}$, then decrease the learning rate to $lr = 10^{-5}$ for 10 epochs.

Result & Discussion The results are shown in Table 3.1. Compare the 32s rows with and without the L-RNN for the FC layers with 512, 1024, and 2048 channels. As can be seen, the addition of the L-RNN always improve the segmentation performance over the pre-trained FCN-32s baselines. However, the improvement is not large – about 1-1.5% mean IOU. This is because the receptive field in the fully connected layers of FCN-32s is sufficiently large to cover 224×224 pixels of the input patch, and consequently, the networks do not benefit much from the context provided by the L-RNN. The benefit is greater when L-RNNs are added to the lower layers (where the receptive fields of the convolutions is much smaller), and we turn to that case next.

3.4.3 FCN-8s with L-RNN Modules

Architecture & Training The architecture FCN-8s (L-RNN) is shown in Figure 3.3, as with the FCN-32s architecture, 1D spatial RNNs are inserted into the fc1 and fc2 layers to form a L-RNN module. L-RNNs are also inserted into the lower layers, namely pool3 and pool4 layers. Unlike the FC layers in the FCN-32s, where the prediction for each central pixel comes from image patches of size 224×224, the predictions from pool3 and pool4 are based on receptive field on the image of much smaller sizes (around 44×44 and 100×100 pixels respectively). Thus, the inserted L-RNN modules must be able to model relatively long-range dependencies.

During training, the network is initialized from the FCN-8s baseline, and then fine-tuned with segmentation data. Again the PASCAL VOC 2012 dataset is used, when comparing to the other previously published methods, the network is further trained on the COCO trainval dataset, and a densely connected CRF is used as post-processing [53].

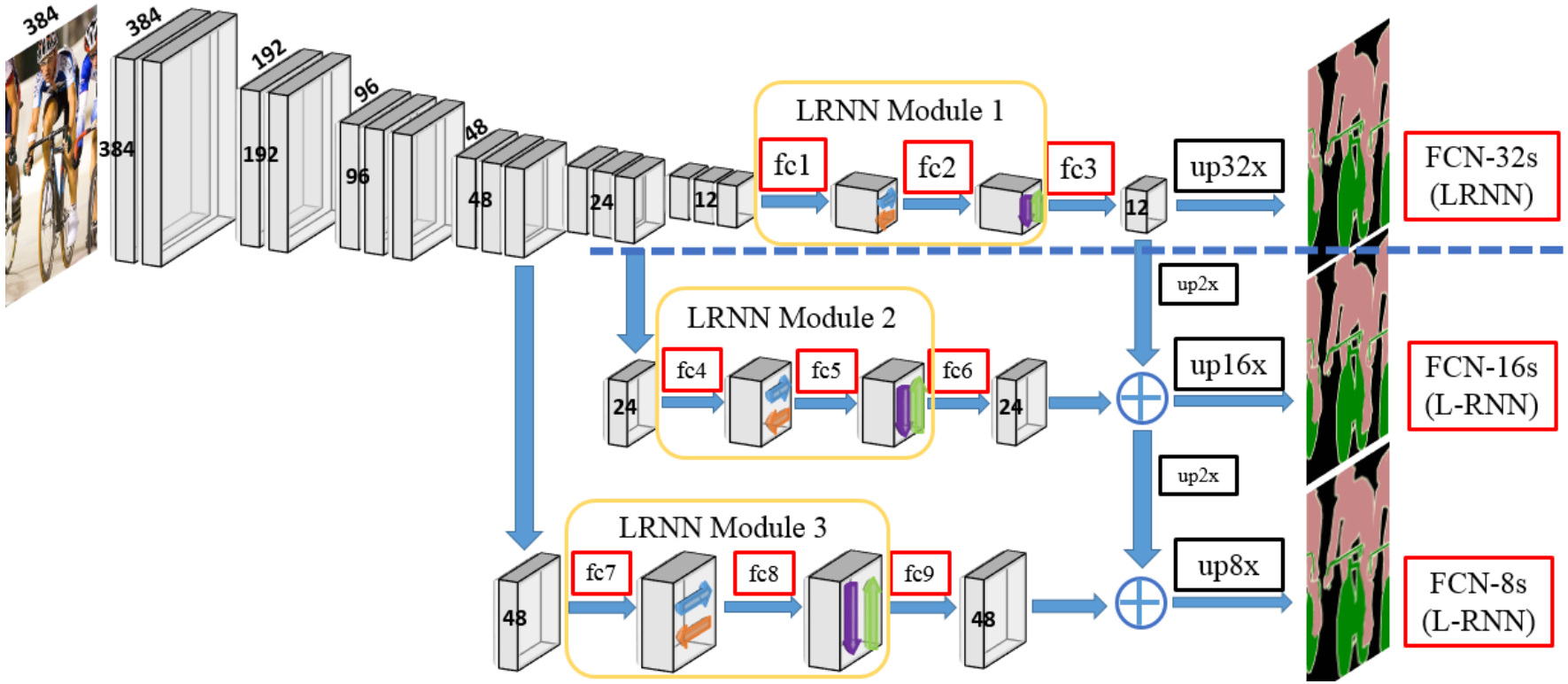


Figure 3.3: FCN-32s and FCN-8s with L-RNN.
Spatial RNNs are inserted to the fully connected (FC) layers in all FCNs, every two FC layers construct a complete L-RNN module. $\{384, 192, 96\}$ indicate the spatial sizes of the feature maps.
Kernel sizes for the fully connected layers (n refers to the number of channels) :
fc1 : $7 \times 7 \times 512 \times \ n\ $, fc2 : $1 \times 1 \times \ n\ \times \ n\ $, fc3 : $1 \times 1 \times \ n\ \times 21$
fc4 : $1 \times 1 \times 512 \times 1024$, fc5 : $1 \times 1 \times 1024 \times 1024$, fc6 : $1 \times 1 \times 1024 \times 21$
fc7 : $1 \times 1 \times 256 \times 1024$, fc8 : $1 \times 1 \times 1024 \times 1024$, fc9 : $1 \times 1 \times 1024 \times 21$

Result on PASCAL VOC validation set The results are shown in Table 3.1. Compare the rows with 2048 channels for 32s with and without L-RNN, to those for 8s with and without L-RNN. It can be seen (for IOU) that going from FCN-32s (62.7) to FCN-8s (64.1) brings an improvement due to the skip layers. However, adding in the L-RNNs brings an improvement from 64.2 for FCN-32s-L-RNN to the very high value of 69.2 for FCN-8s-L-RNN. The reason for this substantial improvement is that when inserting the L-RNN after pool3 and pool4 in FCN-8s, the L-RNN is able to learn contextual information over a much larger range than the receptive field of pure local convolutions. As noted earlier, in the FCN-32s architecture the L-RNN is inserted in the FC layers, and their receptive field already covers the input patch of size 224×224 (less context to contribute here).

Type	**# of channels in FC**	L-RNNs added	**Pixel Acc %**	**Mean IOU %**
32s	512	NO	90.4	61.5
32s	1024	NO	90.5	62.1
32s	2048	NO	90.7	62.7
32s	4096	NO	90.7	62.9
8s	1024	NO	91.3	63.8
8s	2048	NO	91.2	64.1
8s	4096	NO	91.3	64.4
8s ([43])	4096	–	–	61.3
32s	512	YES	90.8	62.7
32s	1024	YES	90.9	63.4
32s	2048	YES	91.1	64.2
8s	**2048**	**YES**	**92.6**	**69.1**

Table 3.1: FCN networks on the PASCAL VOC 2012 validation set.

Result on PASCAL VOC test set Table 3.2 shows the results of the FCN-8s with L-RNNs on the PASCAL VOC test data, and also compares to others who have published on this dataset. The performance is far superior to the original result [43] using a FCN-8s with 4096 channels (whereas only 2048 channels are used here). When compare to the dilated convolution network of [46], comparable, though slightly better performance is obtained. Note that, in [46], multi-scale contextual information is captured by explicitly designing dilated convolution kernels, while the L-RNN is able to learn contextual information implicitly. Finally,

if compare to [49] which adds a densely connected CRF to FCN-8s. If a dense CRF is added as post-processing, the performance can be boosted by 1% in IOU (the same boost as obtained by [46]).

	Mean IOU %			
Methods	**P**	**P+CRF**	**P+C**	**P+C+CRF**
FCN-8s [43]	62.2	n/a	n/a	n/a
CRF-RNNs [49]	n/a	72.0	n/a	74.7
Dilated Conv. [46]	n/a	n/a	73.5	74.7
FCN-8s-LRNN (2048)	**71.9**	**72.7**	**74.2**	**75.7**

Table 3.2: Comparison on the PASCAL VOC 2012 test set.
Training is on P: PASCAL VOC 2012; C: COCO dataset.
`http://host.robots.ox.ac.uk:8080/anonymous/YJBLI7.html`

Model										
FCN-8s	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5
CRF-RNNs	90.4	**55.3**	88.7	**68.4**	69.8	88.3	82.4	85.1	32.6	78.5
Dilated Conv	91.3	39.9	88.9	64.4	69.8	88.9	82.6	**89.7**	34.7	82.7
FCN-8s-LRNN	**92.5**	42.1	**89.7**	66.3	**73.0**	**90.4**	**84.7**	88.3	**36.8**	**83.0**
Model										
FCN-8s	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1
CRF-RNNs	**64.4**	79.6	81.9	86.4	81.8	**58.6**	82.4	53.5	77.4	70.1
Dilated Conv	59.5	83.0	**88.4**	84.2	**85.0**	55.3	**86.7**	**54.4**	81.9	63.6
FCN-8s-LRNN	62.2	**83.8**	84.9	**87.5**	84.7	55.4	83.3	53.7	**82.0**	**70.3**

Table 3.3: Detailed comparison on the PASCAL VOC 2012 test set.

In Figure 3.4, representative samples of semantic segmentations on the PASCAL VOC 2012 validation set are shown. In each figure, it contains predictions from the hybrid networks and the results after CRF post-processing. Comparing with the outputs from the end-to-end trainable CRF-RNN [49], the proposed hybrid networks miss some small details, like the wheel of the bicycle, but show much better performance in determining the class of the segmented regions – something that context can really contribute to.

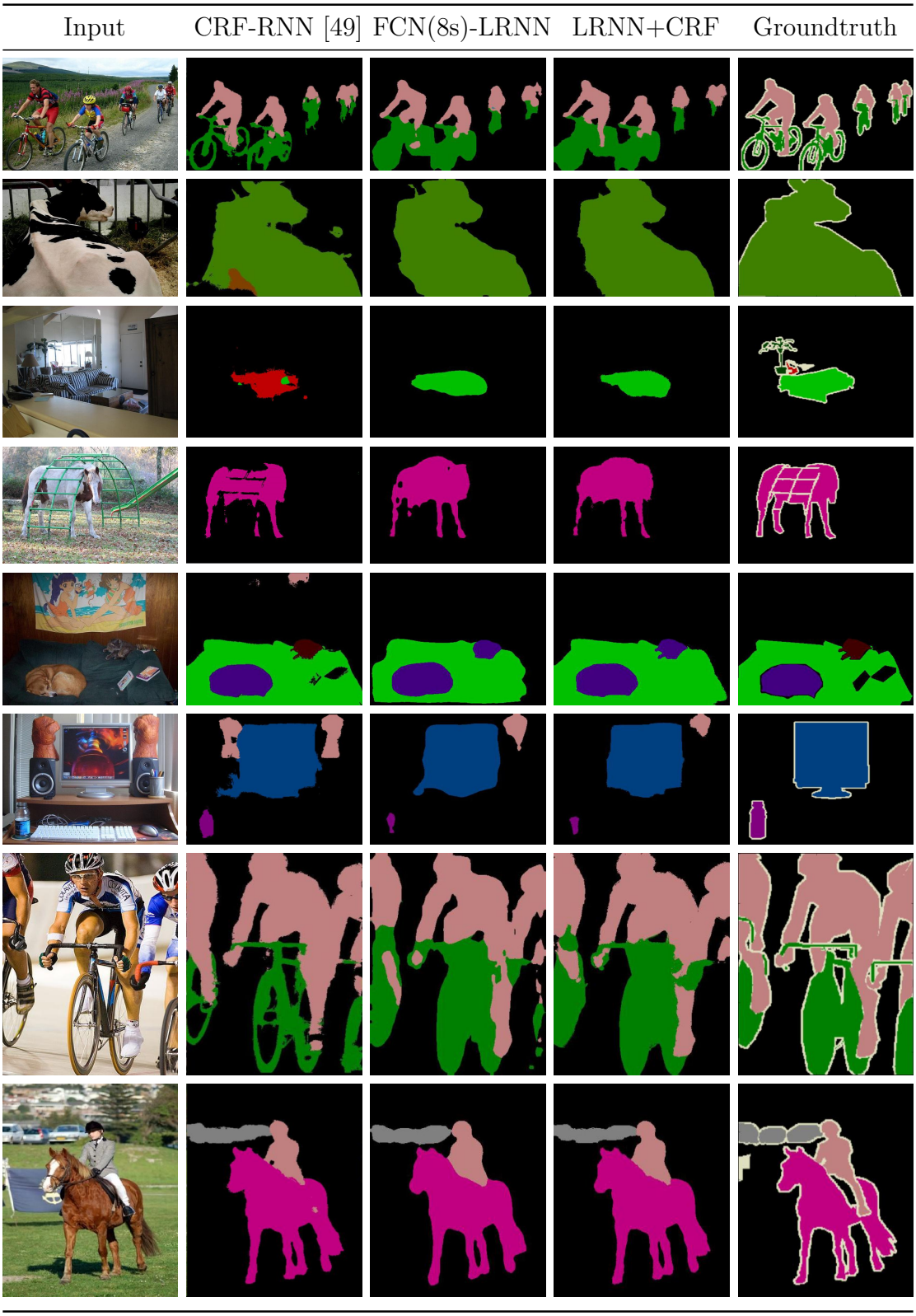


Figure 3.4: Representative results.
First column: input image. Second column: prediction from [49]. Third column: prediction from the our networks. Fourth column: CRF post-processing. Fifth column: ground-truth annotation.

3.5 Discussion

This chapter has proposed to add multi-scale spatial contextual infromation by within-layer RNNs, namely Layer Recurrent Neural Networks (L-RNNs). We have demonstrated that inserting L-RNNs can boost the performance of pre-trained networks, and given an initialization procedure that makes this training a simple matter of end-to-end fine tuning.

4

Comparator Networks for Face Recognition

In this chapter, the goal is to explore new architectures for face recognition problems. We first describe several datasets that are used for training and evaluation (§ 4.2). In § 4.3, we introduce a semi-supervised pipeline for face alignment, where only the image-level supervision is used, i.e. the identity label. In § 4.4, we consider the face recognition problem as a fine-grained verification problem, and propose the Deep Comparator Networks (DCNs), which is designed to learn both image-level and part-level representation from a set of images, and compare these set-based features in a pairwise way.

4.1 Introduction

Face recognition is undoubtly one of the most widely researched area in computer vision. In recent years, driven by the success of image classification [1, 32, 35], Convolutional Neural Networks (CNNs) have also shown remarkable success on face recognition, e.g. VGGFace, DeepFace, FaceNet [54–56]. Based on the assumption that, face descriptors of the same identity should all live in the same sub-space, the faces in the image are detected, cropped and fed into the powerful CNNs, mapped to a fixed-length feature vector, and trained with classification loss or

metric learning losses. Taking benifit of the publicly available large-scale dataset, e.g. VGGFace, MSRA-Celeb, VGGFace2 [13, 54, 57], these methods have already provided very strong baselines on a set of public benchmarks, e.g. LFW, Youtube Face, IJB-A, IJB-B [58–61].

Despite the success of these approaches for face identification and verification, current face recognition systems still suffer from several limitations and challenges:

- In-the-wild face images can potentially have very complicated variations, such as pose, illumination, ages, etc. Naturally, each face image is of different significance when calculating feature representations, for example, profile faces, low resolution images are often not as interesting and useful as the frontal, high resolution images. Therefore, an ideal face recognition system should be able to pick good images and reject those of poor quality.

- While mapping the face image to one fixed-length feature representation, subtle details of the face can be omitted due to pooling operations in CNNs, with only the high-level semantics encoded. However, these small details may be very important for distinguishing identities, for intance, between siblings, or cousins. Therefore, an ideal face recognition system should not only be able to encode global information, but also include facial parts, such as eyes, mouth and mouth.

- Besides good recognition performance, model interpretability should be as important as the accuracy. In the typical real-case senario, we are not only interested in the final prediction result, but also how and why the system makes such decision. Whenever the system gives wrong verification results on two face images, we want to understand what is exactly the reason, for instance, it maybe due to the face is occluded, extremely blurred, ill-posed, terrible illumination, etc. each of these situations may cause the model to fail localize the eyes, nose, or even the entire face.

In the literature, apart from the current CNNs approaches, several previous works have proposed to use part-based representation for face images or face tracks. In [62], each face image is densely partitioned into overlapping patches at multiple scales, and each patch is then represented by local features, such as the Local Binary Pattern (LBP) or SIFT. By clustering, these local features are further fused into the bag of spatial-appearance features. In [63], Fisher Vector (FV) encoding is used to aggregate local features across different video frames, and form a video-level representation. As the paper shows, each Gaussian component acts as dense pseudo part detectors that is invariant to face identities, therefore, the comparison of those Fisher Vectors can be seen as an implicit and reliable comparison of facial parts.

4.2 Dataset Description

VGGFace2 Dataset In this chapter, we will experiment with a new large-scale face recognition dataset, VGGFace2 [13]. The dataset contains about 3.31 million images of 9131 subjects (identities). The images are downloaded from Google Image Search and have large variations in pose, age, illumination, ethnicity and profession (e.g. actors, athletes, politicians). Approximately, the dataset is gender-balanced, with 59.7% males, varying between 87 and 843 images for each identity, with 362.6 images on average.

Two different crops (loose and tight) are used in this chapter, as shown in Figure 4.1. The tight bounding boxes are predicted by the Multi-task Cascaded Convolutional Networks (MTCNN [64]).

We further predict the 5 facial keypoints with the pre-trained landmark detectors (Figure 4.9), and estimate the face poses. In our case, the face poses are simply thresholded into three categories, e.g. frontal, facing left and facing right.

In §4.3 on face alignment, we use the loosely cropped images with only identity label for training, though there may still contain noise in those labels, we refer them as groundtruth labels. While in §4.4 on Comparator Networks, we use the tightly cropped images. Apart from the identity label, we also use the predicted 5 facial landmarks and the estimated poses (*psudo* ground-truth) as regularizers

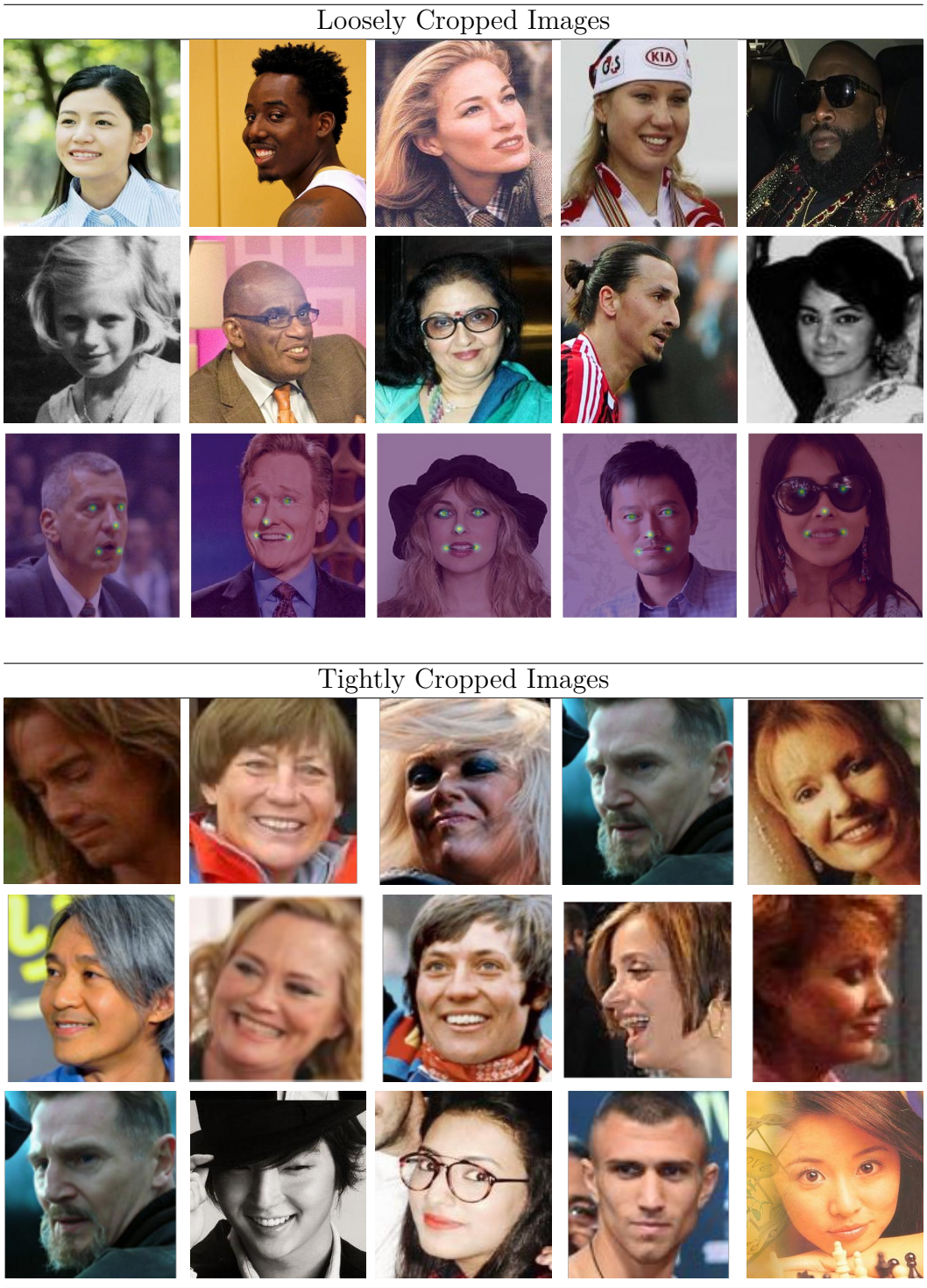


Figure 4.1: Example images from VGGFace2.
The dataset was collected with large variations on pose, age, ethnicity.

during training. For both sections, We train the entire architecture only on the 8123 identities, and keep 1008 identities as test set (following the original VGGFace2 paper).

Figure 4.2: Facial landmark detection for VGGFace2 images.
Face poses are quantized into three categories based on the ration α/θ. Left-facing profile : $\alpha/\theta < 0.3$, right-facing profile: $\alpha/\theta > 3.0$, frontal face: $\alpha/\theta \in [0.3, 3.0]$

4.3 Face Alignment by Recognition

Face alignments have been widely considered as an important component of the face processing pipeline. However, aligning faces in the unconstrained scenario remains a difficult problem due to the complex variations, e.g. poses (both in-plane and out-of-plane rotation), facial expressions, lighting, etc. Recently, Facebook([55]) proposed the DeepFace architecture, where they train landmarks detector for face frontalization and alignment, and achieve better recognition performance. In this section, our goal is to investigate the possibility of aligning face images with only person identity labels.

4.3.1 Method

The proposed model achieves the face alignment in an end-to-end differentiable CNNs framework (Figure 4.3), allowing for the face alignments and recognition to be interdependent. It consists of two phases, *First*, the full-resolution, loosely cropped face image I undergoes a tiny localization network, which is designed to predict four parameters, t_x, t_y for translation, θ for rotation, and s for scaling factor. *Second*, the four predicted parameters will be fed into a spatial transformer to reconstruct the similarity transformation matrix, and resampling the original input image into a form that most benifits the further classification tasks.

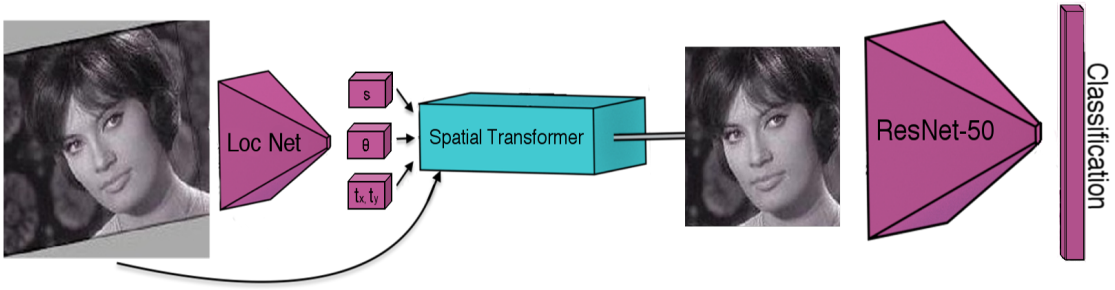


Figure 4.3: Overview of the face recognition architecture. The initial, loosely cropped face image is passed into a localizaiton network with several convolution and max pooling layers, producing four paramters, e.g. t_x, t_y for translation, θ for rotation, and s for scale factor. Resampled image is further passed to a ResNet-50 for face classification.

Geometric Transformation (STN)

The Spatial Transformer Network (STN) was originally proposed as a general-purposed layer for classification tasks that require spatial invariance for high performance. In [40], the STN is to learn whichever transformation parameters that are most helpful for the classification task. The predicted transformation matrix is used to transform the intermediate *feature maps*. In contrast, in this application, we are particularly interested in learning to transform the *input image*.

Localization Network (LocNet) In the LocNet, we use four convolutional layers (3×3 kernel), each followed by batch normalization and 2×2 max poolings. Eventually, the feature maps are flattened and to predict the transformation parameters. As we have restricted the transformation to be a similarity transformation, the matrix can therefore be decomposed into three separate matrices:

$$M = SRT,$$

where T is the translation matrix:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix};$$

R is the rotation matrix:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

and S is the (uniform) scaling matrix:

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that the images are defined on a normalized coordinate space $\{x, y\} \in [-1, +1]$, such that rotation and scaling occur relative to the image center. In practice, the LocNet learns to predict only the relevant parameters, $\mathbf{m} = \begin{bmatrix} t_x & t_y & \theta & s \end{bmatrix}^\top$.

Grid generation and sampling For image sampler, we use the same method as the original STNs paper (described in §2.4.6). The re-sampled image is further passed a ResNet-50, and trained with standard classificaiton loss.

4.3.2 Training Details

During training, images of $256 \times 256 \times 3$ pixels are passed as input to the architecture, and the resampled images is of size $160 \times 160 \times 3$ pixels. We explicitly provide regularizers (pseudo groundtruth) for the transformation parameters, e.g. for rotation parameter,

$$L(\theta) = \begin{cases} 0, & \text{if } -\frac{\pi}{2} < \theta < \frac{\pi}{2} \\ (\theta - \hat{\theta})^2, & \text{otherwise} \end{cases} \tag{4.1}$$

where $\theta, \hat{\theta}$ refer to the network prediction and pesudo ground-truth respectively, and $\hat{\theta} \in Uniform(-0.05, 0.05)$. Similarly, regularizers are also provided for calculating $L(t_x), L(t_y), L(s)$, when t_x, t_y, s are not in the range $[-0.5, 0.5], [-0.5, 0.5], [0.4, 1.2]$ respectively.

4.3.3 Qualitative Results

We run the architecture through the images from test set, as shown in Figure 4.4. Although the entire classification architecture is only trained with identity lables, alignment is achieved as a by-product.

We further transform all the images from VGGFace2, send these transformed images back to the facial landmark detectors (MTCNN), and calculate the poses by simply thresholding the angle ratio. As shown in the Figue 4.5, the average image

and landmark points are calculated to visualize the alignment results. Note that the network is designed to predict similarity transformation, only in-plane rotation is reduced, out-of-plane rotation and facial expression can still lead blurry results in the average image. Meanwhile, CNNs can tolerate small variations, and small rotation jittering is often ignored by the classification networks.

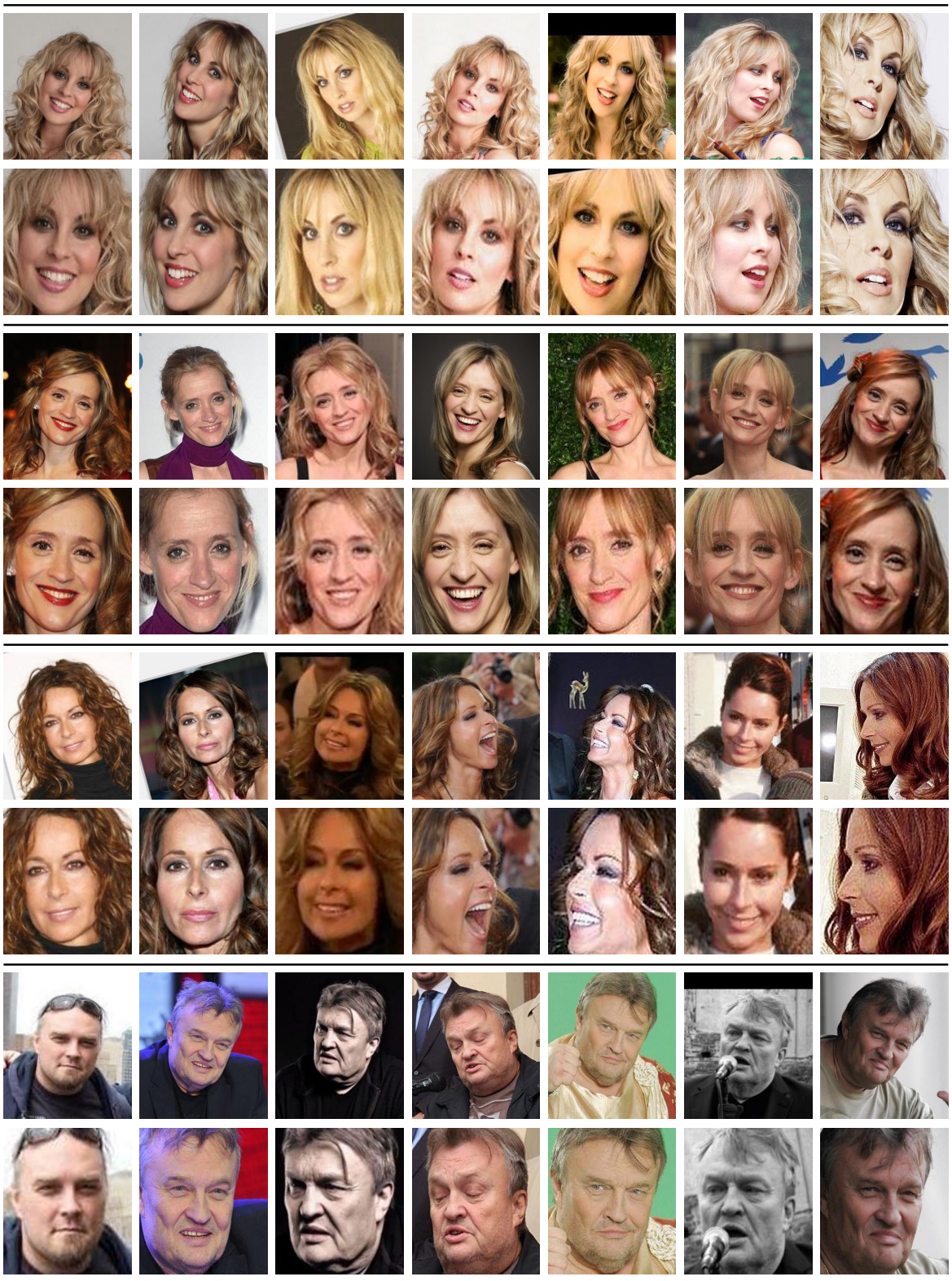


Figure 4.4: Qualitative Results for face alignment. Here we show the face images (from VGGFace2 test set) before and after alignment. In each block, first row shows the loosely cropped face images, and the second row shows the faces transformed with the spatial transformer networks.

Figure 4.5: Alignment results. The average image and landmark points of same identity are calculated to visualize the alignment results. See text for discussion.

Interestingly, the landmark estimations for some difficult cases can be improved by using the tranformed images (by STN) as the inputs (Figure 4.6). Therefore, we further purified all of the landmark predictions by passing the transformed images to MTCNN, and recalculate all the poses.

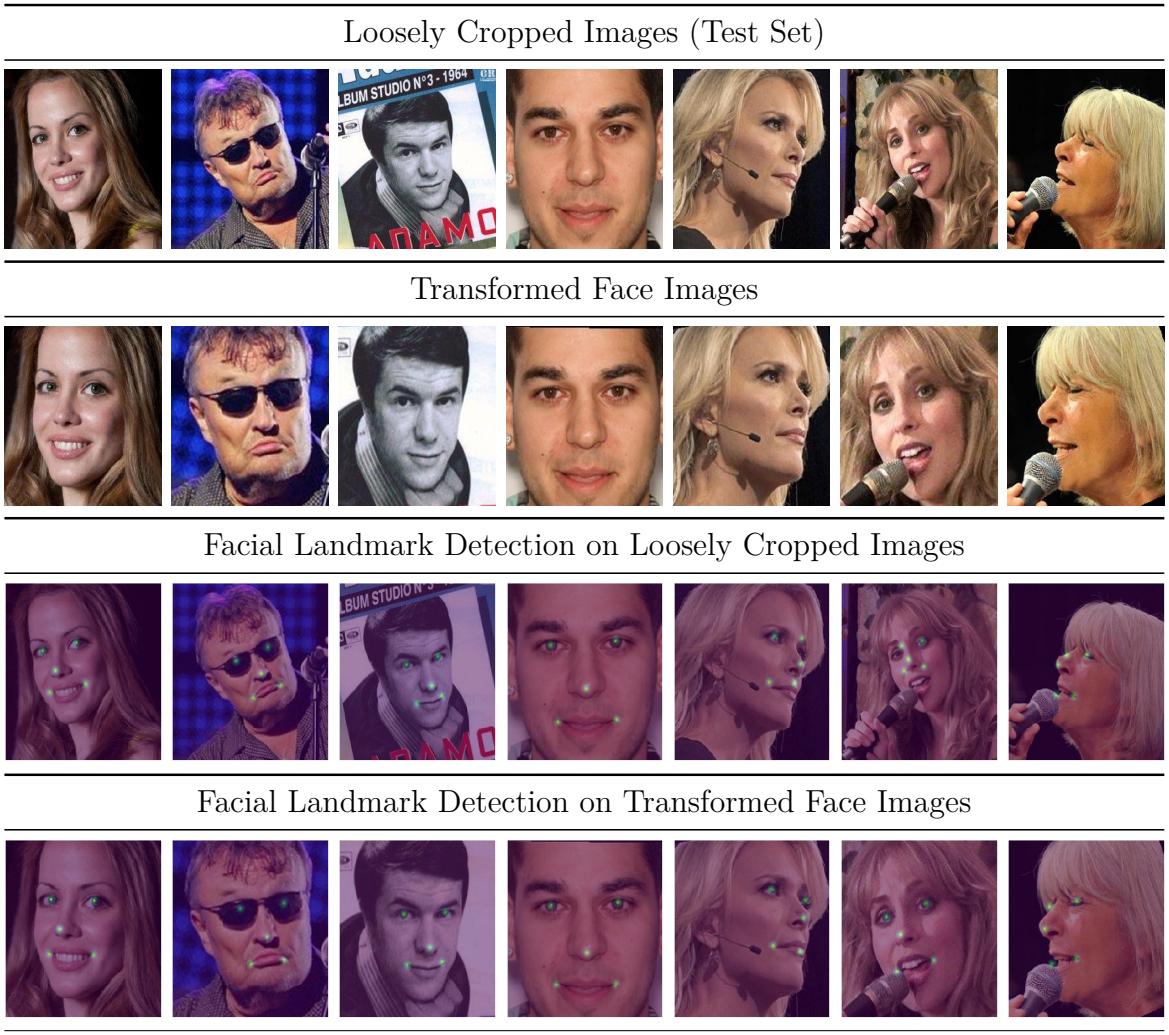


Figure 4.6: Alignment and landmarks detection. The loosely cropped images (may contain tight cropped images, if the face originally occupies the full image), have been transformed to the view that best aids the subsequent classification task. Facial landmarks detections have been improved by passing the pre-aligned images back to MTCNN.

4.4 Deep Comparator Networks

4.4.1 Introduction

The objective here is to determine if two sets of images are of the same object or not. For example, in the case of face verification, the set could be images of a face; and in the case of person re-identification, the set could be images of the entire person. In both cases the objective is to determine if the two sets show the same person or not.

In the following, we will use the example of sets of faces, which are usually referred to as 'templates' in the face recognition literature, and we will use this term from here on. A template could consist of multiple samples of the same person (e.g. still images, or frames from a video of the person, or a mixture of both). With the great success of deep learning for image classification [1, 32, 35, 65], by far the most common approach to template-based face verification is to use a deep convolutional neural network (CNN) to generate a vector representing each face, and simply average these vectors to obtain a vector representation for the template [13, 54–56]. Verification then proceeds by comparing the template vectors. Rather than improve on this simple combination rule, the research drives until now has been to improve the performance of the single image representation by more sophisticated training losses, such as Triplet Loss [54, 55]. This approach has achieved very impressive results on the challenging benchmarks, such as the IARPA IJB-B and IJB-C datasets [61, 66].

However, this procedure of first generating a single vector per face, and then simply averaging these, misses out on potentially using more available informaton in four ways:

First, *viewpoint conditioned similarity* – it is easier to determine if two faces are of the same person or not when they have a similar pose and lighting. For example, if both are frontal or both in profile, then point to point comparison is possible, whereas it isn't if one is in profile and the other frontal;

Second, *local landmark comparison* – to solve the fine-grained problem, it is essential to compare discriminative congruent 'parts' (local regions of the face) such as an eye with an eye, or a nose with a nose.

Third, *within template weighting* – not all images in a template are of equal importance, the features derived from a low resolution or blurred face is probably of less importance than the ones coming from a high-resolution perfectly focussed face;

Fourth, *between template weighting* – what is useful for verification depends on what is in both templates. For example if one template has only profile faces, and the second is all frontal apart from one profile instance, then it is likely that the single profile instance in the second template is of more importance than the frontal ones.

The simple average combination rule cannot take advantage of any of these four – for example, unweighted average pooling ignores the difference in the amount of information provided by each face image. In fact the template vector will be restricted to the convex hull of its constituent face vectors [67], and an abherrant image, such as one that is quite blurred, can have a significant effect (since most blurred face images look similar).

In this part, we introduce a *Deep Comparator Network* (DCN), a network architecture designed to compare pairs of templates. The model consists of three modules: *Detect, Attend* and *Compare*, as illustrated in Figure 4.7, that address the four requirements above, in the *Detect* module, besides the dense feature representation maps, multiple discriminative landmark detectors act on each input image and generate the score maps; the *Attend* module normalizes the landmark responses over the images within template, and output multiple landmark specific feature descriptors by using image specific weighted average pooling on the feature maps, finally, the *Compare* module compares these landmark specific feature vectors between the two templates, and aggregates into *one* vector for final similarity prediction. The DCN can handle any number of images in

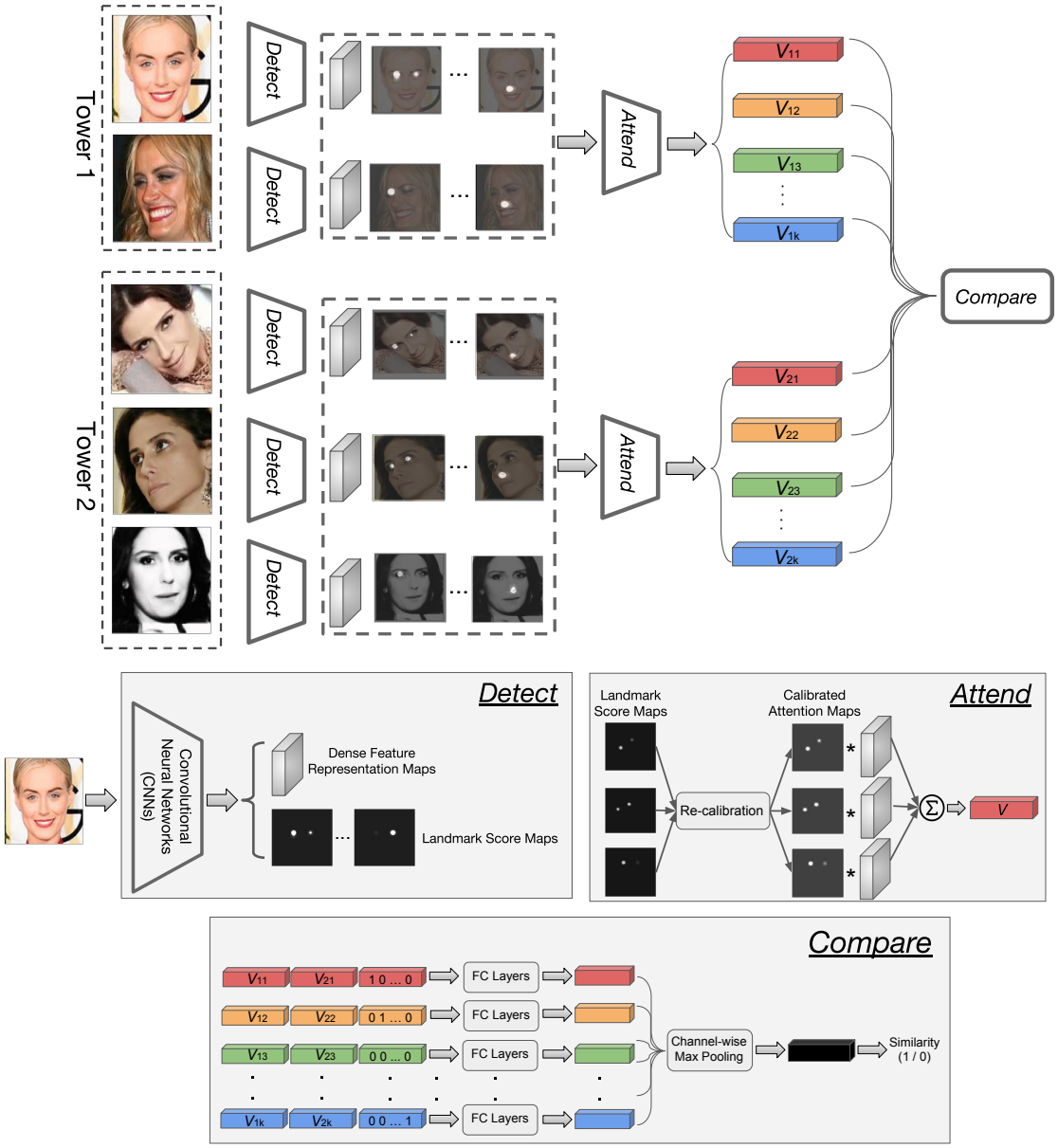


Figure 4.7: *Top*: overview of the Deep Comparator Network (DCN) *Bottom*: functionality of the individual modules, namely, *Detect*, *Attend*, *Compare*. Each of the two towers in the DCN, is able to take a template (with an arbitrary number of images) as input. Each image is fed into a shared *Detect* module and outputs a feature map, as well as multiple discriminative landmark score maps. In the *Attend* module, landmark score maps (predicted from the same filter on different input images) are first re-calibrated within the template, and then landmark specific feature responses for each template are obtained by weighted average pooling on the feature maps. In the *Compare* module, landmark specific feature responses between the two templates are compared with local "experts" (parametrized as fully conneted layers), and aggregated into *one* vector for final similarity prediction.

a template, and is trained end-to-end for the task of template verification. The network is described in detail in § 4.4.3.

As a second contribution we introduce an idea from the instance retrieval literature to face template verification. Large scale instance retrieval systems achieved superior results by proceeding in two stages: given a query image, images are first retrieved and ranked using a very efficient method, such as bag of visual words; then, in a second stage, the top k images are re-ranked using a more expensive method, such as geometric consistency with the query [68]. Since image classification models can be trained very efficiently nowadays, we repurpose this re-ranking idea for template verification as follows: during training, we employ a standard classification model (pre-trained on single image classification) for sampling the hard template pairs, such that we can explicitly control the verification difficulty level, and focus on the template pairs that single-image classification model can not deal with. This is described in § 4.5, together with other training details, such as the training set and loss functions. In § 4.6, we report the verification performance of the DCN on the challenging IARPA Janus face recognition benchmarks – IJB-B [61] and IJB-C [66]. In both datasets, the DCN is able to *substantially* outperform the previous state-of-the-art methods.

4.4.2 Related Works

In this section we review the work that has influenced the design of the DCN.

Multi-column architectures. Recent works [69, 70] extend the traditional image-wise architectures to multi-columns, where the models are designed to take a set of images/frames as inputs, and produce a single vector representation for the entire template. The model is trained to fuse useful information from the multiple inputs based on the image "quality"; for instance, high-resolution, frontal faces are weighted more than faces under extreme imaging conditions or poses. However, these models still try to encode the entire template with *one* vector, and are trained with standard classification losses. They still cannot tackle the challenge of *local*

landmark comparison and *between template weightings*.

Face recognition based on part representations. Several previous works proposed to use part-based representation for a single face image or face tracks. In [62], the face image is densely partitioned into overlapping patches at multiple scales, and each of the patches is represented by local features, such as Local Binary Pattern (LBP) or SIFT, then represented as a bag of spatial-appearance features by clustering. In [63], a Fisher Vector (FV) encoding is used to aggregate local features across different video frames to form a video-level representation. As shown in the paper, each Gaussian component acts as a dense pseudo-part detector that is invariant to face pose and identity, thus, the comparison of those Fisher vectors can be seen as an implicit and robust comparison of face parts.

Attention models. Attention models have been successfully used in machine translation [71], multiple object recognition [72], and image captioning [73]. In [74], the authors propose to extract part-based feature representations from a single input image with attention, and perform fine-grained classifications with these part specific representations. In general, the idea of these attentional pooling can be seen as a generalization of average or max pooling, where the spatial weights are parametrized as a function (usually a small neural network) mapping from input image to an attentional mask. Apart from soft attention, [40] proposed the Spatial Transformer Networks (STNs) that allows to learn whichever transformation parameters best aid the classification task. Although no ground truth transformation is specified during training, the model still implicitly learns to attend and focus on the object of interest. Recent work [75] proposed to use the STNs recursively to localize multiple facial parts from the input image. However, this approach tends to be sensitive to image quality – in the template-based recognition problems, the STNs will potentially zoom into wrong regions if the image quality is low, and therefore contribute noisy features to the templates.

Relation/co-occurrence learning. In [76], in order to perform the spatial relational reasoning, the features at every spatial location are concatenated and modelled with the features at every other locations, To model the co-occurence statistics of features, e.g. "brown eyes", bilinear CNNs [77] was proposed and experimented on the fine-grained classification problems, the descriptor of one image is obtained from the outer product of the feature maps. As for the few-shot learning, in [78], the authors propose to map a small labelled support set and an unlabelled example to its labels by learning a similarity metric with the deep neural features. As an extension, [79] experiments with more powerful representations, where the feature maps of images (from support set and test set) are concatenated and passed to a relation module for similarity learning. Similarly, in this paper, we parameterize local "experts" to compare the feature vectors from two sets.

4.4.3 Methods

We consider the task of template-based verification, where the objective is to decide if two given templates are of the same object or not. Generally, in verification problems the label spaces of the training set and testing set are disjoint. In the application considered here, the images are of faces, and the objective is to verify whether two templates show the same person or not. The identities in the test set are not seen during training.

From a high-level viewpoint, Deep Comparator Network (DCN) focus on the scenario that two tenplates (each has an arbitrary number of images) are taken as inputs, and trained end-to-end for template verification. A DCN consists of three modules (as illustrated in Figure 4.7): *Detect*, *Attend* and *Compare*, to address the four challenges, i.e. *viewpoint conditioned similarity, local landmark comparison, within and between template weighting*. We first overview the function of these modules, and then give more details of their implementation. A detailed description of the architecture of the individual modules is given in the supplementary material.

The *Detect* module is shared for each input image, and a dense feature representation map, as well as multiple discriminative part score maps are outputted,

in the face recognition literature, these discriminative parts are usually termed "landmarks", we will use this term from here on. Note that, the implicitly inferred landmarks aim to best assist the subsequent template verification task, they may not follow the same intuitions as human defined facial landmarks, e.g. a mouth corner. Given a template with multiple images in various poses or illuminations, the landmark filters will be sensitive to different facial parts, viewpoints, or illuminations, e.g. one may be sensitive to the eyes in a frontal face, one may be more responsive to a mouth in a profile face. The *Detect* module acts as the base for fullfilling template comparison conditioned on *viewpoints/local landmarks*.

The *Attend* module achieves the *within template weighting* with an internal competition mechanism, and pools out multiple landmark specific feature descriptors for each template. Given a template with multiple images, we hope to emphasize the feature representations from the relatively high quality images, while suppressing the contribution from the low ones. To achieve this, we recalibrate the score maps (inferred from different samples with the same landmark filter) into a probability distribution. Consequently, multiple landmark specific feature descriptors are calculated by attending to the feature maps with image specific attentional masks. Therefore, the viewpoint factors and facial parts are decomposed and template-wise aligned.

Finally, we use the *Compare* module to achieve the *between template weighting*. The template-wise verification is reformulated as the comparison conditioned on both global and local regions (i.e. landmarks), votings from the local "experts" are aggregated into *one* vector for the final similarity prediction.

4.4.4 Detect

The *Detect* module inputs an image, and generates an intermediate dense representation and multiple (K) landmark score maps. Formally, we parametrize the module as a standard 40-layer ResNet ($\psi(\cdot;\theta_1)$) with outputs for n images (Figure 4.8 shows an example where $n = 3$):

$$[F_1, F_2, ..., F_n, A_1, A_2, ..., A_n] = [\psi(I_1;\theta_1), \psi(I_2;\theta_1), ..., \psi(I_n;\theta_1)] \qquad (4.2)$$

where each input image is of size $I \in R^{W \times H \times 3}$, the output dense feature representation map $F \in R^{\frac{W}{8} \times \frac{H}{8} \times C}$, and a *set* of attention maps $A \in R^{\frac{W}{8} \times \frac{H}{8} \times K}$, where W, H, C, K refer to the width, height, channels, and the number of landmark score maps respectively. Note here, each of the score maps (A's) is the results of *linear convolutions*. A global score map is also obtained by a max over the local landmark score maps.

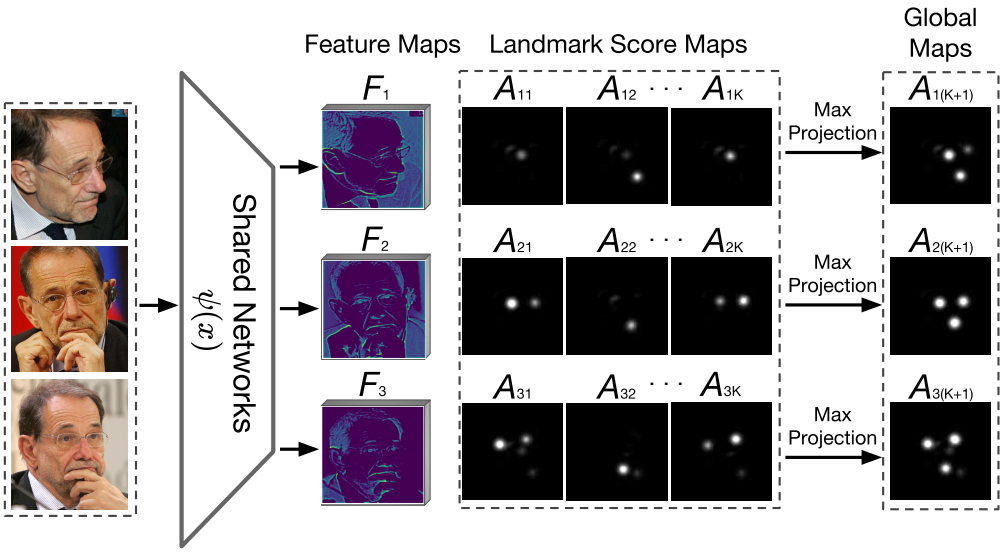


Figure 4.8: The detect module. For each input image the detect module generates an intermediate feature map ($F's$), K landmark attention maps ($A's$), and a global map (obtained by applying a max on the $A's$ channel dimension). In this example there are three input images and three of the K landmark attention maps are shown. .

Ideally, the local score maps for each image should satisfy two conditions, *first*, they should be mutally exclusive (i.e. at each spatial location only one landmark is activated); *second*, the scores on the maps should positively correlate with image quality, e.g. the response of a particular landmark filter should be higher on high-resolution frontal images than on low-resolution frontal images. Indeed the experimental results corrobate that this is so.

4.4.5 Attend

Dynamic Re-calibration (Internal Competition). Given the feature maps and landmark score maps for each input image, cross-normalization is used among

the score maps within same template for *dynamic re-calibration*. Based on the "quality" of images within the tower, the score maps (from different images within a single template) that localize same landmark are normalized as a distribution of weightings. Therefore, no matter how many images are packed into a template, the outputted attention maps in the same column always add up to 1.0 (Figure 4.7). Formally, for every $n \in [1, N]$ and $k \in [1, K]$:

$$A_{n..k} = \frac{exp(A_{n..k})}{\sum\limits_{nij} exp(A_{nijk})} \tag{4.3}$$

Attentional Pooling. With the re-calibrated attention maps for each input image, we next attend to the spatial locations and compute representations by image specific weighted averaging over the entire template. Formally, for each of the input image ($n \in N$), with the feature map as F_n, and one set of attention maps A_n,

$$V_k = \sum_{nij} F_{nij:} \odot A_{nijk} \qquad \text{for } k \in [1 : K + 1] \tag{4.4}$$

Therefore, for each input template, we are able to calculate $K+1$ feature descriptors (K landmark specific descriptors, "1" global feature descriptor), with each feature descriptor representing either one of the facial landmarks or global information.

4.4.6 Compare

Up to this point, we have described how to pool $K+1$ feature vectors from the single template. In this module, we compare these descriptors in pairs between two different templates. In detail, the landmark specific feature descriptors from two templates are first L2 normalized, and concatenated along with a one-hot encoded landmark identifier. Each concatenated vector is the input to a local "expert" modelled by a fully connected (FC) layers [76]. Overall, the local experts are responsible for comparing the landmark specific feature respones from different templates.

Formally, we learn a similarity function $y = C(x; \theta_2)$, where $x = [V_{1k} : V_{2k} : \text{ID}_{\textit{one-hot}}]$, as shown in Figure 4.7. After passing through the fully connected layers, the feature representations given by local "experts" are max pooled, and fused to provide the final similarity score.

Discussion. Unlike the approach of [76, 77], where features at every spatial location are compated with features at every other location, the compare module here only compares the descriptors that encode the same landmark, e.g. frontal mouth to frontal mouth. By attaching the landmark identifier (the one-hot indicator vector), the fully connected layers are able to specialize for each landmark.

4.5 Experiment

4.5.1 Network Architectures

In this part, we give the architectural details of the Comparator Network, covering the *Detect*, *Attend*, and *Compare* modules. As illustrated in the main in previous, the Comparator Network takes two templates as input, and for training each template contains 3 images ($N = 3$) of the same person.

In the *Detect* module, a standard ResNet is shared for every input image, and outputs feature maps and multiple landmark score maps. In the paper, we use $K = 12$ local landmarks. The global map is obtained as the max projection of the local landmark maps.

Module	Output Size (For each template)	Template 1 ($N \times 144 \times 144 \times 3$)	Template 2 ($N \times 144 \times 144 \times 3$)
Detect	$N \times 72 \times 72 \times 64$	conv, 7×7, 64, stride 2	
	$N \times 36 \times 36 \times 256$	max pool, 3×3, stride 2	
		$\begin{bmatrix} \text{conv}, 1 \times 1, 64 \\ \text{conv}, 3 \times 3, 64 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times 3$	
	$N \times 18 \times 18 \times 512$	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times 4$	
	$N \times 18 \times 18 \times 1024$ $N \times 18 \times 18 \times (K+1)$ K local landmark maps 1 global map	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \\ \text{conv}, 1 \times 1, 1024 \end{bmatrix} \times 1024, \begin{bmatrix} \text{conv}, 1 \times 1, K \end{bmatrix}$	

Table 4.1: *Detect* Module.

In the *Attend* module, the local landmark score maps from the same filter are first re-calibrated (cross normalization), and then are used to pool the landmark specific feature descriptors. The normalization of the landmark score maps serves two

important roles: first, low-quality images are effectively weighted down, therefore accomplishing the within-template weighting; second, it also makes it possible to ingest the template with an arbitrary number of images.

Module	Output Size (For each template)	Template 1 $N \times 18 \times 18 \times 1024$ $N \times 18 \times 18 \times (K+1)$	Template 2 $N \times 18 \times 18 \times 1024$ $N \times 18 \times 18 \times (K+1)$
Attend	$N \times 18 \times 18 \times 1024$ $N \times 18 \times 18 \times (K+1)$	Re-calibration	
	$(K+1) \times 1024$	Attentional pooling	
	$(K+1) \times 1024$	L_2 normalization	

Table 4.2: *Attend* Module.

In the *Compare* module, the landmark specific feature descriptors for each template are compared using local "experts", that are parametrized as fully connected layers with 2048 nodes.

Module	Output Size	Template 1 $(K+1) \times 1024$	Template 2 $(K+1) \times 1024$
Compare	$(K+1) \times (2048+K+1)$	Concatenate template descriptors & part ID	
	$(K+1) \times 2048$	Local experts (FC layer wiht 2048 nodes)	
	1×2048	Max pooling	
	1×2	Classification (1 or 0)	

Table 4.3: *Compare* Module.

4.5.2 Landmark Regularizers

In the *Attend* module, the landmark score maps can be considered as a generalization of global average pooling, where the spatial "weights" are inferred implicitly based on the input image. However, in the *Detect* module, there is nothing to prevent the network from learning K identical copies of the same landmark, for instance, it can learn to always predict the average pooling mask, or detect the eyes, or given a network with large enough receptive field, it can always pinpoint the centre of the image. To prevent this, we experiment with two different types of landmark regularizers: a diversity regularizer and a keypoints regularizer.

Diversity Regularizer In order to encourage landmark diversity, the most obvious approach is to penalize the mutual overlap between the score maps of different landmarks. We use the same method as [80]. Each of the landmark score maps is first self-normalized into a probability distribution (p's) by using the softmax (Eq 4.5),

$$p_{nijk} = \frac{exp(A_{nijk})}{\sum_{ij} exp(A_{nijk})} \tag{4.5}$$

where n, i, j, k refer to the image index within the template, width, height, number of attention maps respectively.

Ideally, if all K landmarks are disjoint from each other, by taking the max projection of these normalized distribution, there should be exactly K landmarks, and they should sum to K.

$$\mathcal{L}_{reg} = nK - \sum_{nij} \max_{k=1,..,K} \; p_{nijk} \tag{4.6}$$

Note here, this regularizer is zero only if the activations in the different normalized landmark score maps are disjoint and exactly 1.0.

Keypoints Regularizer Benifiting from the previous fruitful research in facial keypoint detection, pseudo groundtruth for the landmarks can be obtained from pre-trained facial keypoint detectors. Although the facial keypoint predictions are not perfect, we conjecture that they are sufficiently accurate to guide the network training at the early stages, and, as the training progresses, the regularizer weights is scheduled to decay, gradually releasing the parameter search space. As preprocessing, we predict 5 facial keypoints (Figure 4.9) over the entire dataset with a pre-trained MTCNN [64], and estimate three face poses by thresholding angle ratios.[1]

Similar to the diveristy regularizer, the inferred landmark score maps are also self-normalized first (Eq 4.5), and the $\mathcal{L}2$ loss between the prediction (p) and the pseudo groundtruth ($\hat{p}$) is applied as auxiliary supervision. Note that, given each

[1]In our training, we only use 4 facial landmarks, left-eye, right-eye, nose, mouth. The mouth landmarks are obtained by averaging the two landmarks at mouth corners.

Figure 4.9: Facial landmark detection for VGGFace2 images.
Face poses are quantized into three categories based on the ration α/θ. Left-facing profile : $\alpha/\theta < 0.3$, right-facing profile: $\alpha/\theta > 3.0$, frontal face: $\alpha/\theta \in [0.3, 3.0]$

face image belongs to only one of the three poses, only 4 of the 12 landmark map are actually useful for supervising an individual image.

$$\mathcal{L}_{reg} = \begin{cases} \sum_{nij} \frac{1}{2}(p_{nijk} - \hat{p}_{nijk})^2 \text{ for k in } \{\text{pose-specific keypoints}\} \\ \\ 0 \end{cases} \tag{4.7}$$

To make the experiments comparable, in both experiments, we use $K = 12$ landmark score maps in the *Detect* module.

4.5.3 Loss Functions

The proposed comparator network is trained end-to-end by optimizing three types of losses simultaneously: *first*, template-level identity classification soft-max loss, using a global feature representation obtained by attentional pooling with the re-calibrated global maps (refer to Figure 4.8); *second*, a standard classification loss (2 classes) on the similarity score prediction from the *Compare* module; *third*, a regularization loss from the landmark score maps in the *Detect* module.

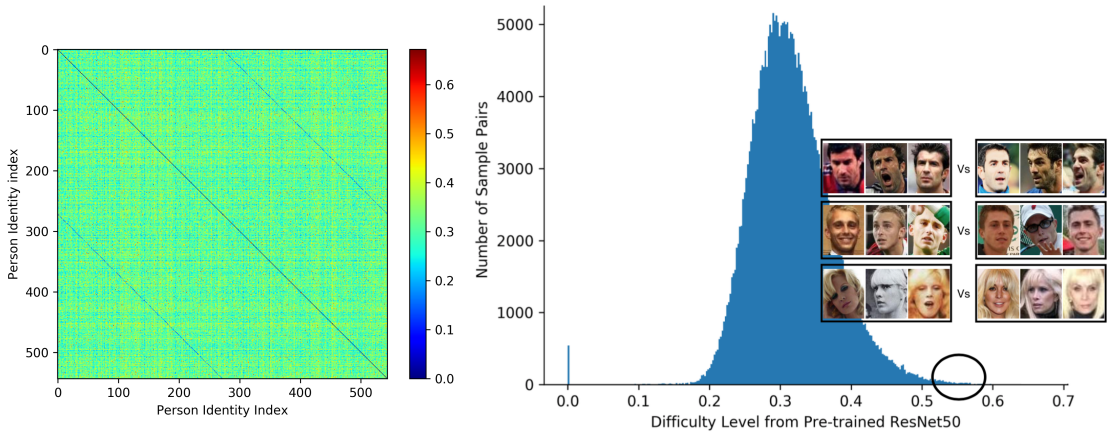
$$\mathcal{L} = \alpha_1(\mathcal{L}_{cls1} + \mathcal{L}_{cls2}) + \alpha_2\mathcal{L}_{sim} + \alpha_3\mathcal{L}_{reg} \tag{4.8}$$

where $\alpha_1 = 2.0, \alpha_2 = 5.0$ refer to the loss weights for classification and similarity prediction, α_3 refers to the weights for regularizer, which was initialized as 30.0 and decayed by half every $60,000$ iterations. Note that, α_3 is scheduled to decrease, thus, even for the training with the keypoints regularizer, the auxiliary supervision only guides the network training in early stages.

4.5.4 Hard-sample Mining

In order to train the Comparator Network for re-ranking we require a method to sample hard template-template pairs. Here we described the procedure for this. The key idea is to use the features generated by a standard ResNet-50 trained for face image classification (on the VGGFace2 training set) to approximate the template descriptor, and use this approximate template descriptor to select hard template pairs. Note, these templates are hard relative to the ResNet-50 trained for single image face classification. In detail, the template-level descriptors are obtained by averaging the feature vectors (pre-computed from ResNet-50) of 3 images and L2-normalized.

The selection of hard template pairs is then integrated into the training of the comparator network. At each iteration 256 identities are randomly sampled, and used to create 512 templates with 3 images in each template (i.e. two templates for each identity). In total, there are therefore 256 positive template pairs, and a large number of negative pairs.



(a) Confusion matrix (b) Sampling histogram

Figure 4.10: Sampling strategy based on the pre-trained single-image classification networks. Larger values refer to more difficult template pairs. The Comparator Network is trained by sampling template pairs at different levels.

By calculating the cosine similarity between different pairs of templates, we generate a 512×512 similarity matrix M_s for the template-to-template verification,

where small values refer to the predicted dissimilar pairs from the pre-trained ResNet50. We further define the verification difficulty matrix as:

$$d = |groundtruth - M_s| \tag{4.9}$$

where groundtruth label is either 0 (dissimilar) or 1 (similar). Therefore, in the difficulty matrix, the small values refer to the easy sample pairs, and the large values refer to the difficult samples. Based on this difficulty matrix, we are able to sample the training samples that cover the very difficult template pairs that the pre-trained ResNet-50 can not distinguish (Figure 4.10).

4.5.5 Training Details

We train the entire Comparator Network end-to-end from scratch on the VGGFace2 dataset. In the *Detect* module, a simple ResNet [35] with 40 layers is shared across the two towers, images are passed through until the width and height of the feature maps is 1/8 of the original resolution. During training, the shorter side of the input image is resized to 144, while the long side is center cropped, making the input images 144×144 pixels with the face centered, and 127.5 is subtracted from each channel. In each tower, 3 images are packed as a template input. Note that, there is a probability of 20% that the 3 images within one template are identical images[2]. In this case, the Comparator Network become equivalent to training on single image. Data augmentation is operated separately with probability of 20%, including flipping, gaussian blur, motion blur, monochrome transformation. Adam [24] was used for optimization with an initial learning rate as $1e^{-4}$, and mini-bateches of size 64, with equal number of positive and negative pairs. The learning rate is decreased twice with a factor of 10 when errors plateau. Note that, although the batch size is small, the network is actually seeing 64×6 images every training step. Also, although the network is only trained with 3 images per tower, at test time it can be applied to any number of images per template.

[2]This guarantees a probability of 64% that both templates contain 3 different images, and a probability of 36% that at least one template contains 3 identical image.

4.6 Result

We evaluate all models on the challenging IARPA Janus Benchmarks, where all images and videos are captured from unconstrained environments and show large variations in viewpoints and image quality. Note, in contrast to the traditional closed-world classification tasks (where the identities are the same during training and testing), verification is an open-world problem (i.e. the label spaces of the training and test set are disjoint), and thus challenges the capacity and generalization of the feature representations.

We evaluate the models on the standard 1:1 verification protocol (matching between the Mixed Media probes and two galleries), the performance is reported as the true accept rates (TAR) vs. false positive rates (FAR) (i.e. receiver operating characteristics (ROC) curve). During testing, we first report the results from only DCN, however, note that, in order to align with the re-ranking process during training, the DCN should ideally be used along with models trained with single image.

IJB-B Dataset [61] The IJB-B dataset is an extension of IJB-A [60], having $1,845$ subjects with 21.8K still images (including $11,754$ face and $10,044$ non-face) and 55K frames from $7,011$ videos.

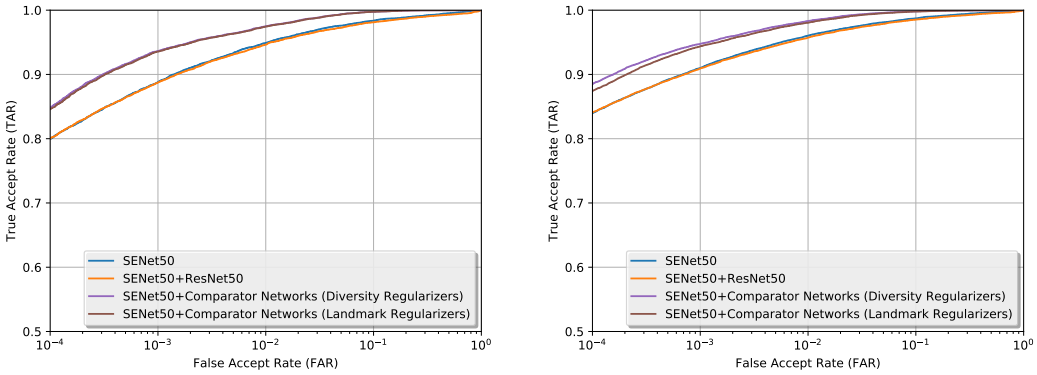
Model	1:1 Verification TAR			
	FAR=$1E-4$	FAR=$1E-3$	FAR=$1E-2$	FAR=$1E-1$
Whitelam *et al.* [61]	0.540	0.700	0.840	– –
Navaneeth *et al.* [81]	0.685	0.830	0.925	0.978
ResNet50 [13]	0.784	0.878	0.938	0.975
SENet50 [13]	0.800	0.888	0.949	0.984
DCNs(Kpts)	0.823	0.921	0.966	0.991
DCNs(Divs)	0.835	0.923	0.971	0.995
ResNet50+SENet50	0.800	0.887	0.946	0.981
ResNet50+DCN(Kpts)	**0.850**	0.927	0.970	0.992
ResNet50+DCN(Divs)	0.841	0.930	0.972	0.995
SENet50+DCN(Kpts)	0.846	0.935	0.974	0.997
SENet50+DCN(Divs)	0.849	**0.937**	**0.975**	**0.997**

Table 4.4: Evaluation on 1:1 verification protocol on IJB-B dataset. (Higher is better)
Note that the result of Navaneeth *et al.* [81] was on the Janus CS3 dataset.
DCN(Divs) : Deep Comparator Network trained with Diversity Regularizer
DCN(Kpts): Deep Comparator Network trained with Keypoints Regularizer.

IJB-C Dataset [66] The IJB-C dataset is a further extension of IJB-B, having 3,531 subjects with 31.3K still images and 117.5K frames from 11,779 videos. In total, there are 23124 templates with 19557 genuine matches and 15639K impostor matches.

Model	1:1 Verification TAR			
	FAR=$1E-4$	FAR=$1E-3$	FAR=$1E-2$	FAR=$1E-1$
GOTS-1 [66]	0.160	0.320	0.620	0.800
FaceNet [66]	0.490	0.660	0.820	0.920
VGG-CNN [66]	0.600	0.750	0.860	0.950
ResNet50 [13]	0.825	0.900	0.950	0.980
SENet50 [13]	0.840	0.910	0.960	0.987
DCNs(Kpts)	0.851	0.921	0.969	0.992
DCNs(Divs)	0.862	0.930	0.972	0.994
ResNet50+SENet50	0.841	0.909	0.957	0.985
ResNet50+DCN(Kpts)	0.867	0.940	0.979	0.997
ResNet50+DCN(Divs)	0.880	0.944	0.981	0.998
SENet50+DCN(Kpts)	0.874	0.944	0.981	0.998
SENet50+DCN(Divs)	**0.885**	**0.947**	**0.983**	**0.998**

Table 4.5: Evaluation on 1:1 verification protocol on IJB-C dataset. (Higher is better) Results of GOTS-1, FaceNet, VGG-CNN are read from ROC curve in [66].



(a) ROC for IJB-B (Higher is better) (b) ROC for IJB-C (Higher is better)

Figure 4.11: ROC curve of 1:1 verification protocol on IJB-B & IJB-C dataset.

4.6.1 Discussion

Three phenomena can be observed from the evaluation results: *first*, comparing the previous state-of-the-art model [13], the DCN trained by re-ranking can boost the performance significantly on both IJBB and IJBC (about 4-5%, which is a substantial reduction in the error); *second*, although the ResNet50 and SENet50 are

designed differently and trained separately, ensembles of them do not provide any benifit (the performance even drops slightly). This shows that the difficult template pairs for ResNet50 remains difficult for another more powerful SENet50, showing that the models trained on single image classification are not complementary to each other; while in contrast, the DCN can be used together with either ResNet50 or SENet50 to improve the recognition system; *third*, the performance of DCN trained with different regularizers are comparable to each other, showing that groundtruth of facial keypoints is not critical in training DCN.

4.7 Visualization

Figure 4.12 shows the attention maps for a randomly sampled template that contains multiple images with varying poses. Visualizing the maps in this way makes the models interpretable, as it can be seen what the landmark detectors are concentrating on when making the verification decision. The *Detect* module has learnt to pinpoint the landmarks in different poses consistently, and is even tolerant to some out-of-plane rotation. Interestingly, the landmark detector actually learns to localize the two eyes simultaneously; we conjecture, that this is due to the fact that human faces are approximately symmetric, and also during training, the data is augmented with horizontal flippings.

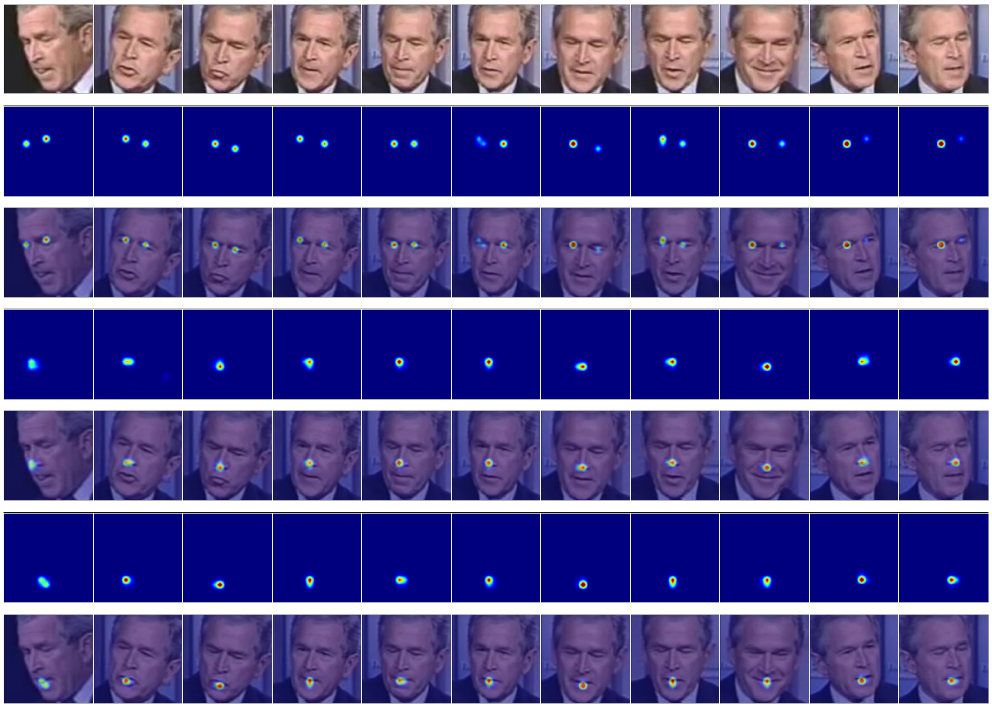


Figure 4.12: Predicted facial landmark score maps after self-normalizing for three of the landmark detectors. Additional examples are given in the supplementary material. *1st row*: raw images in the template, faces in a variaty of poses are shown from left to right; *2nd,4th,6th row*: self-normalized landmark score maps (attention maps); *3rd, 5th, 7th row*: images overlayed with the attention maps.

4.8 Discussion

We have introduced a new network that is able to compare templates of images to verify if they match or not, and demonstrated its use on face template verification. The network is very *flexible*, in that the number of images in each template can be varied at test time, it is also *opportunistic* in that it can take advantage of local evidence at test time, such as a specific facial features like a tattoo or a port-wine stain that might be lost in a traditional single tower per face encoding. Its performance substantially improves the state-of-the-art on the recent and very challenging IJB benchmarks.

Although we have used face templates in this work, the Comparator Network could also potentially be applied to other fine-grained classification, e.g. to determine the species of a bird or flower from multiple images of the same instance.

5 Microscopy Cell Counting & Detection

5.1 Introduction

Counting and detecting cells [82–85] in microscopy images is an extremely tedious and time-consuming task encountered in many real-world applications, for instance, a patient's health can be inferred from the number of red blood cells and white blood cells; in clinical pathology, cell counts from images can be used for investigating hypotheses about developmental or pathological processes; and cell concentration is important in molecular biology, where it can be used to adjust the amount of chemicals to be applied in an experiment. While detection on its own, is able to determine the presence (and quantity) of an object of interest, such as cancer cells in a pathology image, moreover, detection can also be used as seeds for further segmentation or tracking.

Recently, CNNs are starting to become popular in microscopy image analysis and have achieved state-of-the-art performance in several areas, such as mitosis detection [86], neuronal membranes segmentation [87], analysis of developing C. elegans embryos [88], and cell segmentation [44].

In this chapter, we first review the methods that treats counting and detection as a regression problem (§ 5.1.1, 5.1.2). In § 5.2, we propose to tackle the microscopy cell counting and detection with Fully Convolutional Regression Networks (FCRNs),

the developed methodology can equally be used for other counting or detection applications. Another issue that has been investigated in this chapter, is whether the networks trained solely on the synthetic data can generalize to real microscopy images (§ 5.3). The great advantage of this is that it avoids the problem of obtaining large data sets with manual annotation, where the data is limited, expensive, and time-consuming to collect and annotate. In § 5.5, we provide networks visualization by inverting the feature representations.

In the literature, the automatic cell counting was mostly approached from two directions, one is detection-based counting [82, 84, 89], which requires prior detection or segmentation; the other is based on density estimation without the need for prior object detection or segmentation [83, 85, 90].

5.1.1 Counting by Density Estimation

Cell counting in crowded microscopy images with density estimation avoids the difficult detection and segmentation of individual cells. It is a good alternative for tasks where only the number of cells is required. Over the recent years, several works have investigated this approach. In [90], the problem was cast as density estimation with a supervised learning algorithm, $D(x) = c^T \phi(x)$, where $D(x)$ represents the ground-truth density map, and $\phi(x)$ represents the local features. The parameters c are learned by minimizing the error between the true and predicted density map with quadratic programming over all possible sub-windows. In [85], a regression forest is used to exploit the patch-based idea to learn structured labels, then for a new input image, the density map is estimated by averaging over structured, patch-based predictions. In [83], an algorithm was proposed that allows fast interactive counting by simply solving ridge regression with various local features.

5.1.2 Detection by Regression

Although there has been much recent work on detection in *natural* images, there has been little application so far to *microscopy* images. Approaches on natural images include detections based on region proposal and classification networks [89,

91, 92], sliding window and classification networks [93], and using modes from heat map regression [94].

One work that has been developed to use CNNs for cell detection is [95]. In their work, they cast the detection task as a structured regression problem with the dot annotation near the cell centre. They train CNN model that takes an image patch of fixed size as input, and predicts a "proximity patch" of half the resolution of the original input patch. During training, the pre-defined proximity mask $\mathcal{M}$ corresponding to image I is calculated as,

$$\mathcal{M}_{ij} = \begin{cases} \frac{1}{1+\alpha D(i,j)} & \text{if } D(i,j) \leq \gamma, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where $D(i,j)$ represents the Euclidean distance from pixel (i,j) to the nearest manually annotated cell centre ($\alpha = 0.8$ and $\gamma = 5$ in their paper). Therefore, $\mathcal{M}_{ij}$ gives value 1 for the cell centre, and decreases with distance from the centre. During inference, in order to calculate the proximity map for an entire testing image, they propose to fuse all the generated proximity patches together in a sliding window way. After this, the cell detection is obtained by finding the local maxima in this average proximity map.

In contrast to these approaches, we focus on models that enable end-to-end training and prediction of density maps for images of arbitrary size by using fully convolutional regression networks (FCRNs). Cell counting and detection in the specific region of microscopy images can then be obtained simultaneously from the predicted density map.

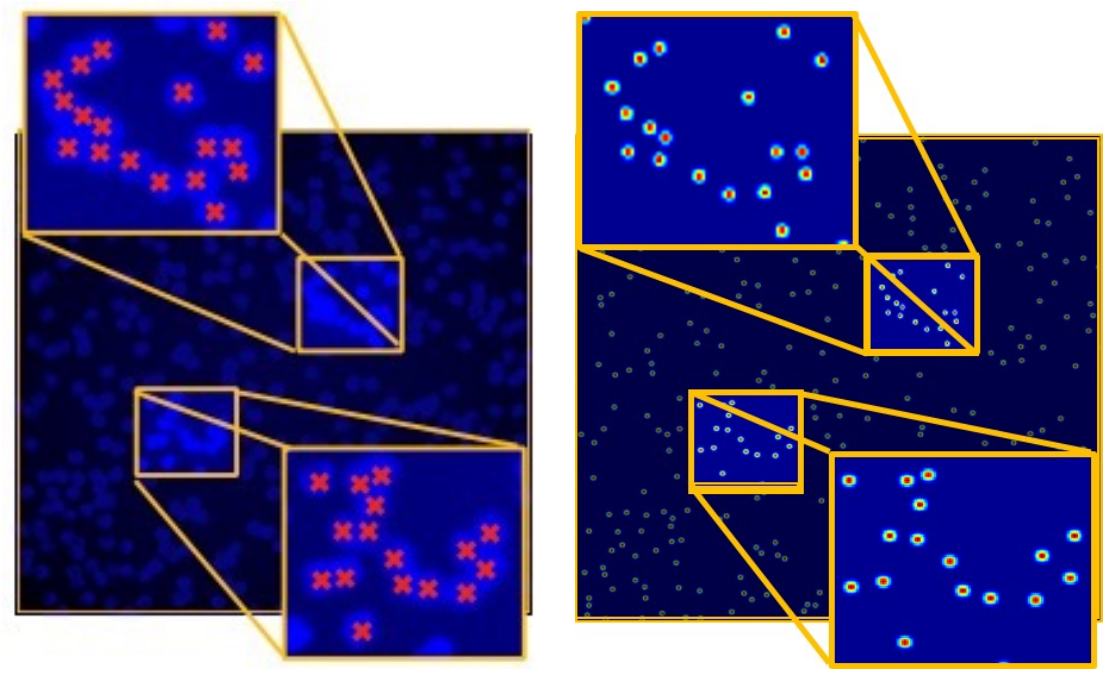
5.2 Method

Inspired by [90], we treat the cell counting and detection as a regression problem that aims to learn a function $f(\cdot)$, mapping an image $I(x)$ to a density map $D(x)$ (Figure 5.1):

$$D(x) = f(I(x); W) \tag{5.2}$$

where $I(x), D(x) \in R^{m \times n}$ and $W's$ refer to the trainable parameters.

To be specific, the ground-truth annotations are created with dots, where each is then represented as a Gaussian, and the density map $D(x)$ is formed by the superposition of these Gaussians. The central task is to regress this density map from the corresponding cell image $I(x)$, then the cell count in a specific region can be obtained by integrating over $D(x)$, and cell detection by local maxima detection on $D(x)$.



(a) Training Image: $I(x)$ **(b)** Density Map: $D(x)$

Figure 5.1: Training process.
During training, we aim to find a mapping between $I(x)$ and the density map $D(x)$.
(a) Red crosses on $I(x)$ are dot annotations near the cell centres.
(b) The density map $D(x)$ is a superposition of Gaussians at the position of each dot.
Integration of the density map $D(x)$ over specific region gives the count of cells.

5.2.1 Architecture Design

We proposed two fully convolutional networks, namely FCRN-A, FCRN-B, as shown in Figure 5.2. In both FCRNs, the number of feature maps in the higher layers is increased to compensate for the loss of spatial information caused by max pooling. In FCRN-A, all of the kernels are of size 3×3 pixels, and three max-poolings are used to aggregate spatial information leading to an effective receptive field of size

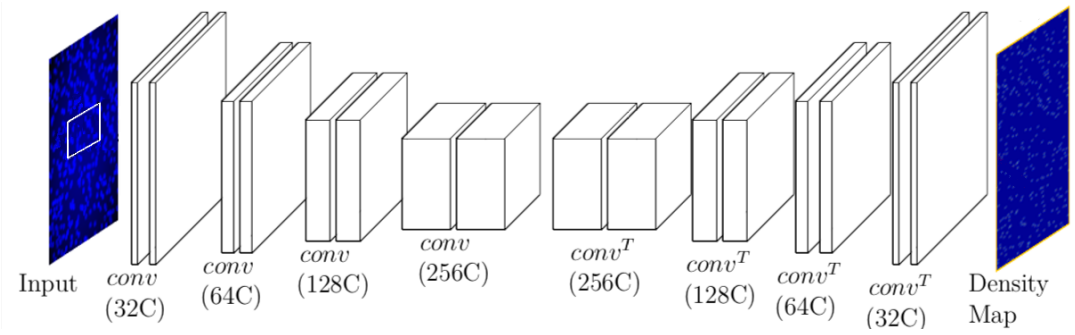


Figure 5.2: Architecture of Fully Convolutional Regression Networks (FCRNs) for cell counting & detection. Notation : *conv* : convolutional layer, $conv^T$: convolution transpose. $32C$ -> 32 channels.

38×38 pixels (i.e. the input footprint corresponding to each pixel in the output). FCRN-A provides an efficient way to increase the receptive field, while containing about 1.3 million trainable parameters. In contrast, max pooling is used after every two convolutional layers to avoid too much spatial information loss in FCRN-B. In this case, the number of feature maps is increased up to 256, with this number of feature maps then retained for the remaining layers. Comparing with FCRN-A, FCRN-B trains 5×5 upsampling kernels leading to the effective receptive field of size 32×32 pixels. In total, FCRN-B contains about 3.6 million trainable parameters, which is about three times as many as those in FCRN-A.

5.2.2 Training Details

During training, large images are cut into patches, for instance, patches of size 100×100 pixels are randomly sampled from 256×256 images. Simple data augmentation techniques are also used, e.g. small rotations, horizontal flipping. Before training, each patch is normalized by subtracting its own mean value and then dividing by the standard deviation.

The cost function is defined as :

$$l(X_0; W) = \frac{1}{M} \sum_{i=1}^{M} (Y_n - X_n^{(i)})^T (Y_n - X_n^{(i)}) \tag{5.3}$$

where W are all the trainable parameters, X_0 is the input patch, Y is the ground-truth density map with Gaussians of $\sigma = 2$, X_n is the predicted density map

for the input patch.

The parameters of the convolution kernels are initialized with an orthogonal basis [96]. Standard stochastic gradient descent (SGD) with momentum are used for optimization. Then the parameters w are updated by:

$$\triangle w_{t+1} = \beta \triangle w_t + (1-\beta)(\alpha \frac{\partial l}{\partial w}) \tag{5.4}$$

where β is the momentum parameter. The learning rate α is initialized as 0.01 and gradually decreased by a factor of 10. The momentum is set to 0.9, weight decay is 0.0005, and no dropout is used in either network. Since the non-zero region in the ground-truth density map only occupies very small amount of pixels, and even for non-zero regions, the peak value of a Gaussian with $\sigma = 2$ is only about 0.07, the networks tend to be overfitting to the zero regions. To alleviate this problem, Gaussian-annotated ground truth is simply sacled (Figure 1(b)) by a factor of 100, forcing the network to fit the Gaussian shapes rather than background zeros.

After pre-training with patches, we further fine-tune the parameters with the whole images to smooth the estimated density map, since the 100×100 image patches sometimes may only contain part of a cell on the boundary.

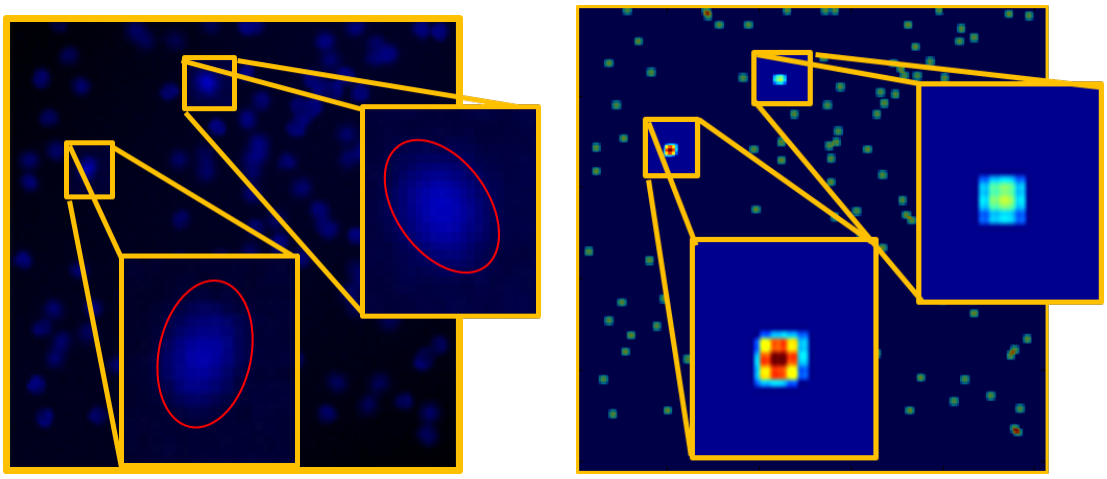
5.3 Experiment

In this section, FCRN-A and FCRN-B are first compared with previous work on cell counting using synthetic data. Then the network trained only on synthetic data is applied to a variety of real microscopy images without fine-tuning. Finally, comparison of the performance before and after fine-tuning will be presented on real microscopy images, this fine-tuning can be seen as a re-calibration process.

In terms of cell detection, they are obtained simply by taking local maxima based on our predicted density map. Detection results are also shown both for synthetic data and real microscopy images.

5.3.1 On Synthetic Data

The synthetic dataset [90] consists of 200 images of cell nuclei on fluorescence microscopy generated with (Lehmussola et al., 2007). Each synthetic image has an average of 174±64 cells. Severe overlap between instances are often observed in this dataset, which makes it challenging for counting or detection. As shown in Figure 5.3, under this situation, it even becomes impossible for a human expert to tell the difference between overlapping cells and a single cell. The synthetic dataset is divided into 100 images for training and 100 for testing, and several random splits of the training set are used. Such splits consist of five sets of N training images and N validation images, for N = 8, 16, 32, 64. The mean absolute errors and standard deviations for FCRN-A and FCRN-B are reported.



(a) Training Image: $I(x)$ **(b)** Density Map: $D(x)$

Figure 5.3: Annotation noise.
(a) The image from a standard synthetic dataset. For reader convenience, the rough boundaries of the cells have been manually drawn with a red ellipse.
(b) Put a Gaussian at the centre of each generated cell.
The highlighted *upper-right* region contains one single cell, while the *lower-left* region actually contains two overlapping cells.

5.3.2 On Real Data

FCRN-A and FCRN-B are also applied to predict on four different kinds of data; (1) retinal pigment epithelial (RPE) cell images. The quantitative anatomy of RPE can be important for physiology and pathophysiology of the visual process, especially in

evaluating the effects of aging [97]; (2) embryonic stem cells. Cell counting is essential to monitor the differentiation process [98]; (3) plasma cell. The relative number of plasma cells in a bone marrow specimen is a clinical parameter important in the diagnosis and management of plasma cell dyscrasia [99]; (4) images of precursor T-Cell lymphoblastic lymphoma. Lymphoma is the most common blood cancer, usually occurs when cells of the immune system grow and multiply uncontrollably.

5.4 Result

5.4.1 On Synthetic Data

During testing, each image is mapped to a density map first, then integrating over the map for a specific region gives the count, or taking local maxima gives the cell detection of that region (Figure 5.4). The performances of the two networks for cell counting are compared in Table 5.1 as a function of the number of training images. As shown in Table 5.1, FCRN-A performs slightly better than FCRN-B. The *size* of the receptive field turns out to be more important than being able to provide more detailed information *over* the receptive field, it is hypothesized that this is because the difficulty in cell counting lies in regression for large cell clumps, and a larger receptive field is required to span these. For both networks, the performance is observed to improve by using more training images from $N = 8$ to $N = 32$, and only a small additional increase for N to 64. FCRN-A shows about 9.4% improvement over the previous best method of [85] when $N = 32$.

In terms of the errors, we conjecture that there are mainly three key sources: first, from the dataset itself. As shown in Figure 5.3, the annotation for the dataset itself is noisy. In this case the L2 regression loss tends to over-penalize. second, from the boundary effect due to bilinear up-sampling. Cells on the boundary of images tend to produce slightly wrong predictions in this case; and the third source of error is from very large cell clumps, where four or more cells overlap. In this case, larger clumps can be more variable in shape than individual cells and so are harder to regress; furthermore, regression for large cell clumps requires the network to have an even larger receptive field that can cover important parts of the entire

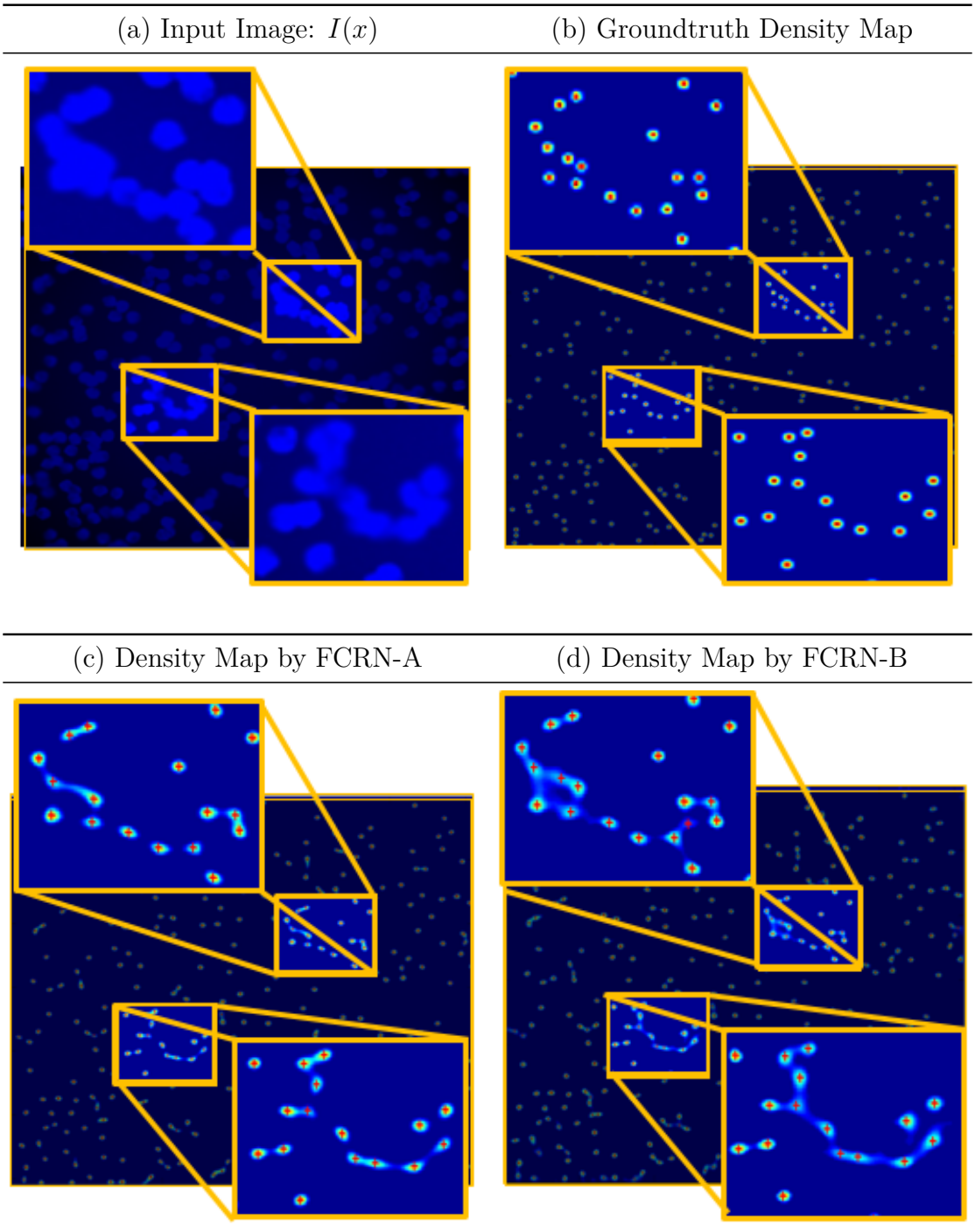


Figure 5.4: Counting inference process for FCRNs.
(a) Input image from test set.
(b) Ground-truth density map. **Count**: 18 (Upper-left), 16 (Lower-right).
(c) Estimated density map from FCRN-A. **Count**: 17(Upper-left) 16(Lower-right).
(d) Estimated density map from FCRN-B. **Count**: 19(Upper-left) 16(Lower-right).
Red crosses on (c) and (d) indicate cell detection results.

Method	174±64 cells			
	N=8	N=16	N=32	N=64
[90]	8.8±1.5	6.4±0.7	5.9±0.5	N/A
[90]	4.9±0.7	3.8±0.2	3.5±0.2	N/A
[85]	**3.4±0.1**	N/A	3.2±0.1	N/A
[83]	4.5±0.6	3.8±0.3	3.5±0.1	N/A
Proposed FCRN-A	3.9±0.5	**3.4±0.2**	**2.9±0.2**	**2.9±0.2**
Proposed FCRN-B	4.1±0.5	3.7±0.3	3.3±0.2	3.2±0.2

Table 5.1: Cell counting results on synthetic dataset.
The columns correspond to the number of training images. Standard deviation corresponds to five different draws of training and validation sets.

clumps, like concavity information, or curved edges in specific directions. Since both FCRNs are relatively shallow and only have a receptive field of size 38×38 pixels and 32×32 pixels, for elongated cell clumps, their curved edges can usually be covered, and correct predictions can be expected. However, for a roughly rounded cell clump with four or more cells, it can be bigger than our largest receptive field, and this usually leads to an incorrect prediction.

5.4.2 On Real Data

Both regression networks are applied on real datasets for detection and counting. Here, results from FCRN-A are shown in Figure 5.5, 5.6, 5.7 (without fine-tuning) and Figure 5.8 (before and after fine-tuning). During fine-tuning, two images of size 2500×2500 pixels, distinct from the test image, are used for fine-tuning pre-trained FCRNs in a patch-based manner, the same annotations following Figure 1b are performed manually by one individual, each image contains over 7000 cells. It can be seen that the performance of FCRN-A on real images can be improved by fine-tuning, reducing the error of 33 out of 1502 (before fine-tuning) to 17 out of 1502 (after fine-tuning).

When testing FCRN-B on two datasets of real microscopy data, for RPE cells: Ground-truth / Estimated count = 705 / 699, and for Precursor T-Cell LBL cells: Ground-truth / Estimated count = 1502 / 1473 (Without fine-tuning). Surprisingly, FCRN-B achieves slightly better performance on real data than FCRN-A. We

conjecture that the real data contains smaller cell clumps than synthetic data, therefore, the shape of cell clumps will not dramatically. The network is then able to give a good prediction even with a small receptive field.

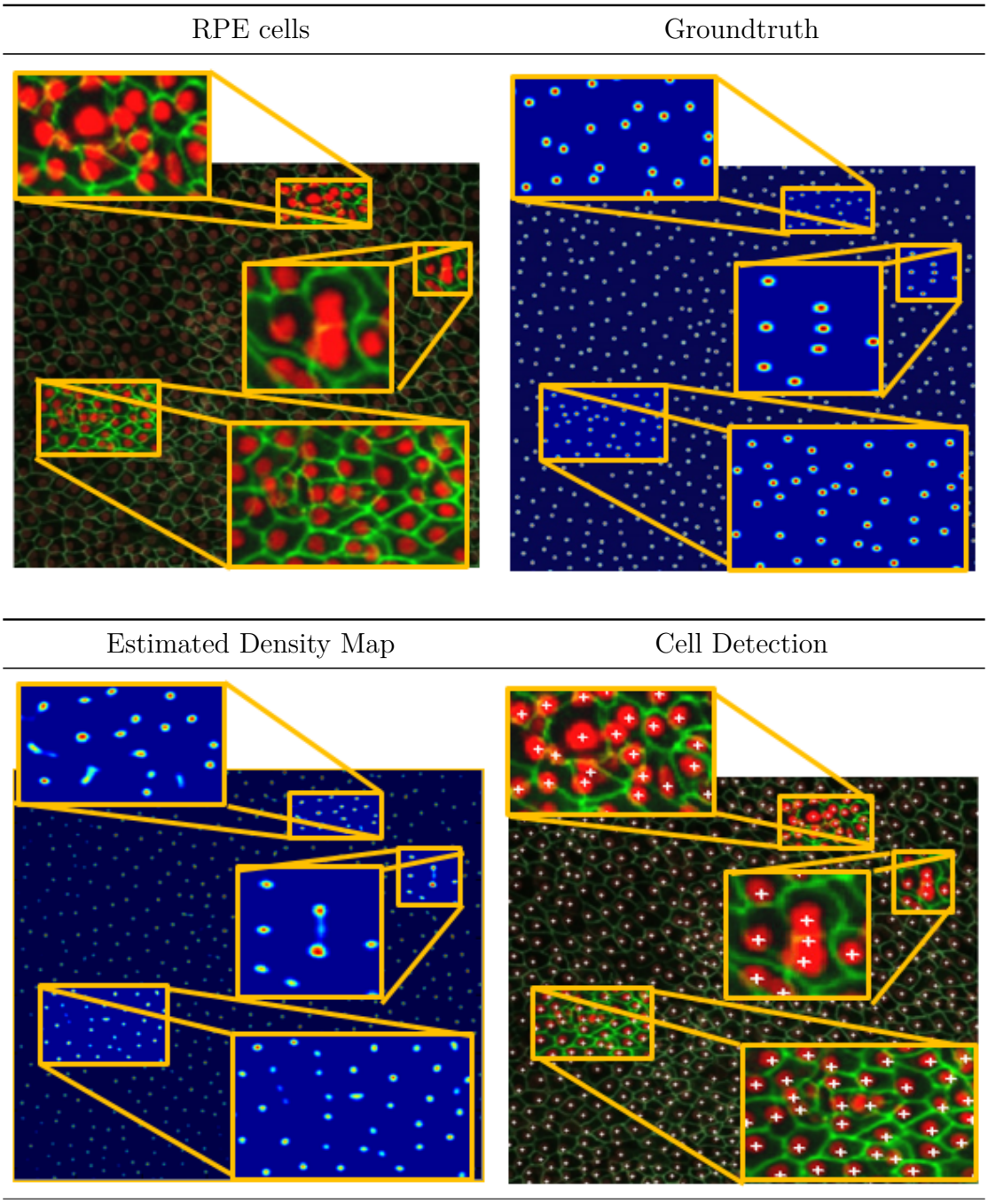


Figure 5.5: FCRN-A applied on retinal pigment epithelial (RPE) cells.
Only one channel is used for counting and detection.
Cell count: Ground-truth vs. Estimation : 705 / 697

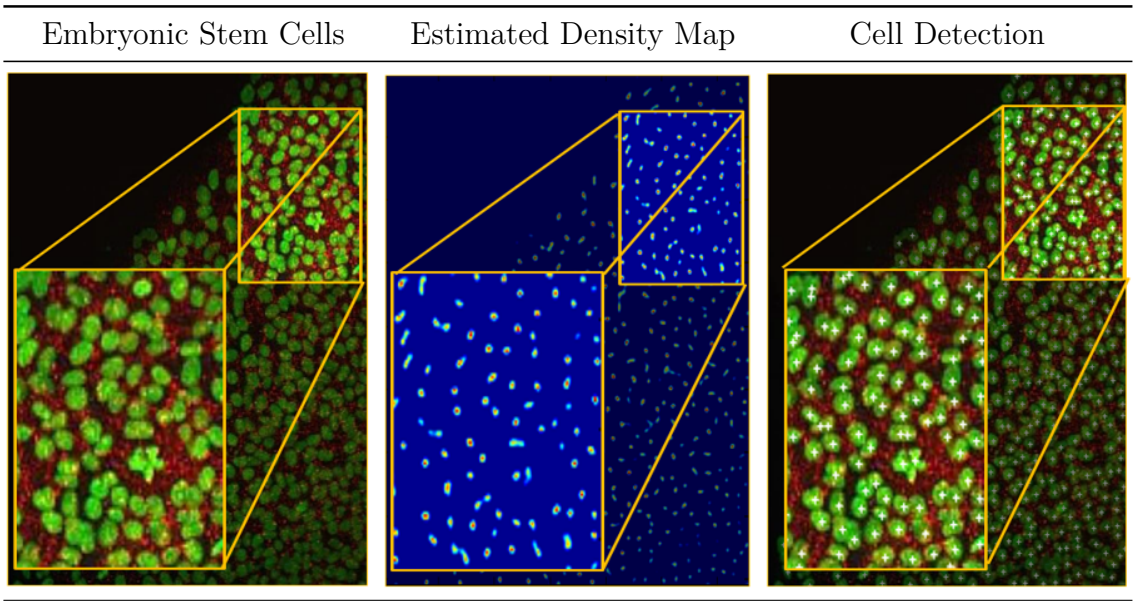


Figure 5.6: FCRN-A applied on embryonic stem cells.
Cell Count: Ground-truth vs. Estimated : 535 / 530

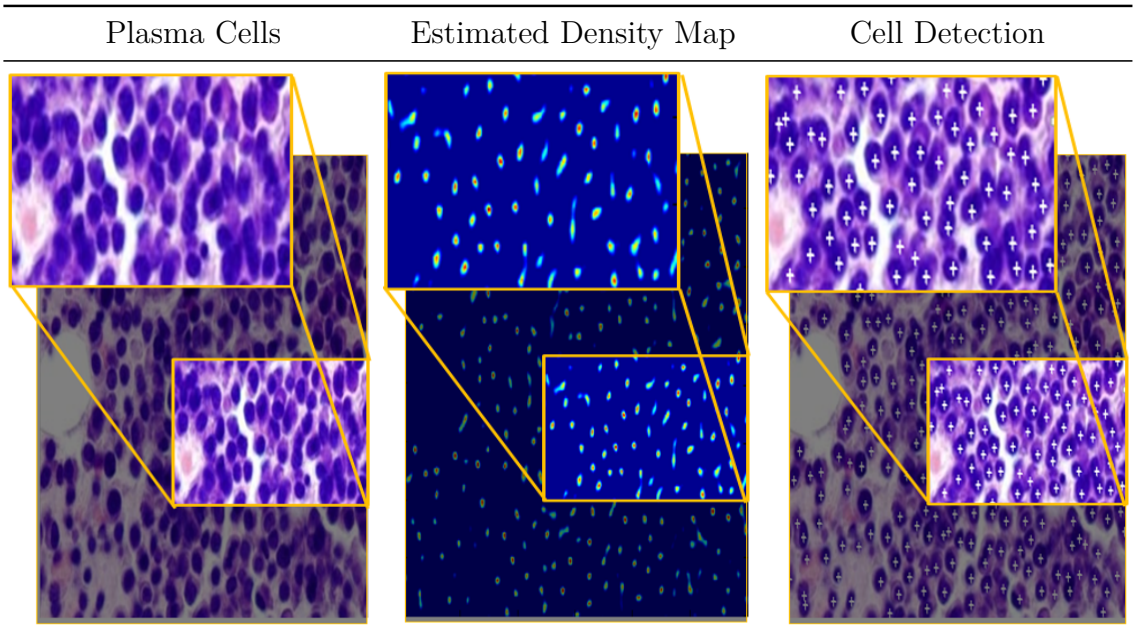


Figure 5.7: FCRN-A applied on plasma cells.
Cell Count: Ground-truth vs. Estimated : 297 / 294

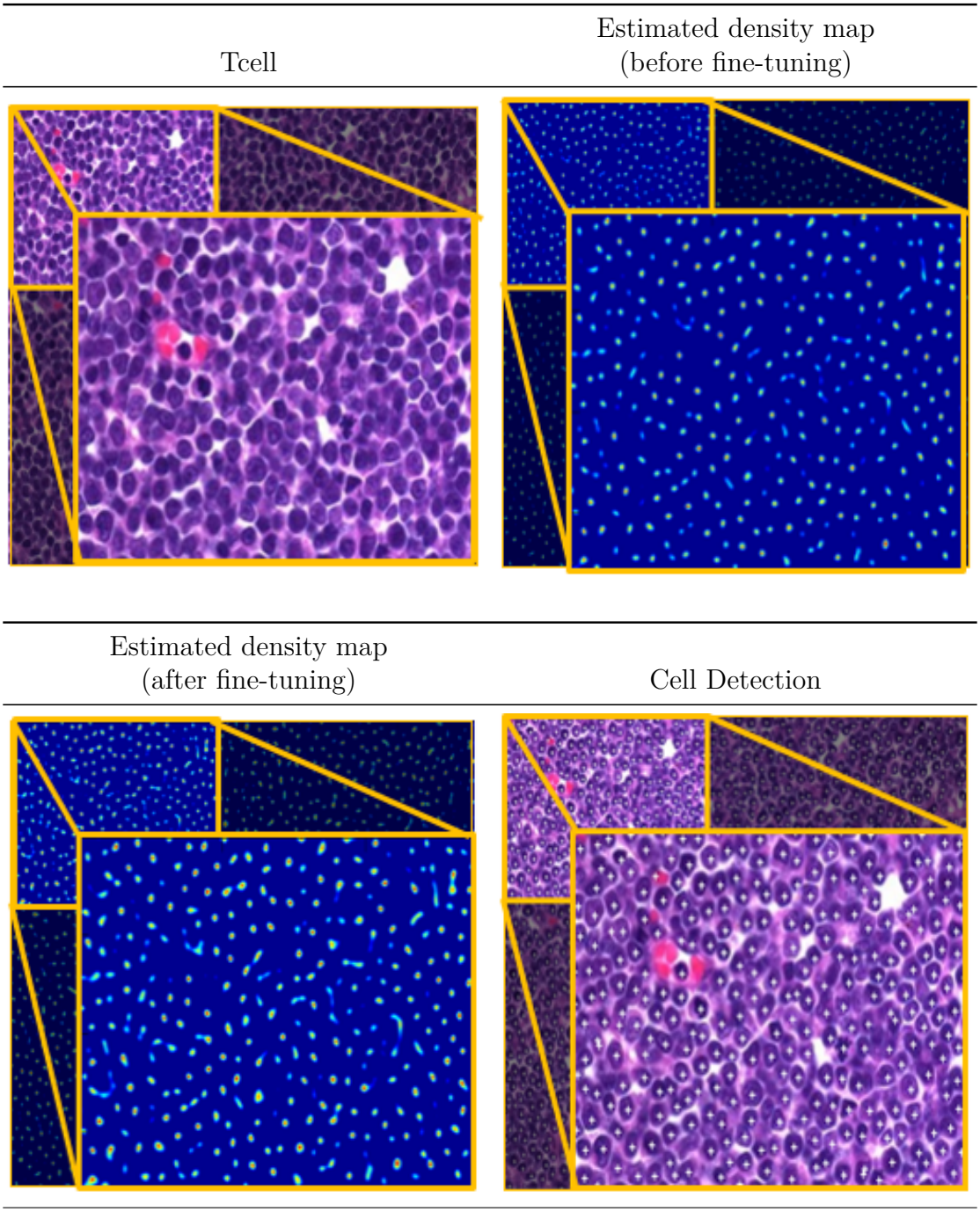


Figure 5.8: FCRN-A applied on precursor T-cell LBL.
Only one channel is used for counting and detection.
Cell count: Ground-truth vs. Before fine-tuning : 1502 / 1469
Cell count: Ground-truth vs. After fine-tuning : 1502 / 1485

5.5 Visualization

In order to understand the feature representations that have been captured by the deep networks, we consider the following question: "given an encoding of an image, to what extent is it possible to reconstruct that image"? In other words, the goal is to visualize how much information of the input image has been captured by the feature representations of different layers in the deep networks [100].

5.5.1 Inverting Representations

The problem scenario is therefore formalized as a reconstruction problem (Figure 5.9). Given a representation function $F : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^d$ and a representation $\phi_0 = \phi(x_0)$ to be inverted, the reconstruction process aims to find another image $x \in \mathbb{R}^{H\times W\times C}$ that minimizes the objective:

$$x^* = \underset{x\in\mathbb{R}^{H\times W\times C}}{\operatorname{argmin}} \; l(\phi(x), \phi_0) + \lambda L_2(x) \tag{5.5}$$

$$l(\phi(x), \phi_0) = \|\phi(x) - \phi_0\|^2 \tag{5.6}$$

where the loss l compares the image representation $\phi(x)$ to the target ϕ_0, and L_2 penalty is used to penalize large pixel values.

5.5.2 Optimization

Similar to training deep networks, simple gradient descent (GD) algorithms with *momentum* is used to optimize the reconstruction loss :

$$\begin{aligned} \Delta x_{t+1} &:= \beta \Delta x_t - \eta_t \nabla E(x) \\ x_{t+1} &:= x_t + \Delta x_t \end{aligned} \tag{5.7}$$

where $E(x) = l(\phi(x), \phi_0) + \lambda L_2(x)$ is the objective function, weight decaying factor $\beta = 0.9$, and learning rate η_t is gradually reduced until convergence.

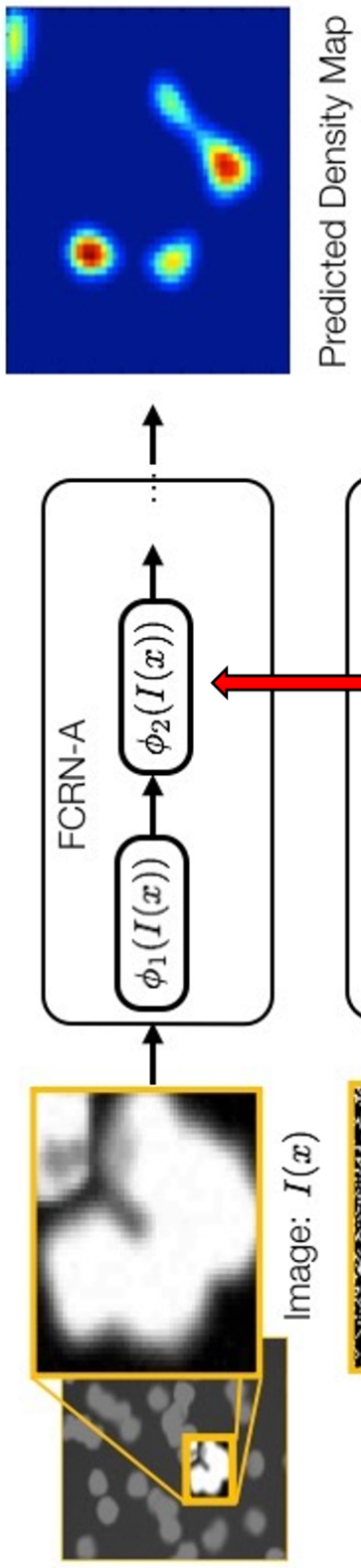


Figure 5.9: Inverting feature representation based on layer ϕ_2.
Step 1: Feed an input image $I(x)$ to the pre-trained FCRN-A, and make a record of the feature representations $\phi_2(I(x))$.
Step 2: Feed a random input image $I'(x)$ to FCRN-A, similarly, calculate feature representations $\phi_2(I'(x))$.
Step 3: Optimize the random input image $I'(x)$ with gradient descent (GD), such that $\phi_2(I(x)) = \phi_2(I'(x))$.

5.5.3 Reconstruction Results

For simplicity, we only show the visualization results from FCRN-A, but the same procedure can be performed for FCRN-B as well. In essence, CNNs are initially designed as an hierarchical model, which aim to extract more semantic information as the networks get deeper. For the density map prediction tasks, the biggest challenge is caused by the highly overlapping cell clumps of various shapes. In Figure 5.10, it shows to what extent the information from original cell clump has been encoded by the feature responses of different layers, and try to present an intuition about how the predictions are done by these FCRNs.

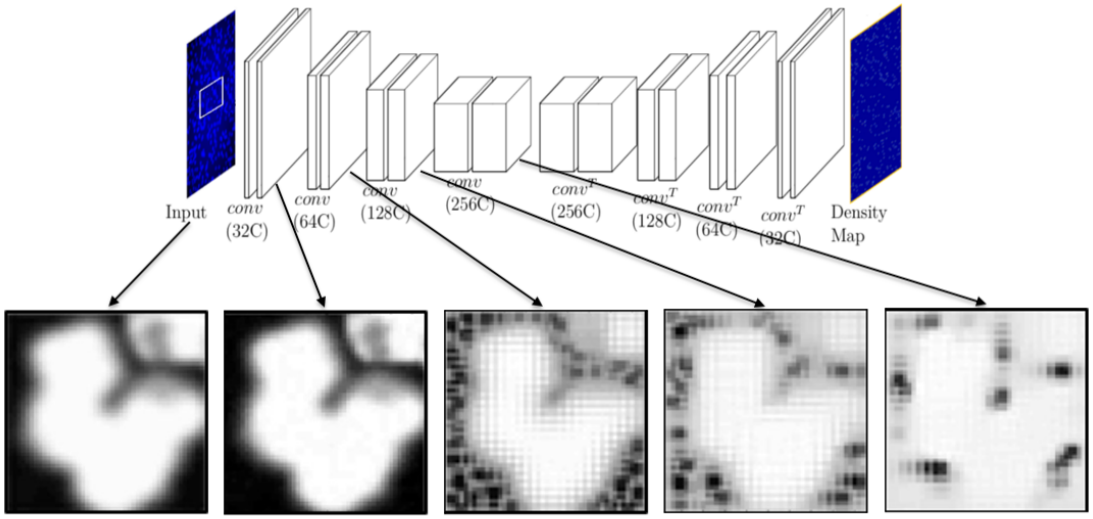


Figure 5.10: Image reconstruction from feature representations.

As shown in Figure 5.10, when the networks get deeper, feature representations for this cell clump become increasingly abstract. Therefore, when trying to reconstruct the input image from those feature representations, reconstruction quality decreases with the depth of networks, and only important information has been kept by deep layers, for instance, in **Conv3** (third convolutional block), edges around the cell clump are captured, and for **Conv4** (fourth convolutional block) that contains most abstract information in this network, only concavity information has been kept for prediction.

5.6 Discussion

This chapter has proposed to tackle cell counting and detection with Fully Convolutional Regression Networks (FCRNs). By learning a function mapping from image to density map, cell counts are calculated by integreting over the region, and detection can be obtained by taking the local maxima of the predicted density map. Moreover, we show that the networks trained on synthetic dataset can generalize to the real data, and further be improved by finetuning. Finally, visualizations of the pre-trained networks show that, as the networks get deeper, feature representations are getting increasingly abstract, while the concavity information of the cell clump is always kept.

6

Standardized Reorientation in 3D Fetal Neurosonography

6.1 Introduction

Fetal neurosonography has improved significantly in the last few decades, and it is emerging as a clinically useful imaging technology for assessing brain development and detecting cerebral abnormalities in the womb, which has applications in settings where expensive magnetic resonance imaging (MRI) is unavailable or not well-suited. Regardless of imaging modality, fetal brain localization and geometric alignment are the primordial steps for neuroimage analysis. This analysis relies on: (*i*) initial localization of the brain, (*ii*) removal of extracranial and maternal tissues, and (*iii*) alignment of the region of interest to a referential coordinate system. Establishing a coordinate system over the fetal brain serves as a precursor for several applications, such as localization of anatomical landmarks, extraction of standard clinical planes for biometric assessment of fetal growth, and extraction of oblique planes to study the evolution of image-based biomarkers from womb to cot.

Indeed, 3D fetal neurosonography has also been shown to capture neuro-developmental image signatures from which to predict gestational age and brain maturation, as demonstrated observationally.

However, processing these data remains a challenging task due to interactions between the skull and the ultrasound (US) signal. Specifically, as the density of the skull increases, it blocks the signal from penetrating to deep-seated tissues and creates strong acoustic shadows. The concave shape of the skull also creates shadows that limit clear structural visibility to only *one* of the cerebral hemispheres (located in the lower half of the image). These factors complicate the design of a neurosonography-specific coordinate reference, but these high-intensity structures are also the most salient cues for alignment.

In this chapter, we propose an automated tool which capitalizes solely on sonographic image signatures to achieve an standardized reorientation of the fetal brain at any gestational time point (Figure 6.1). The pipeline normalizes image volumes to a reference space through estimated affine transformations. As such, in § 6.2, we proposed to tackle the 3D standardized reorientation by decomposing the complex similarity transformation into several simple ones, which can be easily tackled with multi-task CNNs. The model leverages domain specific information to train these closely related tasks by sharing low-level features in the early layers, before branching into independent output streams for each task. The task-specific predictions are then combined to produce a transformation matrix as the desired final output. In § 6.3, we experiment six different networks architectures by varying the depth, kernel size, etc. In § 6.4, we show that the model is generalizable to a wide gestational age range, and does not require additional age or fetal size input to achieve head segmentation, eye localization, and prediction of brain orientation.

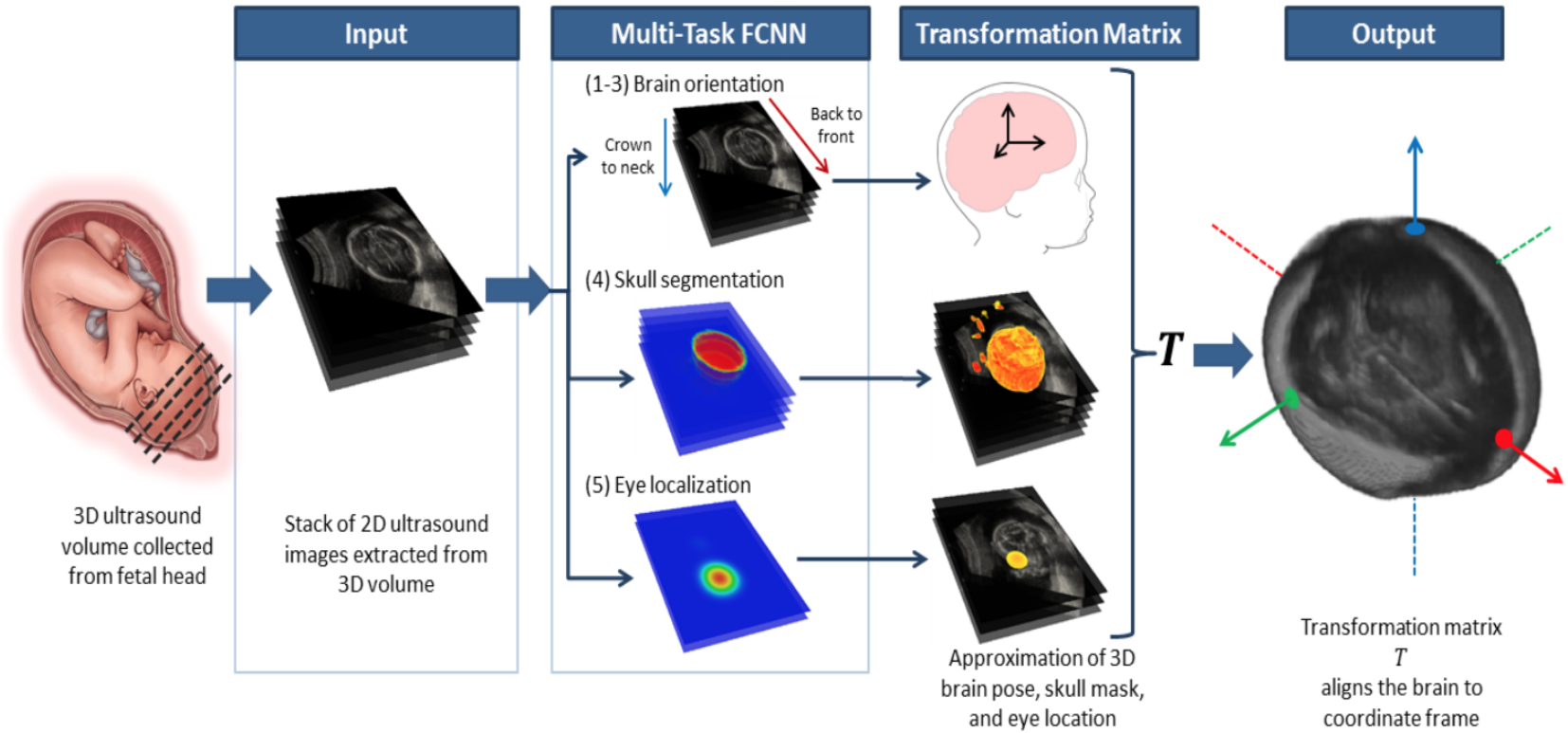


Figure 6.1: Pipeline for automated alignment of fetal 3D neurosonography to a canonical coordinate space Each slice extracted from the volume is classified and segmented by the multi-task network. The predicted labels are combined the estimate a transformation matrix T that aligns the volume to a pre-defined reference space.

6.2 Method

Fetal neurosonography scans are typically acquired following a pre-defined protocol set by international standards. In the case of the ISUOG guidelines, the acquisition plane of the US probe should be orthogonal to the midsagittal plane (or falx plane, defining the falx cerebri), and the axial plane traversing the centre of the brain should be located in the middle of the US volume. Given such a pre-defined standard acquisition protocol, the approximate geometry of image acquisition is assumed to be known. Specifically, the US probe is assumed to be positioned such that the head is imaged axially, and the insonation plane is perpendicular to the midsagittal plane (which separates the cerebral hemispheres. It is also expected that the skull appears as the most echogenic oval-shaped structure, occupying at least 50% of the image space. This information has efficiently been used to perform brain masking, and guide the prediction of brain orientation.

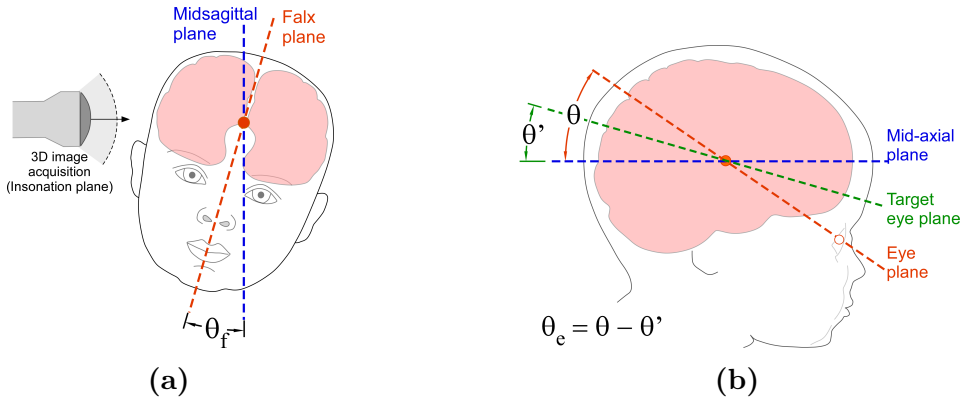


Figure 6.2: Data acquisition.
(a) Schematic of probe position for fetal neurosonography. Insonation plane is expected to be orthogonal to the midsagittal plane ($\mathcal{P}_o^s$) in the sagittal view. The falx plane ($\mathcal{P}_f$) must be rotated by angle θ_f to align to the midsagittal plane.
(b) Forward tilt correction is achieved by aligning the fetus' eye plane ($\mathcal{P}_e$) with the target eye plane by rotating by angle θ_e.

Although the location of the head can be assumed from the acquisition protocol, the geometric orientation of the brain regions relative to the coordinate axes remains unknown. Due to interactions between the US signal and tissue boundaries, the sonographic appearance of brain structures depends on the insonation plane, and is thus informative of overall orientation. The appearance of an individual

2D axial slice differs from its adjacent slices as a function of spatial distance (Figure 6.3). That is, the anatomies seen in the slices at the base of the brain (near the neck) have a different spatial configuration from those near the crown. The proposed model relies on this appearance-based ordering of slices, the unlikely sudden transition between superior and inferior classes, and the low similarity between them to achieve a prediction of head pose. Furthermore, axial slices collected at the level of the thalami contain structures which are arranged in a consistent *spatial configuration*, clearly distinguishing anterior from posterior regions. Regardless of gestational age, the structures observable in these slices are generally reliable and consistent in terms of echobrightness.

Thus, the brain re-orientation is treated as a transformation which relates the data volume to a 3D common coordinate space. The problem is defined as a 3D affine transformation, $\mathbb{T}$, composed of the product of four individual sub-transforms,

$$\mathbb{T} = \mathbb{M}_e \mathbb{M}_o \mathbb{M}_s \mathbb{M}_a \tag{6.1}$$

Here, $\mathbb{M}_e$ encodes the rotation matrix which refines brain alignment by using eye location to correct for forward brain tilt (Figure 6.2b). $\mathbb{M}_o$ corresponds to the rotation and translation of the imaged falx plane ($\mathcal{P}_f$) onto the common coordinate frame, as shown in Figure 6.2a. $\mathbb{M}_s$ defines the transformation required to flip (or reflect) the brain, such that it is correctly aligned in the superior-to-inferior direction of the coordinate axes, and lastly, $\mathbb{M}_a$ corresponds to the anterior-posterior reflection matrix (e.g. ensuring that all brains are 'forward-facing').

6.2.1 Multitask Model

In order to estimate the geometric transformation that maps the 3D brain volume into a common coordinate space, the complex problem is decomposed into several simple ones, that can be easily tackled with CNNs. A deep multi-task network is trained for three classification tasks, and two segmentations tasks that produce maps with binary units at each pixel location (m, n). A stack of 2D axial slices are sampled from a 3D volume, spanning from the crown to the neck regions of the fetal

head. Each 2D image is then passed into the network to predict whether the image is located near the superior or inferior regions of the head (task 1). The model simultaneously determines the anteroposterior direction of the fetal head (task 2), and the presence or absence of the eye socket (task 3). The segmentation tasks are such that task 4 aims to localize the eye socket, and task 5 segments the skull.

To summarize, the proposed network (Figure 6.5) takes a 2D axial slice as input and outputs task-specific labels $\{y_1, y_2, y_3, \mathbb{Y}_4, \mathbb{Y}_5\}$, where labels are from a finite set: $y_{\{1,2,3\}} \in \{0, 1, 2\}$, and $\mathbb{Y}_{\{4,5\}} \in \{0, 1\}$. Figure 6.3 shows the slice annotations for each task. More specifically, $y_1 \in$ {ignore, inferior, superior}, $y_2 \in$ {ignore, posterior, anterior}, and $y_3 \in$ {ignore, absent, present}.

Note that, y_k are 3-way classification labels where $y_k = 0$ corresponds to the 'ignore' class. Intuitively, by adding an 'ignore' label in all classification tasks, the multi-task problem enables the sequential prediction. In contrast, $\mathbb{Y}_k \in \mathbb{R}^{m \times n}$ are heat maps, indicating the presence probability of structures. Therefore, the corresponding ground-truth (GT) label for $\mathbb{Y}_4$ is annotated with binary values indicating the voxels of an eye socket (Figure 6.3d). It is, however, worth noting that in standard 3D fetal US data, only the eye socket nearest to the US probe is clearly visible due to reverberation artefacts and fetal position in relation to the probe. As such, the GT eye mask represents the annotation of only one of the fetal eye sockets. For $\mathbb{Y}_5$, another GT binary mask is produced for each slice to indicate the image region within the skull boundary (foreground), or extracranial regions such as maternal and uterine tissues (background), as depicted in Figure 6.3e. We postpone the description of how the output of these five tasks is used to compute the transformation $\mathbb{T}$ until § 6.2.2 (Estimation of Transformation), and turn next to the design of the multi-task network.

Architecture Design Concurrent to the recent DenseNet ([101]), the basic module in our proposed network extensively uses skip layers to fuse feature representations at multiple scales by concatenation (Figure 6.4). In order to build a multi-task learning architecture, the model is subdivided into two parts:

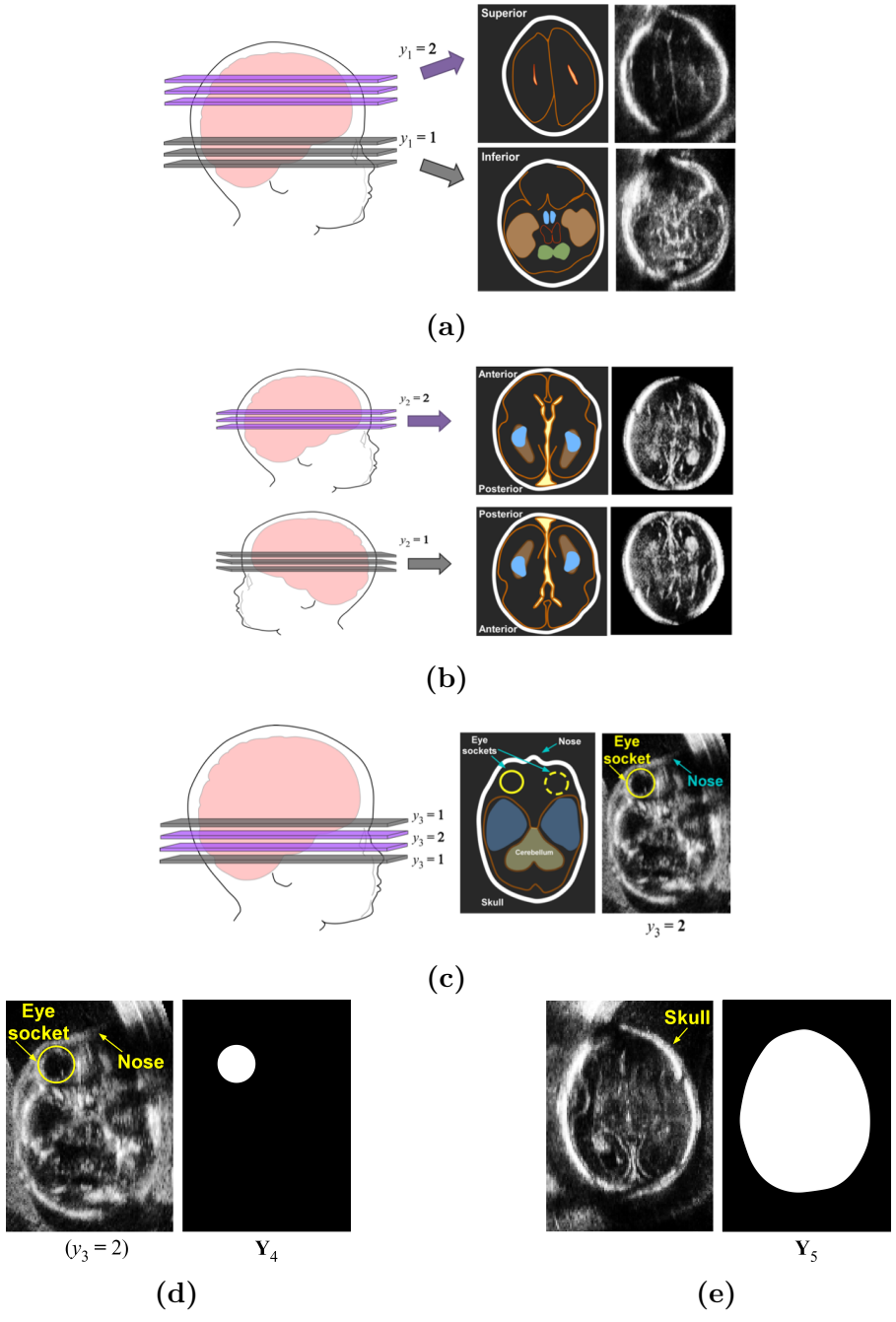


Figure 6.3: Annotation description. Schematic of a set of axial slices extracted from a 3D brain volume, spanning the cerebrum. For each task k, respectively, the slices to be considered for classification (red: $y_k = 2$, blue: $y_k = 1$), and the slice regions to ignore ($y_k = 0$) are labelled with a cross.
(a) y_1: Superior (annotated as 2) vs Inferior (annotated as 1);
(b) y_2: Anterior (annotated as 2) vs Posterior (annotated as 1);
(c) y_3: Eye socket Present (annotated as 2) vs Absent (annotated as 1)
(d) Y_4: Eye localization (binary mask).
(e) Y_5: Skull segmentation (binary mask).

one comprising low-level features that are shared among all tasks, and the second branches out into task-specific streams. The reason for this choice is that rather than training individual CNNs for each task, we aim to fully exploit the correlation between different tasks, thereby providing more supervision on learning the shared features. Thus, during training, the label prediction loss for all the tasks are minimized simultaneously, which ensures both the invariance of underlying low-level features and the five discriminative classifiers operating on task-specific features.

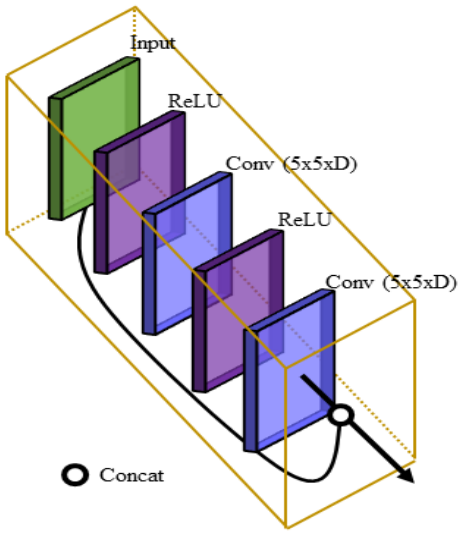


Figure 6.4: Basic building block. A zoomed-in schematic of the basic building block, and its constituent parts. The linear convolutions are followed by ReLU. Concatenation is used for skip layers to fuse multiple-level representations, and to avoid gradient vanishing problems.

In the proposed architecture, all the convolutional layers use kernels of fixed size (3, 5 or 7) with sliding step size $\delta = 1$. Rectified linear unit (ReLU) activations are applied after each convolutional layer. Max-pooling layers with kernel of size 2×2 ($\delta = 2$) are used to avoid too much spatial information loss. Naturally, this leads to relatively deep networks. To compensate for the loss of spatial information after max-pooling, the number of feature maps is gradually increased to and then retained at 256 to keep the number of parameters under control. To avoid overfitting during training, a dropout layer in which 50% of the neurons are randomly set to zero is used in the last shared convolutional layer. This step also encourages the neurons to operate independently, thus avoiding co-adaptation of neurons [30]. In the end, the network branches out from the convolutional layer to produce outputs for each task-specific stream.

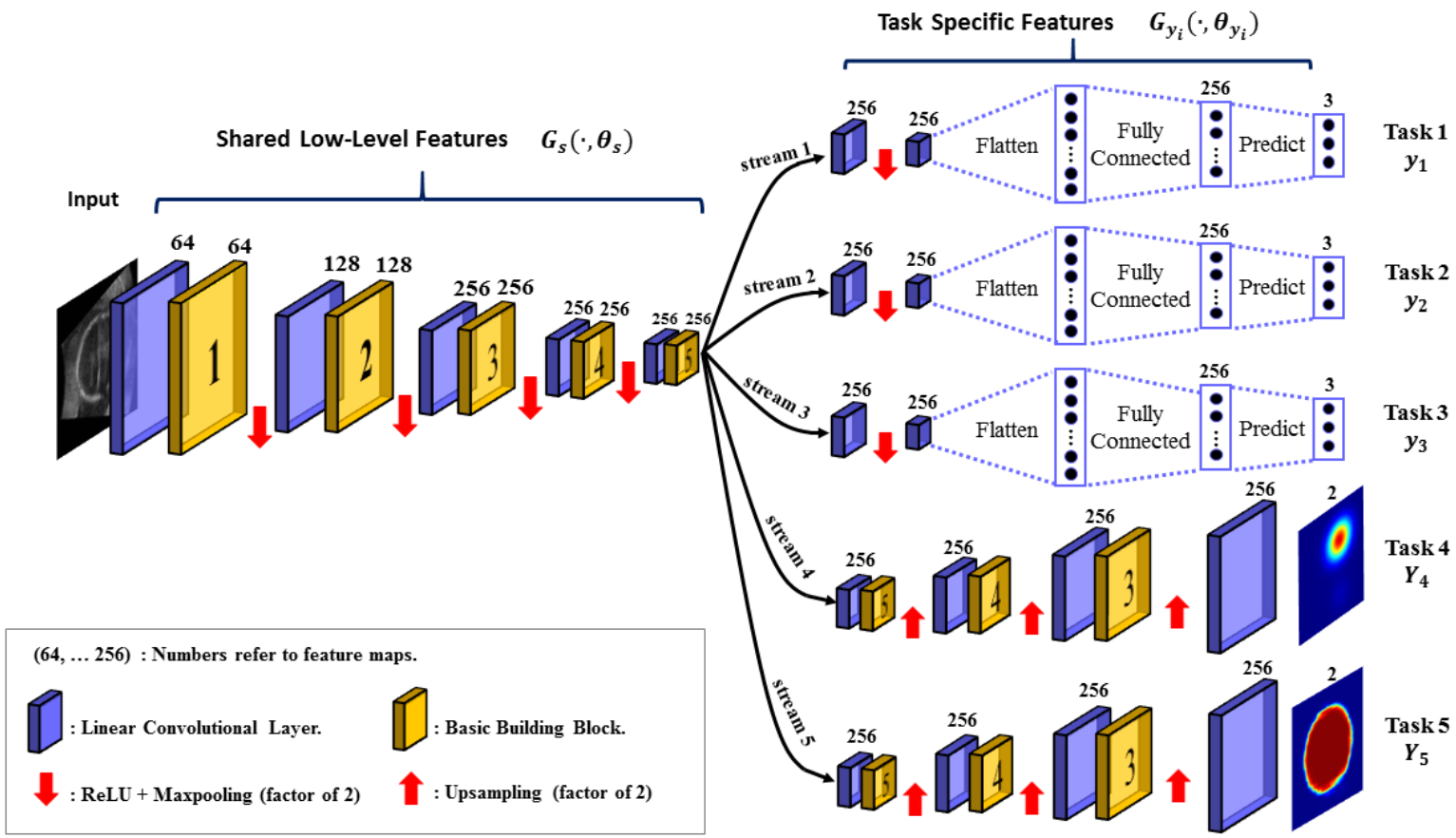


Figure 6.5: Proposed architecture (4 pooling layers) for multi-task learning.
The architecture contains two parts, namely the shared low-level features, and task-specific features. Every convolution is followed by ReLU as non-linearity functions. Convolutional layers are displayed as blue blocks, and red arrows indicate upsampling or downsampling operations, depending on arrow direction.

In the eye localization and brain segmentation tasks, to avoid adding too much complexity, the outputs are only upsampled to produce probability maps of size 64×48 pixels, where each pixel position indicates the likelihood (0.0 to 1.0) of lying within the object of interest. Although this choice results in slightly imprecise eye location after up-sampling to original input image size, evaluation shows this size yields sufficient accuracy for forward tilt correction, and accelerates the inference speed.

In the experiments, we explored different network architectures by varying the kernel size, and network depth to investigate whether the extraction of more complex features improves predictions.

Training Details During training, each input image (384×256 pixels) is rescaled to $\mathbb{X} \in \mathbb{R}^{256\times192\times1}$, and $\mathbb{X}$ is then mapped to feature maps of size $32 \times 24 \times 256$ or $16 \times 12 \times 256$ (depending on network depth) by the low-level feature extractor $G_s(x, \theta_s)$. These feature maps are further passed through several task-specific streams, and convolved by different sets $G_{y_k}(\cdot, \theta_{y_k})$ for prediction of task k. For tasks $k = 4$ (eye localization) and $k = 5$ (brain segmentation), upsampling kernels are trained to recover the spatial resolution of the predicted segmentation map back to 64×48 pixels (Figure 6.5).

More formally, we consider the energy function:

$$\begin{aligned} E(\theta_s, \theta_{y_1}, \theta_{y_2}, \theta_{y_3}, \theta_{y_4}, \theta_{y_5}) = \sum_{i=1}^{N} \Bigg(\sum_{k=1}^{3} \Big(\alpha_k \cdot L_k^i(\theta_s, \theta_{y_k}) \Big) \\ + \sum_{k=4}^{5} \Bigg(\alpha_k \cdot \sum_{mn} L_{k(mn)}^i(\theta_s, \theta_{y_k}) \Bigg) \Bigg) \end{aligned} \tag{6.2}$$

Here, $L_y^i(\cdot, \cdot)$ is the loss for label prediction (multinomial for y_1, y_2, y_3, and binary for y_4 and y_5). In this work, we use the negative log-likelihood as a loss function, and α's refer to the weights for the different tasks ($\alpha_1 = \alpha_2 = \alpha_3 = 3$, $\alpha_4 = \alpha_5 = 1$). Thus, the optimal parameters $(\hat{\theta}_s, \hat{\theta}_{y_1}, \hat{\theta}_{y_2}, \hat{\theta}_{y_3}, \hat{\theta}_{y_4}, \hat{\theta}_{y_5})$ deliver a point, where:

$$\hat{\theta}_s, \hat{\theta}_{y_1}, \hat{\theta}_{y_2}, \hat{\theta}_{y_3}, \hat{\theta}_{y_4}, \hat{\theta}_{y_5} = \underset{\hat{\theta}_s, \hat{\theta}_{y_1}, \hat{\theta}_{y_2}, \hat{\theta}_{y_3}, \hat{\theta}_{y_4}, \hat{\theta}_{y_5}}{\operatorname{argmin}} E(\theta_s, \theta_{y_1}, \theta_{y_2}, \theta_{y_3}, \theta_{y_4}, \theta_{y_4}) \tag{6.3}$$

6.2.2 Estimation of Transformation ($\mathbb{T}$)

The final goal of the pipeline is to estimate a 4×4 geometric transformation matrix $\mathbb{T}$ to align 3D brain images to a common coordinate space (Equation 6.1). $\mathbb{T}$ is estimated by combining predictions from a stack of slices extracted from a given volume. As the object of interest is the fetal brain, only the segmented brain volume is considered for alignment. Thus, the brain extraction mask is first achieved by stacking the skull segmentations ($\mathbb{Y}_5$) in 3D space, and approximating the centre point c and ellispoidal dimensions $[e_x, e_y, e_z]$ of the skull (Figure 6.7b). The remaining matrices are computed as follows:

$\mathbb{M}_s$ is the reflection matrix aligning the head in the inferior-to-superior direction. It is approximated by processing the y_1 labels of the axial slices, in their spatial ordering within the volume (i.e. z-position, Figure 6.6). The ambiguous slices in the volume (i.e. $y_1 = 0$) are first detected, and the modal labels in the slices above and below these z-positions determine whether the volume must be flipped in order to align to an inferior-to-superior configuration. In Figure 6.6, for instance, the ambiguous slice is at $z = 0.71$; below this, the modal label is $y_1 = 1$, and above it is $y_1 = 2$. Thus, the head is in the correct pose and does not require reflection across the xy-plane about the segmented skull centre (i.e. $\mathbb{M}_s$ is the identity matrix).

$\mathbb{M}_a$ is the reflection matrix which aligns the head in an anterior-to-posterior direction, across the yz-plane about the skull centre. It is approximated by ignoring all ambiguous slices and determining the modal label of all slices voting for a specific y_2 direction, namely forward- ($y_2 = 1$) or backward-facing ($y_2 = 2$). In Figure 6.6, the brain is forward-facing.

$\mathbb{M}_o$ defines a rotation about the brain center c correcting for lateral tilt. The brain centre is the centre of mass of the masked brain region. To estimate $\mathbb{M}_o$, the 2D plane $\mathcal{P}_f$ describing the midsagittal plane is recovered by detecting the membrane of the falx traversing the midline of the masked brain in each axial

slice. $\mathcal{P}_f$ is approximated by compiling line candidates recovered from the stack of slices, and plane fitting is achieved using the RANSAC method [102]. The lateral tilt transform $\mathbb{M}_o$ is then derived from the center-point (c) and normal vector ($\vec{n}$) of the midsagittal plane $\mathcal{P}_f$, and its angular deviation (in this case, the dihedral angle) from plane $\mathcal{P}_o$ which is incident to the insonation plane in the sagittal view (s). Ultimately, $\mathbb{M}_o : \mathcal{P}_f \mapsto \mathcal{P}_o^s$.

$\mathbb{M}_e$ If any slices vote for $y_3 = 2$, the eye localization stream is activated (Figure 6.3d), and the stack of $\mathbb{Y}_4$ yields an approximate location of the eye in 3D space (Figure 6.7a). We consider the eye sockets as anchor points to correct for rotation of the brain in the sagittal direction, about the brain centre. The eyes are annotated in each axial slice such that eye segmentation provides a localization of eye center position, and hence $\mathbb{M}_e$. As shown in Figure 6.2b, the rotation angle is determined by the dihedral angle between the plane intersecting the head center and two eye locations ($\mathcal{P}_e$), and the axial plane on the coordinate system ($\mathcal{P}_o^a$).

6.2.3 Training Details

During training, the learning rate is initialized to be $\alpha_{lr} = 10^{-3}$ and divide by 10 every 10 epochs, and the learning momentum is set to $\rho_a = 0.9$. The training took approximately 10 minutes/epoch and 26 minutes/epoch for the 3- and 4-layered networks, respectively. All networks were trained for a total of 36 epochs to ensure convergence. The model was evaluated on an Intel Xeon E5-2630 CPU (2.4 GHz, 16 cores) and a NVIDIA Titan X GPU. Processing takes on average 15 ms per 2D slice (3.4s per volume) to output all the classification labels (y_k) and probability maps ($\mathbb{Y}_k$), and an average of 1s per 3D volume.

6.3 Experiment

6.3.1 Image Data

Data used to validate the proposed framework was obtained from the INTERGROWTH-21st and INTERBIO-21st multi-site databases of fetal US images [1], comprising of healthy and growth-restricted fetuses. Gestational age ranged from 18 to 34 gestational weeks (GW), spanning a period of active brain maturation and hence rapid developmental changes. Thus, structural appearance and composition varied across images. Images were unregistered and not pre-emptively cropped, thus encompassed maternal tissues and other soft tissues observed in a typical obstetric scan. Subject inclusion was determined by visibility of internal brain structures in at least one cerebral hemisphere (accounting for shadowing caused by skull bones). The 3D volumes were acquired using a Philips HD9 curvilinear probe (2.5 MHz wave frequency) by different clinicians, adding variability to probe positioning and therefore image appearance. All images were resampled to an isotropic resolution of $0.6 \times 0.6 \times 0.6$ mm. Training data was comprised of 599 volumes, from which approximately 16 axial slices were extracted from each volume for tuning the multi-task network, yielding a total of 9770 slices for training and validation. The test set was comprised of 140 volumes.

6.3.2 Preprocessing

Two-dimensional axial slices were extracted from each 3D volume and manually annotated with GT labels pertaining to each of the tasks considered during training. For this model, all sampled slices are expected from the cerebrum (i.e. above the cerebellum, Figure 6.3), to include only structural information from the cerebral hemispheres and the eye sockets. Each axial slice is associated with three classification labels and two 2D binary label maps, $\{y_1, y_2, y_3, \mathbb{Y}_4, \mathbb{Y}_5\}$.

Acknowledging the skull's shape as not being strictly ellipsoidal, we generated referential GT segmentations of the skull by deforming ellipsoidal meshes to adhere to the inner cranial contour, as described in [103].

[1] www.intergrowth21.org.uk

6.3.3 Network Architectures

As shown in Table 6.1, we tested six different architectures (A,B,C,D,E,F). By varying the depth of the shared low-level layers (3 or 4 pooling layers), we explored the effect of the size of the receptive field on performance. For instance, it is expected that accurate skull segmentation is achieved by extracting information from a large region, yet the opposite might be true for eye segmentation. By varying the size of the convolution kernel (3×3, 5×5, or 7×7), we aimed to find the balance between the model capacity and data availability. Following the VGGNets [32], the selected kernel size remains unchanged in all convolutional layers.

6.3.4 Evaluation Metrics

Five-fold cross-validation was performed on randomly selected volumes, splitting the data into 80% for training, and 20% for validation. After that, the best-performing model was re-trained using the full dataset and applied to an independent left-out test set of age-matched 3D US data. The proposed pipeline was evaluated for slice-wise classification, brain segmentation, and eye localization accuracy. Slice classification was evaluated by calculating the number of correctly classified slices for tasks y_k ($k = \{1, 2, 3\}$), both from three-way confusion matrices, and two-way confusion matrices where ignore labels ($y_k = 0$, $k = \{1, 2, 3\}$) were not taken into account. Eye localization error was measured as the mean distance between the center of the GT eye annotation and the mean position of the detected eye pixels, in 2D and 3D.

Brain segmentation was evaluated using precision and recall measures, to quantify successful extraction of brain pixels whilst minimizing the inclusion of non-brain tissues. These metrics are independent of the background pixels, which is ideal for the dataset of varying head sizes (foreground) in relation to extracranial tissues (background). Recall is the proportion of correctly segmented brain regions:

$$R(\mathbb{Y}_g, \mathbb{Y}_p) = \frac{\|\mathbb{Y}_g \cap \mathbb{Y}_p\|}{\|\mathbb{Y}_g\|} \tag{6.4}$$

where $\mathbb{Y}_g$ and $\mathbb{Y}_p$ are the GT and predicted image maps, respectively. $\|\cdot\|$ represents the voxel count. Precision is the proportion of correctly excluded background voxels:

$$P(\mathbb{Y}_g, \mathbb{Y}_p) = \frac{\|\mathbb{Y}_g \cap \mathbb{Y}_p\|}{\|\mathbb{Y}_p\|} \tag{6.5}$$

These two measures are further combined to compute the Jaccard index, evaluating overlap between the GT annotation and the predicted segmentation:

$$J(\mathbb{Y}_g, \mathbb{Y}_p) = \frac{\|\mathbb{Y}_g \cap \mathbb{Y}_p\|}{\|\mathbb{Y}_g \cup \mathbb{Y}_p\|} \tag{6.6}$$

where $\|\mathbb{Y}_g \cup \mathbb{Y}_p\| = \|\mathbb{Y}_g\| + \|\mathbb{Y}_p\| - \|\mathbb{Y}_g \cap \mathbb{Y}_p\|$.

6.4 Result

In this section, slice- and volume-wise results are evaluated for each task. Figure 6.6 shows an example obtained by applying the network to a set of slices extracted from a single volume. Figures 6.7a and 6.7b show how slice-wise predictions can be combined to infer 3D eye ($\mathbf{Y}_4^{3D}$) and brain masks ($\mathbf{Y}_5^{3D}$), respectively.

6.4.1 Slice Classification Results

Table 6.1 summarizes the slice prediction results on five-fold validation set, and the comparison of different networks. For all models, an average accuracy of 87.9% or higher was achieved when considering the three-way classification ($y_k \in \{0, 1, 2\}$) on slice identification tasks ($k = \{1, 2, 3\}$). Excluding the ambiguous slices ($y_k = 0$), however, yielded a performance drop for tasks 2 and 3, regardless of network design. This phenomenon can be explained by the severely unbalanced training sets: task 1 consisted of more relevant than ambiguous slices, whereas the opposite was true for tasks 2 and 3.

From networks A, B, to C, kernel size was increased in all convolutional layers. Two-way classification accuracy steadily improved, particularly for task 2 (from 63.8% to 68.6%). These increases in kernel sizes also resulted in models with

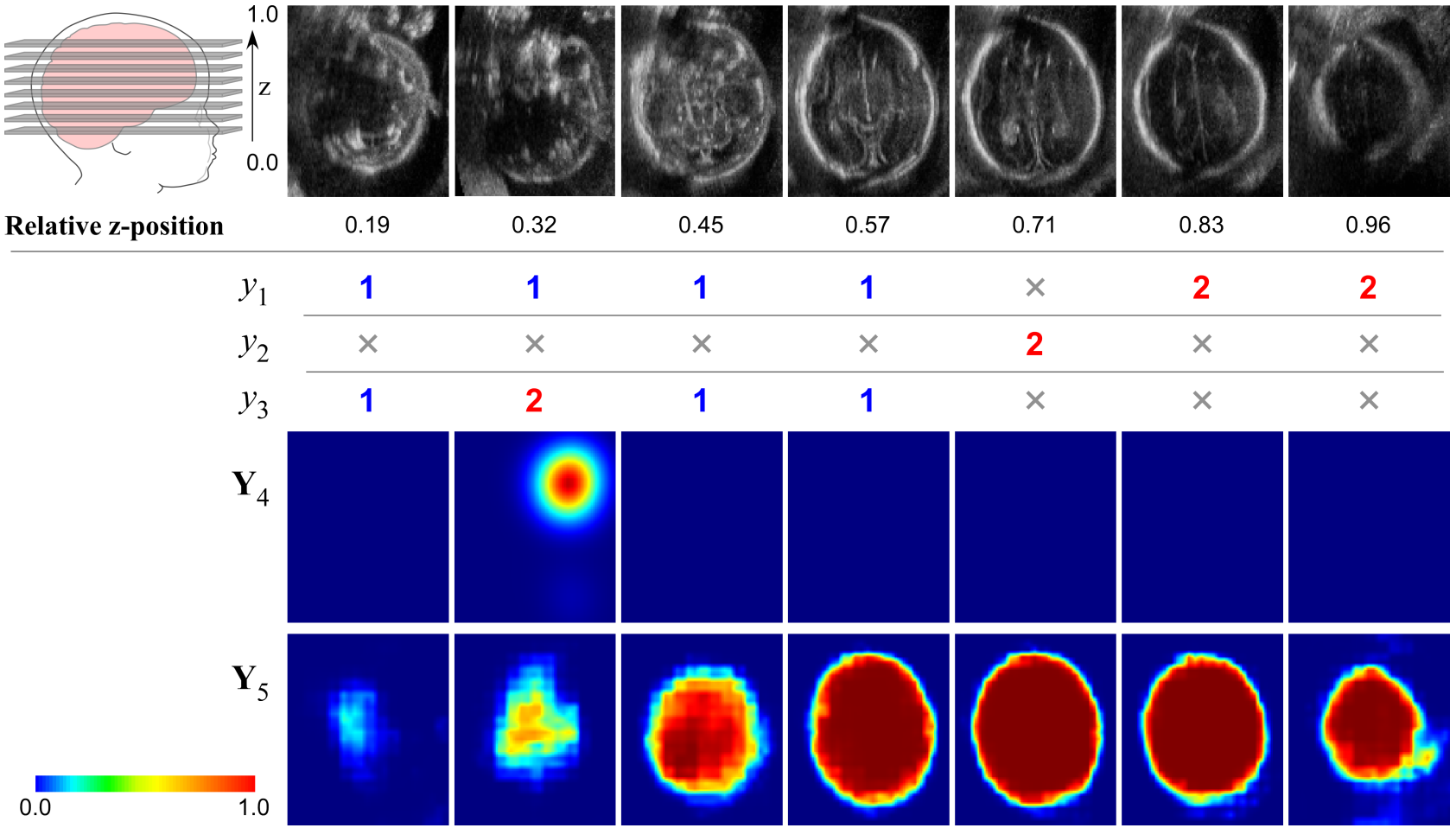


Figure 6.6: Network output example.
Representative example of a typical output from the multi-task network for a single 3D brain volume. Task-specific slice classifications and output segmentations for a stack of axial slices, according to their relative position (z) in the volume. $z = 0.0$: base of the brain, $z = 1.0$: crown.

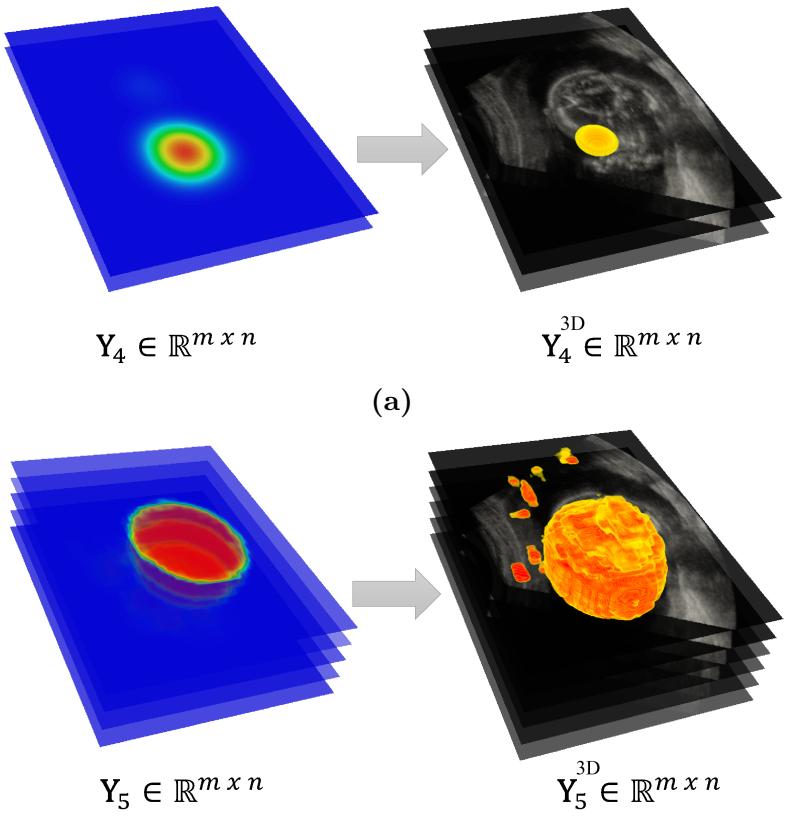


Figure 6.7: Inferring 3D volume results from 2D slice predictions.
Slice-to-volume segmentation of the (a) eye and (b) brain, respectively. Stacking and thresholding of 2D heat maps to obtain a 3D segmentation of the eye ($\mathbf{Y}_4^{3D}$) and brain ($\mathbf{Y}_5^{3D}$), respectively. The smaller outlying brain regions in the 3D brain map are removed by selecting the largest connected region as the final 3D brain mask.

13.7 to 40.0 million parameters. A similar trend was observed from networks D to E with 4 poolings.

When keeping the kernel size fixed and increasing network depth (comparing network B with E), the performance for all tasks was boosted, especially for task 2 (by about 5%). The same behavior was observed when comparing between networks A and D. Thus, network depth helps to deal with the problem of unbalanced training samples. Moreover, network E improves on both segmentation and classification tasks, which may be attributed to the importance of the receptive field in slices where the skull occupies a large region.

Following the trend, the kernel size was further increased in network F (55.4 million parameters). Interestingly, the performance dropped for all tasks when

Network		Kernel Size	Classification (3-way)			Classification (2-way)		
			y_1	y_2	y_3	y_1	y_2	y_3
3-pool	A	3×3	90.4 ± 1.0	90.2 ± 1.0	87.9 ± 0.6	94.7 ± 0.5	63.8 ± 5.3	83.2 ± 0.4
	B	5×5	90.6 ± 1.0	90.7 ± 0.9	89.5 ± 0.8	94.9 ± 0.7	66.7 ± 3.3	85.4 ± 1.3
	C	7×7	91.4 ± 1.4	91.3 ± 1.3	89.1 ± 1.1	95.0 ± 1.0	68.6 ± 5.3	84.6 ± 2.1
4-pool	D	3×3	91.0 ± 0.5	90.8 ± 0.9	88.5 ± 1.0	94.4 ± 0.5	68.9 ± 2.2	83.5 ± 1.6
	E	$\mathbf{5 \times 5}$	$\mathbf{92.1 \pm 0.7}$	$\mathbf{91.9 \pm 0.7}$	$\mathbf{90.3 \pm 0.6}$	$\mathbf{95.8 \pm 0.4}$	$\mathbf{70.8 \pm 2.0}$	$\mathbf{86.4 \pm 1.5}$
	F	7×7	91.3 ± 1.1	91.3 ± 0.7	89.7 ± 1.2	95.5 ± 1.1	66.3 ± 2.2	85.7 ± 2.2

Table 6.1: Slice-wise classification accuracy.
Mean accuracy (± standard deviation) of slice-wise classification computed over the five-fold cross-validation sets. Axial slice label prediction accuracy on 2D images for tasks y_1, y_2, y_3. The accuracy for 3-way (where $y_k \in \{0, 1, 2\}$) and 2-way (excluding $y_k = 0$) classification is reported for all six network architectures. Network E (in bold) outperformed the others in all classification tasks.

Network		Kernel Size	Segmentation		Localization (mm)		No. Params
			$\mathbf{Y}_4$ (eye)	$\mathbf{Y}_5$ (skull)	d_4 (eye)	d_5 (skull)	
3-pool	A	3×3	0.47 ± 0.13	0.71 ± 0.22	2.50 ± 2.96	1.82 ± 1.71	13.7 M
	B	5×5	0.53 ± 0.13	0.79 ± 0.21	2.11 ± 2.01	1.50 ± 1.68	25.1 M
	C	7×7	0.52 ± 0.14	0.80 ± 0.20	2.49 ± 3.02	1.49 ± 1.64	40.0 M
4-pool	D	3×3	0.52 ± 0.15	0.79 ± 0.20	2.49 ± 2.95	1.50 ± 1.68	11.9 M
	E	$\mathbf{5 \times 5}$	$\mathbf{0.55 \pm 0.12}$	$\mathbf{0.83 \pm 0.18}$	$\mathbf{1.92 \pm 1.36}$	$\mathbf{1.19 \pm 1.26}$	**29.6 M**
	F	7×7	0.55 ± 0.13	0.82 ± 0.19	2.14 ± 2.17	1.24 ± 1.25	55.4 M

Table 6.2: Slice-wise segmentation and object localization accuracy.
Mean accuracy (± standard deviation) of slice-wise segmentation and object localization computed over the five-fold cross-validation sets. Jaccard index ($J_k = 0.0$: no overlap; $J_k = 1.0$: perfect overlap) and center-point distance (d_k in mm, between GT and predicted segmentations) shown for the eye and skull segmentation tasks, $\mathbf{Y}_4$ and $\mathbf{Y}_5$, respectively.

compared with network E. One explanation would be that the model started to overfit given the amount of training data in each fold (about 7000 slices).

Ultimately, architecture E is selected as the best performing model in terms of accuracy. We calculated the receiver operator characteristics (ROC) curves (Figure 6.8) for the three classification tasks. The AUC (Areas Under Curve) was 0.84 ± 0.02 when distinguishing between superior and inferior brain slices (task 1), 0.84 ± 0.02 for determining the direction in which the brain faces (task 2), and 0.82 ± 0.01 for identifying eye-containing slices (task 3). The small standard deviations show that the model is stable, as designed, for performing the tasks. Network E is therefore re-trained using all the training data (about 10,000 slices), and report results of applying it to the independent test set in the subsequent sections.

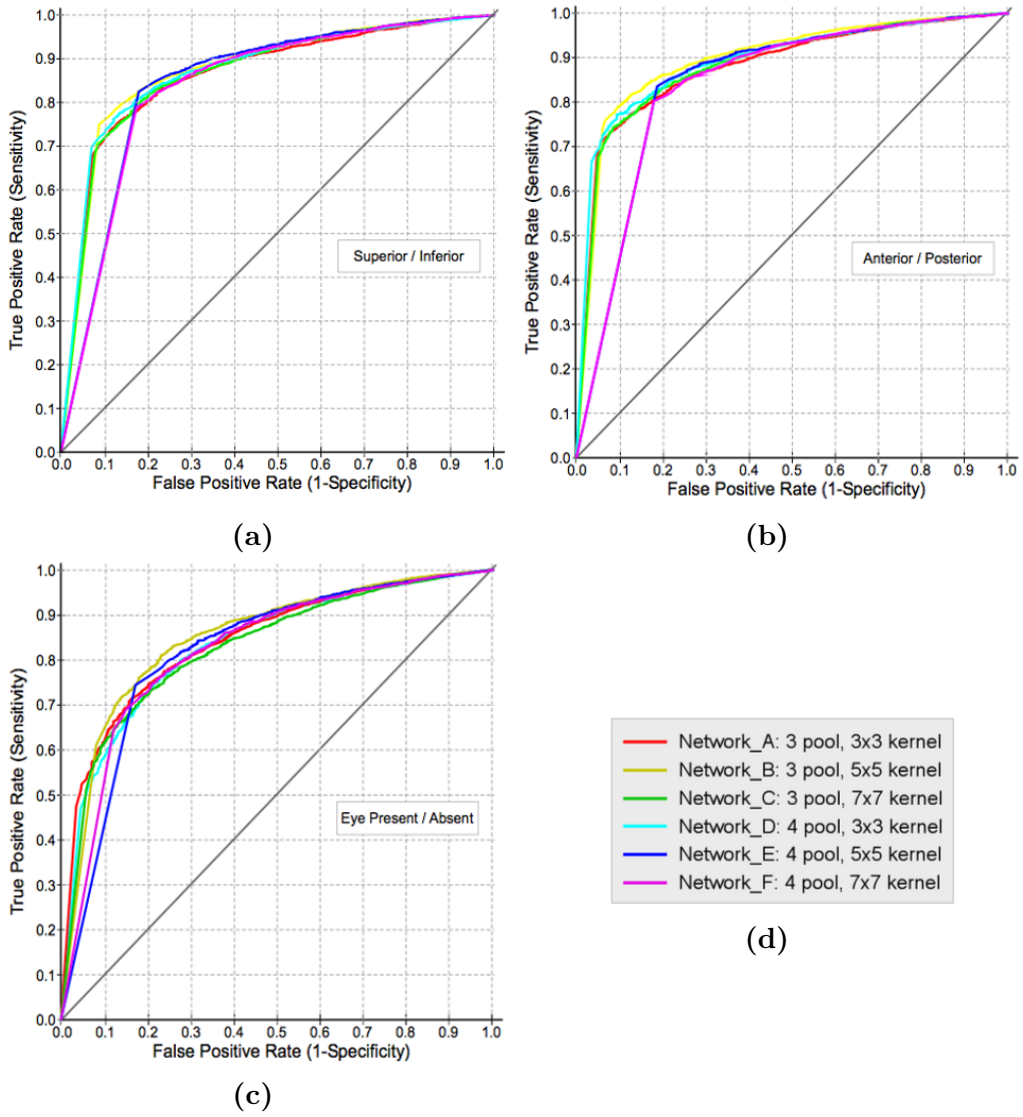


Figure 6.8: Receiver Operator Characteristics Curve.

6.4.2 Volume Classification Results

An approximation of overall brain orientation can be inferred by combining the predictions from a stack of slices extracted from a given volume. For instance, the brain's superior-to-inferior configuration is revealed by the order in which the axial slices are labelled relative to slice position. Figure 6.6 shows a brain volume with 'inferior' labels ($y_1 = 1$) in the lower half of the volume, and 'superior' labels ($y_1 = 2$) closer to the top, consistent with the brain being in an inferior-to-superior configuration ($y_1^{3D} = 1$). Table 6.3 presents the 3D volume-based prediction, showing that prediction of superior-to-inferior brain orientation was 99.8% accurate for the

cross-validation set. The y_1^{3D} task failed on only one of the 140 test volumes, where the brain did not occupy at least 30% of the image space.

The proposed model was capable of detecting an eye socket in 96.4% (135/140) of the test volumes. As with slice-wise classification, the most challenging prediction proved to be the anterior-to-posterior orientation (y_2^{3D}), where average accuracy was 88.6% (124/140), also due to a small amount of y_2 voting slices in the stack ($y_2 = 1$ or $y_2 = 2$), relative to the number of ambiguous slices.

Dataset	Samples	Age range (weeks)	Classification			Localization (mm)	Segmentation	Alignment
			y_1^{3D}	y_2^{3D}	y_3^{3D}	$\mathbf{Y}_4^{3D}$	$\mathbf{Y}_5^{3D}$	error (mm)
Val set	111 ± 5	26.6 ± 4.3	99.8 ± 0.5	96.1 ± 1.7	93.5 ± 2.5	5.0 ± 5.3	0.82 ± 0.08	9.3 ± 4.1
Test set	140	26.0 ± 4.4	99.3	88.6	96.4	6.9 ± 6.9	0.82 ± 0.07	9.3 ± 4.4

Table 6.3: Volume classification performance.
Application of network E with 4 pooling layers and kernel of size 5×5 on an independent test set. Mean accuracy ($\pm$ standard deviation) of volume-wise classification computed for tasks y_k^{3D}. Classification scores indicate the accuracy of combining slice-based classification labels to generate a prediction for overall brain orientation.
Eye localization performance is quantified by Euclidean distance, and Jaccard index scores express brain segmentation performance. Target alignment error is quantified as the average of Hausdorff distances between pairs of skull surface landmarks (manual vs predicted alignment). A total of 6144 skull landmarks were used per volume.

6.4.3 Brain Segmentation Results

Segmentation of the brain is achieved by classifying the pixels within the skull boundaries on a slice-by-slice basis. In Table 6.2, we compare the performance of the networks in terms of slice-wise segmentation. Network E yielded the best performance, with a Jaccard score of 0.83 ± 0.18. Figure 6.9a demonstrates that the probabilities of voxels near the brain center are typically above 90%. This value, if taken as a threshold, could inform a brain localization task, despite the presence of a few outlying regions. Indeed, we found that by thresholding the probability map at 0.5, the mean error distance between the centroids of the GT and the detected 2D brain regions was 1.19 ± 1.26 mm, only slightly increasing with gestational age, accounting for brain growth ($r = 0.22,\ p < 0.002$).

It is also evident from Figure 6.10a that segmentation performance is best in the axial slices extracted near the brain centre (Jaccard > 0.80), whereas segmentation

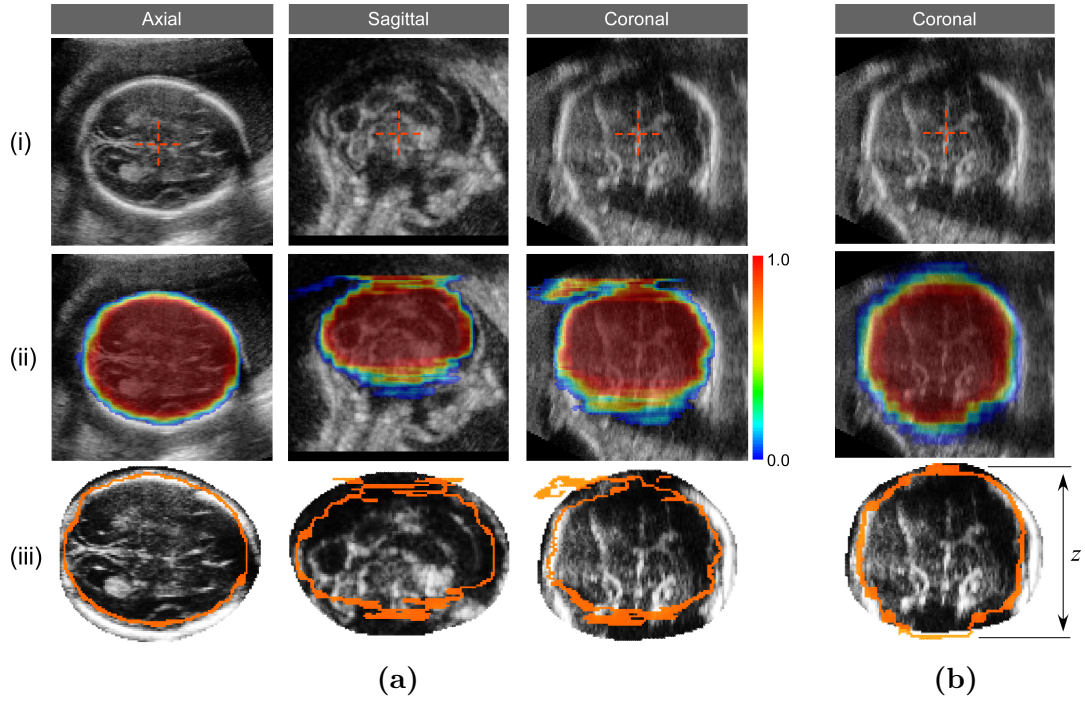


Figure 6.9: Brain segmentation performance.
(a) Sagittal, coronal, and axial views of an example US image (top row), with the output slice-wise probability maps superimposed (middle row). A 3D brain mask is obtained by preserving the largest connected component of the probability map (bottom row, in orange), which is then compared to the GT cranial segmentation (used here for brain masking).
(b) Segmentation result of passing a coronal slice as input to the network. Skull boundaries are clearly detected, and the range of axial slices (z) can be determined.

accuracy lowers in the extremal brain slices. These slices either contain the cerebral fontanelles or the infratentorial region, where skull tissues are not fused and so image boundaries are incomplete. This behaviour was observed in all test cases.

The slice-wise predictions can be further combined to obtain a 3D brain mask ($\mathbf{Y}_5^{3D}$. Figure 6.9a shows a typical segmentation result superimposed on the 3D volume from which the axial slices were extracted. However, as shown from the sagittal view, simply stacking the axial slices to reconstruct 3D volumes may introduce small prediction inaccuracies due to incomplete skull boundaries. Therefore, we run the same model on the coronal slices. Note that, although the networks have only been trained with axial slices, it still shows good generalization for skull segmentation of the coronal slices, and there is only a narrow margin of low probability pixels (meaning uncertainty), near the skull edges (Figure 6.9b).

Conveniently, by obtaining a segmentation map of a coronal slice (which has clear skull edges), it is possible to exclude the extremal slices by inferring the top and bottom bounds (z) of the axial view. As shown in Figure 6.9b, this enables the restriction of axial slices to test so as not to extend beyond the skull bounds.

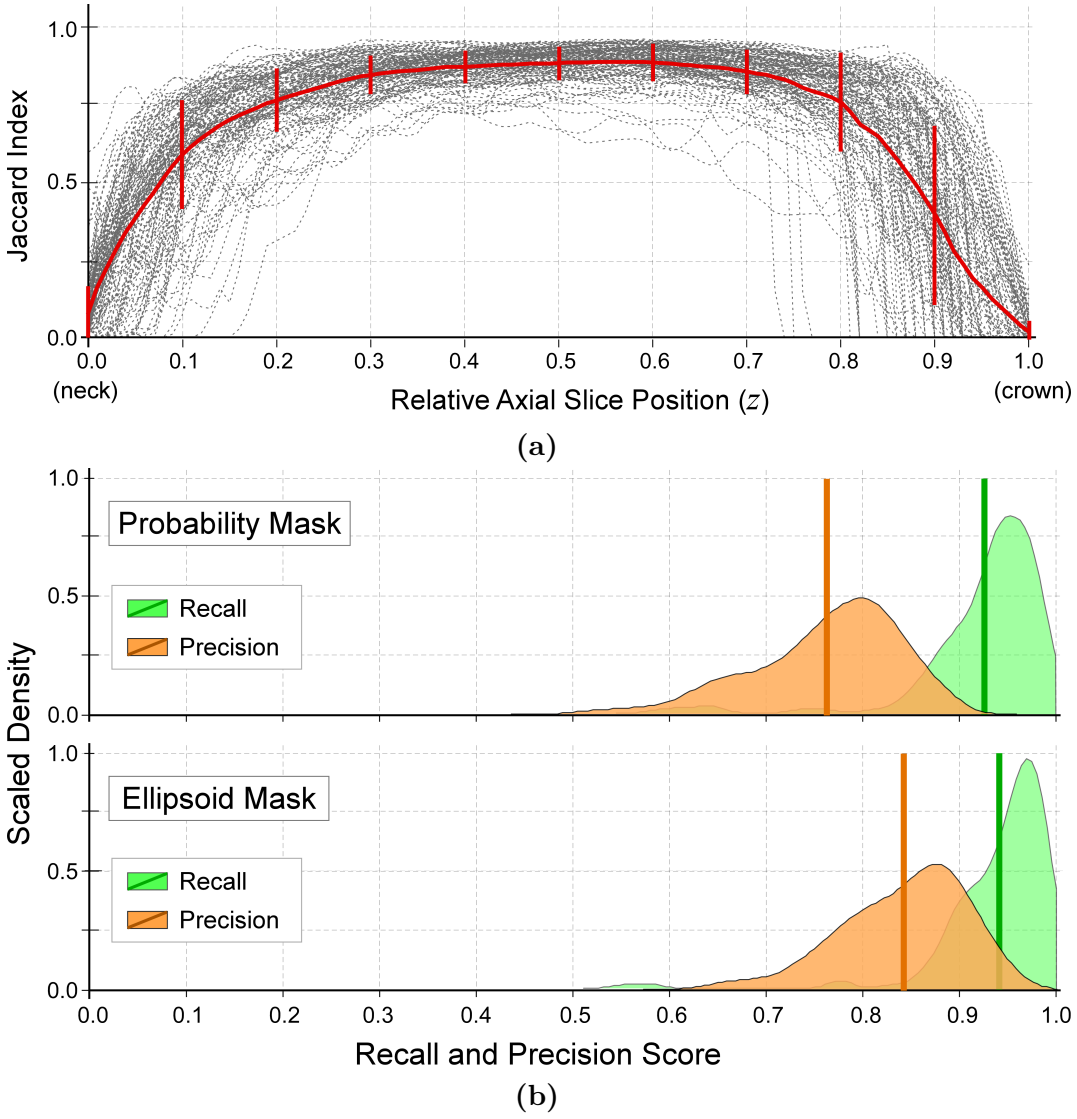


Figure 6.10: Brain segmentation performance.
(a) Slice-wise brain segmentation overlap with respect to axial position. Each dotted line represents a single volume. Overlap is highest in the middle slices of the brain, and deteriorates in the extremal slices, near the fetal neck ($z = 0$) or the crown ($z = 1$). (b) Histogram plots showing the precision and recall 3D brain segmentation. When using the probability mask (top), the recall and precision of brain voxels is lower than if an ellipsoid is approximated from the probability mask. The latter tends to exclude incorrectly classified voxels in the extremal slices.

To recover the 3D brain region, outlying regions were removed from the

thresholded probability map by selecting the largest 3D connected component. The histogram plots in Figure 6.10b summarize the distributions of precision and recall of segmenting the brain region. The probability mask yielded high recall (0.93) and a slightly lower precision (0.76), when compared to the GT mask. Generally, the probability mask confidently includes tissues inside the fetal skull (high recall), but over-segments by incorrectly including a few non-brain voxels in the extremal axial slices (near the top and bottom of the brain, Figure 6.10a).

An ellipsoidal mask was generated by fitting an ellipsoid to the probability mask. Such a mask generally excludes non-brain tissues in the extremal slices, and fully covers all intra-cranial structures. As illustrated in Figure 6.7b, this mask yields high average recall (0.94) and good precision (0.84) on account of full brain coverage. However, it fails to adhere to the brain's non-ellipsoidal shape, thereby always overestimating by including some background voxels.

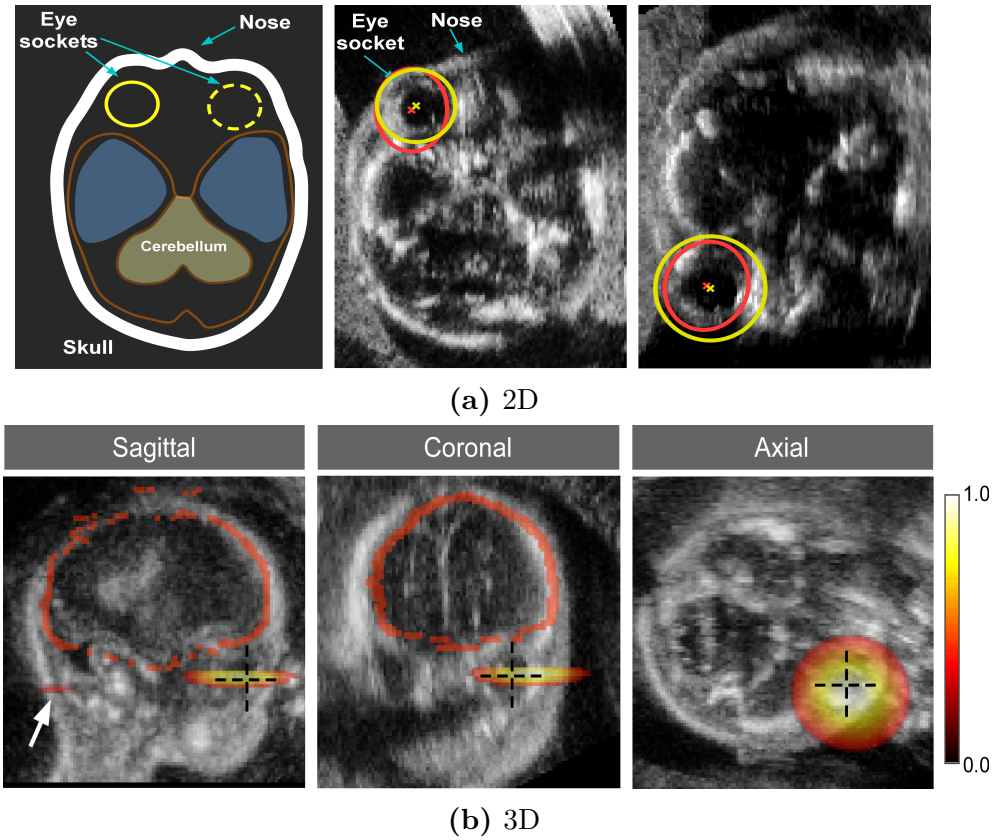
6.4.4 Eye Localization Results

The model's eye detection stream outputs a 2D probability map $\mathbf{Y}_4 \in \mathbf{R}^{m \times n}$, indicating possible eye locations (Figure 6.11). The detections were able to incorporate most foreground eye pixels, which yielded a recall of 0.744. The mean eye localization error on 2D slices was 1.92±1.36 mm (Table 6.2), and gestational age was found to be of no statistical significance to this behaviour ($r = -0.07,\ p = 0.45$).

Each axial slice typically votes for one eye position, and by averaging the stack of eye locations for a given volume, the eye location can be approximated in 3D space. However, there are instances in which the probability map yields two locations. In such cases, both locations detected on each axial slice are kept and a Ball Tree algorithm is used to retain only the position within a 95% confidence interval of the eye locations recovered from adjacent slices. When the slice-based eye detections were averaged to estimate 3D eye localization for each head volume, the mean error was 6.94 ± 6.87 mm (Table 6.3).

We consider an eye detection successful if it lies within 50% of the age-appropriate biocular distance (or eye-to-eye diameter). By this definition, eye detection was

successful in 95.5% (127/133) of cases. Of the remaining six cases, eye detection failed when the fetal head did not fill at least 50% of the image (4/6), or in images with incomplete eye socket boundaries caused by fetal motion (1/6) or partial occlusion (3/6). Nevertheless, the model detected structures with similar intensity and boundary characteristics as the eye orbits, but of a larger scale or containing stronger edges. Ultimately, failure to detect the eye sockets can be attributed to improper image acquisition.



(a) 2D

(b) 3D

Figure 6.11: Representative examples of eye localization.
(a) Axial slices collected at the level of the eye sockets, displaying GT eye annotations (yellow), and predicted segmentations (orange). Corresponding eye centers are annotated with crosses. Localisation error (d_c) is the distance between these GT and predicted eye centers. (b) Sagittal, coronal and axial views of eye annotation in 3D space. White arrow indicates second eye hot-spot location detected by the network but with lower probability and in fewer slices.

6.4.5 Brain Standardized Re-orientation Estimation

The quality of brain standardized re-orientation was scored on the basis of how well skull geometry was approximated by the automatically-recovered transformation $\mathbf{T}$ (Equation 6.1). Due to low definition of skull boundaries and longitudinal variations in structural shape and intensity, a geometric comparison between a manually-aligned ellipsoidal surface and a surface aligned by $\mathbf{T}$ is performed (Figure 6.12a). Surface agreement was measured as the Euclidean distance between the manually- and automatically-aligned surface points, whilst preserving the topology of the surface points on alignment (Figure 6.12a). The Hausdorff distances between the manual and automated surfaces are shown in Figure 6.12b. The mean distance error was 9.3±4.7 mm for volumes in which the model correctly predict all tasks (124/140). Among the 16 volumes whose orientations (y_k^{3D}) were misclassified, only one had an incorrect superior-inferior prediction (y_1^{3D}). The remaining 15 subjects had misclassified anterior-to-posterior directions. Their surface distance error increased with gestational age, correlating with occipitofrontal diameter ($r = 0.981$): the longest diameter extending from the front to the back of the brain.

From the ellipsoidal surface, a binary mask that encompasses the brain can be extracted, and thus approximates intra-cranial volume. Figure 6.12c shows the good correlation between true and predicted cranial volumes ($r = 0.969$). This suggests that the proposed model is also capable of approximating intra-cranial volume, which has been proposed as a proxy for brain growth [104, 105]. This is true even when the prediction of brain orientation fails.

The quality of brain standardized re-orientation is further illustrated in Figure 6.13, which shows the average intensity and voxel variance of all correctly classified brains, using the estimated transformations. Intensity variance is a measure of the difference between the intensities of a set of images [106], defined as follows:

$$\mathbf{X}^{var} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\mathbf{X}_i \left(\mathbf{T}_i \right) - \overline{\mathbf{X}} \right) \tag{6.7}$$

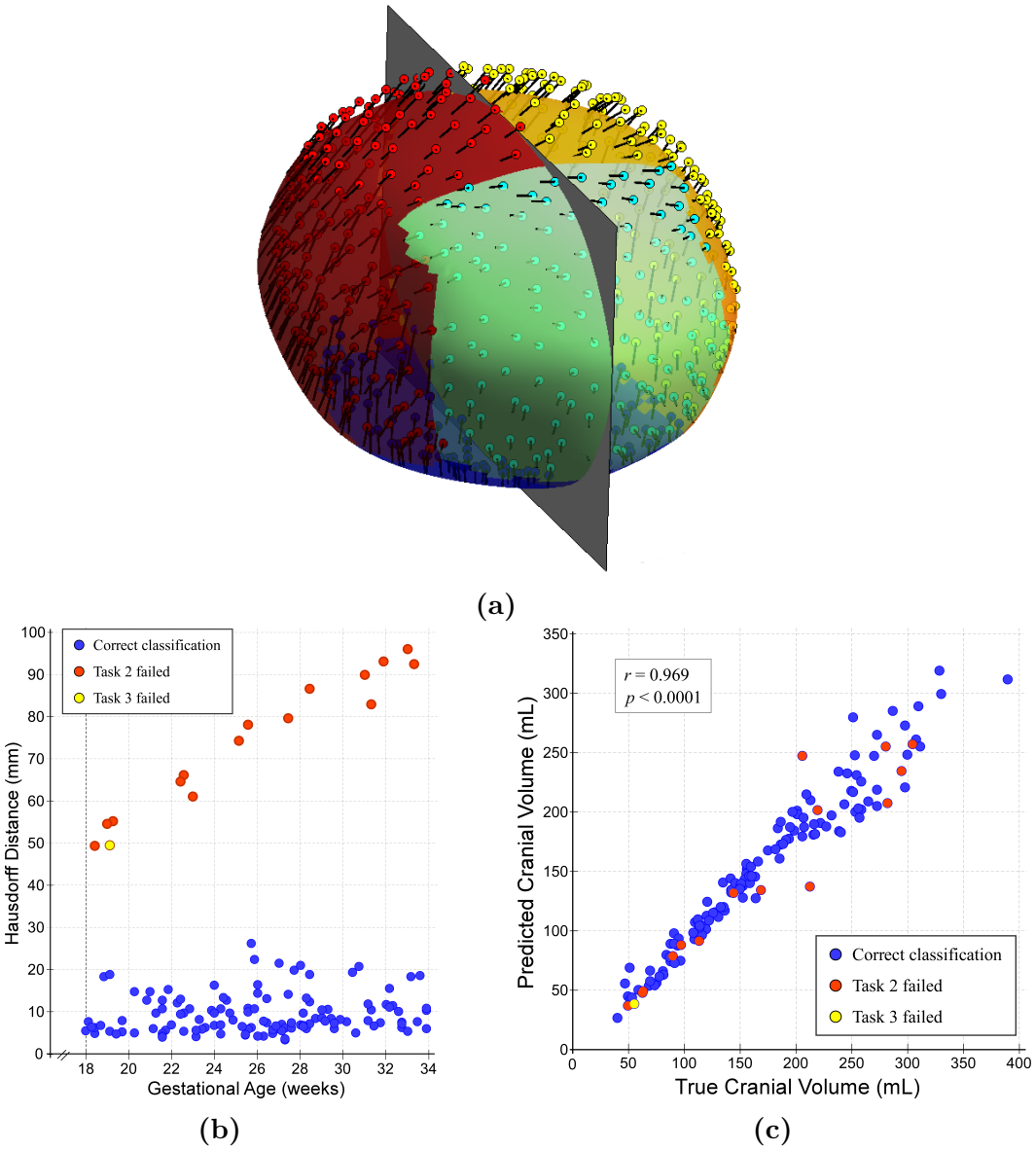


Figure 6.12: Brain re-orientation results.
(a) Illustration of the distance between the manually defined cranial surface and the points on the automatically-recovered surface (shown as coloured circles).
(b) Hausdorff distance (in mm) plotted against gestational age. Each data point represents a single 3D volume. Data points with correctly classified brain orientation (blue) had a lower distance error than those with incorrect classification of superior-inferior (yellow) or anterior-posterior (orange) orientation.
(c) High agreement between model-estimated and true cranial volume.

and the intensity average is

$$\overline{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}_i \left(\mathbf{T}_i \right) \tag{6.8}$$

where $\mathbf{X}_i \left(\mathbf{T}_i \right)$ is the i^{th} 3D image transformed by matrix $\mathbf{T}_i$, and N is the number

of images in the set. Strong agreement is characterized by the minimum pixel intensity variance among the registered images. It is evident from Figure 6.13 that before re-orientation there is high variance, mostly corresponding to variations in image acquisition. The structures are also indiscernible in the average image. After standardized re-orientation using the proposed method, variance values were lower and high correspondence was observed in several anatomical regions, such as the Sylvian fissure, thalami, corpus callosum, and choroid plexus. Variance will, in part, reflect structural differences between fetuses at different stages of development, but also accounting for the fact that our approach estimates non-deformable transformations. Strikingly, both the average and variance maps captured hemispheric symmetry, despite the fact that each image contributed structural information from only one hemisphere. This demonstrates that the proposed method successfully re-oriented the 3D US scans of the fetal brain.

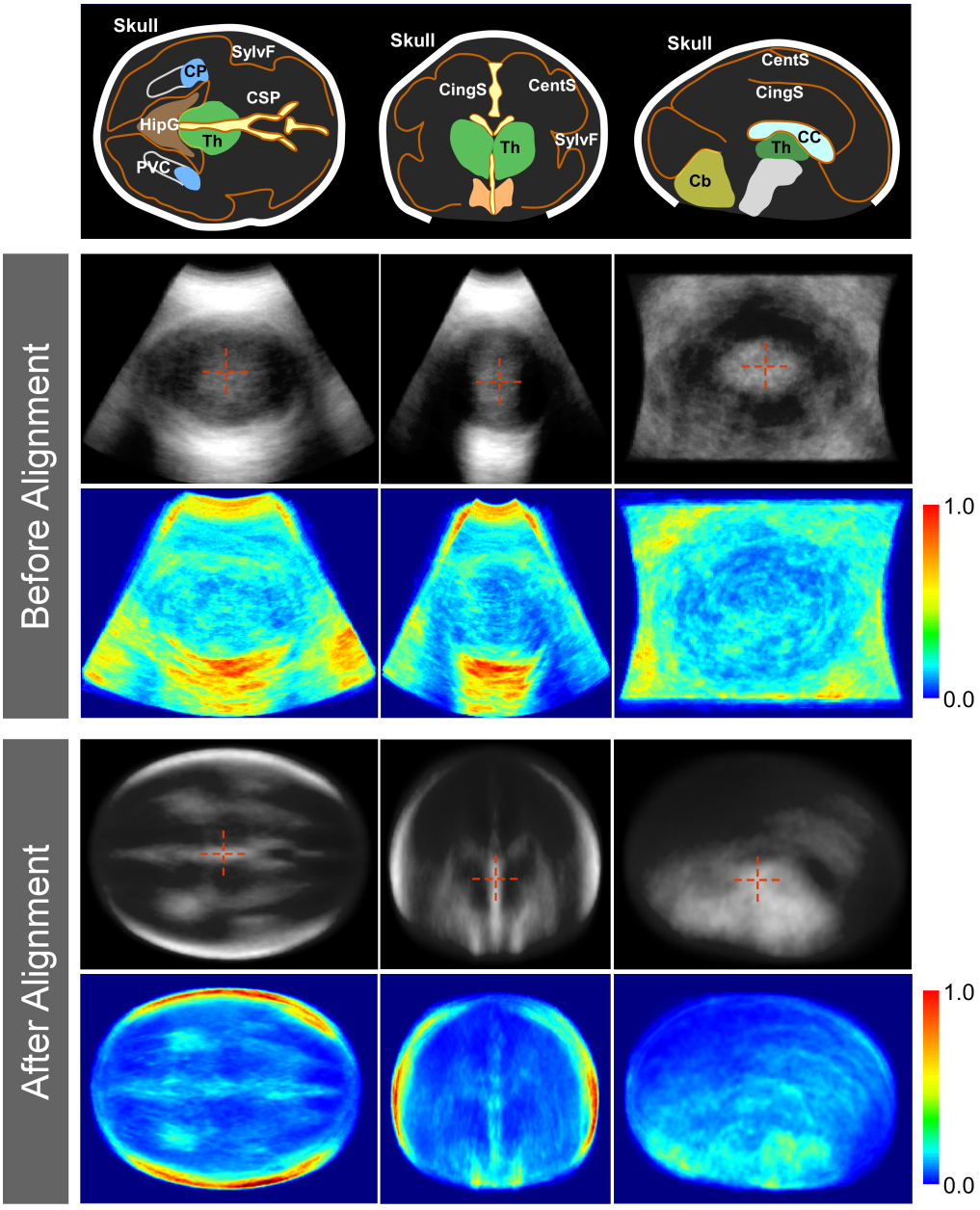


Figure 6.13: Mean and vairiance brain maps..
Axial, coronal, and sagittal views of US images and the intensity average (grayscale) and variance (colour) maps and of all 124 successfully classified volumes, before and after alignment. Mean brain maps constructed by aligning the images and averaging their pixel intensities (red: strong structural correspondence; blue: low correspondence). Prior to alignment, brain structures are unintelligible.
SylvF: Sylvian fissure; CC: Corpus callosum; CSP: Cavum septum pellucidum; Th: Thalami; HipG: Hippocampal gyrus; CP: Choroid plexus; CingS: Cingulate sulcus; CentS: Central sulcus; PVC: Posterior ventricle cavity; Cb: Cerebellum.

6.5 Discussion

The methodology presented in this work was developed to address the challenge of aligning 3D US images of the fetal brain, to further advance automated analysis of brain maturation (e.g. [103]). One of the factors limiting the use of 3D neurosonography for such studies is the partial appearance of brain structures, caused by fetal position and interactions between the skull bones and the US signal. Nonetheless, certain structures are consistently observed over the course of pregnancy, regardless of gestational age and fetal pose. In this chapter, we present a single multi-task fully convolutional neural network which automatically locates and segments the fetal brain and eye sockets in 2D and 3D images. The proposed model is capable of incorporating this information to estimate 3D brain orientation, providing a tool for fast, age-independent, and consistent co-alignment of 3D fetal neurosonographic images.

The model is developed from 599 volumes of data ranging from 18 to 34 gestational weeks, and evaluated on a large clinical dataset consisting of 140 healthy and growth-restricted fetuses, acquired from several ethnic and geographical groups. This age range is characteristically marked by dramatic neurodevelopmental changes and increasing cranial ossification, which lead to variations in observable structures. Despite these longitudinal variations, the brain standardized re-orientation results are consistent and generalizable. The model does not require any age input, and is applicable to a wide distribution of gestational ages (18-34 weeks). It rather addresses the problem of intensity-based image alignment by segmenting and localizing stable structures observed across different developmental stages. The average of the co-aligned images depicted high correspondence in fetal brain anatomies, revealing the potential of the proposed framework in aligning large datasets for inter-subject comparisons, or longitudinal studies for tracking brain maturation.

We have shown that this fully-automated method successfully aligns 3D fetal neurosonographic data onto a pre-defined coordinate system, which could enable visual interpretation of brain anatomies. In a clinical setting, this may assist in the navigation through the brain volume and establishing coordinate locations of

key cerebral landmarks [107], and allow neurosonographers to localize the standard biometry planes from which to extract 2D measurements for fetal growth monitoring, and to extract oblique planes for neuropathological assessment [108].

As the brain develops during the fetal period, some structural changes are independent of head size and their assessment would warrant the need for affine alignment. The proposed framework can readily be used to estimate an affine transformation matrix that excludes brain size from the standardized re-orientation process. Thus, a size-independent coordinate system can be defined over the brain, which would facilitate feature extraction for further analysis of brain maturation. This, for instance, has clear applications in fully-automating the gestational age prediction framework presented in the previous work [103].

In general, it was observed that, some errors in predicting the overall head orientation can be attributed to the fetal head failing to occupy at least 50% of the image space, as the protocol recommendations [109]. Such images would classify as failed acquisitions, and this result indicates the model's sensitivity to image resolution and scaling, and highlights a current requirement for any input data.

In summary, this chapter has presented a deep learning-based approach for predicting the affine transformation which maps a 3D brain image onto a common coordinate space. To obtain the affine transformation, the whole task was decomposed into several simply, easily-solvable tasks. This fully-automated pipeline for segmentation and standardized re-orientation of 3D fetal neurosonography provides a solution towards anatomical assessment early in pregnancy.

7

Detection, Re-orientation & Segmentation in MR Cardiac Videos

7.1 Introduction

In Magnetic Resonance (MR) image analysis, pixelwise segmentation of the left ventricular (LV) myocardium and the four cardiac chambers in 2-D steady state free precession (SSFP) cine sequences is an essential preprocessing step for volume estimation (e.g. ejection fraction, stroke volume, and cardiac output); morphological characterization (e.g., myocardial mass, regional wall thickness and thickening, and eccentricity); and strain analysis [110]. However, automatic MR cardiac segmentation remains a notoriously difficult problem, given:

- Biological variability in heart size, orientation in the thorax, and morphology (both in healthy subjects and in the context of disease).
- Variability in contrast and image appearance with different scanners, protocols, and clinical planes.
- Poorly defined borders between the ventricles and the atria, as well as between the chambers and the vasculature.

Three broad approaches have been employed to address this complexity. First, the scope of the problem can be restricted, e.g. to segmentation of the LV myocardium and bloodpool in the SA view only. Second, user interaction can be used to provide a sensible initialization, supply anatomical landmarks, or correct errors. Third, prior knowledge of cardiac anatomy may be incorporated into model-based approaches. Clearly, none of these approaches are ideal: the first limiting the information which can be gleaned from the algorithm; the second being labor-intensive for the clinician; and the third requiring careful construction of algorithmic constraints. Recently, Convolutional Neural Networks have been applied to short axis MR segmentation of LV blood-pool [111–113], the RV blood-pool [114], and both simultaneously [115, 116]. In these methods, either detection and segmentation was performed separately [111–114], or detection was assumed as a component of image preprocessing [115, 116]. Although biomedical image segmentation is more efficiently accomplished if structures of interest have first been detected and transformed into a canonical orientation, neither end-to-end detection and segmentation nor transformation into a canonical orientation prior to segmentation has been described in the literature.

In this chapter, we propose the Ω-Net (Omega-Net) in § 7.2, a novel CNN architecture trained end-to-end to tackle three important tasks: detection, transformation into a canonical orientation, and segmentation. For simplicity, we only use the U-Net (in VGGNets style) as the fundamental component of the coarse- and fine-grained segmentation modules [44], though more advanced networks such as ResNets [35] could be substituted as well. Inspired by the spatial transformer network [40], the fully differentiable module is further simplified and trained to achieve detection and transformation into a canonical orientation simultaneously. The transformed image is then fed into a fine-grained segmentation module that resembles the stacked hourglass architecture [117]. This is motivated by the fact that CNNs in essence is known to be not rotation equivariance or invariance, therefore, adding the pre-alignment into the end-to-end trainable pipeline can effectively decrease the complexity of segmentation for later stage of the model.

Through experiments (§ 7.3), we demonstrate that the Ω-Net is capable of the fully automatic segmentation of five foreground classes (LV myocardium, the left and right atria, and the left and right ventricles) in three orthogonal clinical planes (short axis, SA; four-chamber, 4C; and two-chamber, 2C), with simultaneous rigid transformation of the input into a canonical orientation based on detection (defined separately for each view, Figure 7.1). Moreover, the network is trained on a multi-center population of patients with hypertrophic cardiomyopathy (HCM), which increases the complexity of the problem due to the highly variable appearance of the LV in these patients. As a result, network performance as measured by weighted foreground intersection-over-union (IoU) was substantially improved in the best-performing Ω-Net compared with the baseline U-Net segmentation without detection and orientation alignment (0.858 **vs** 0.834). Meanwhile, the intermediate detection and re-orientation procedures make subsequent processing more effective.

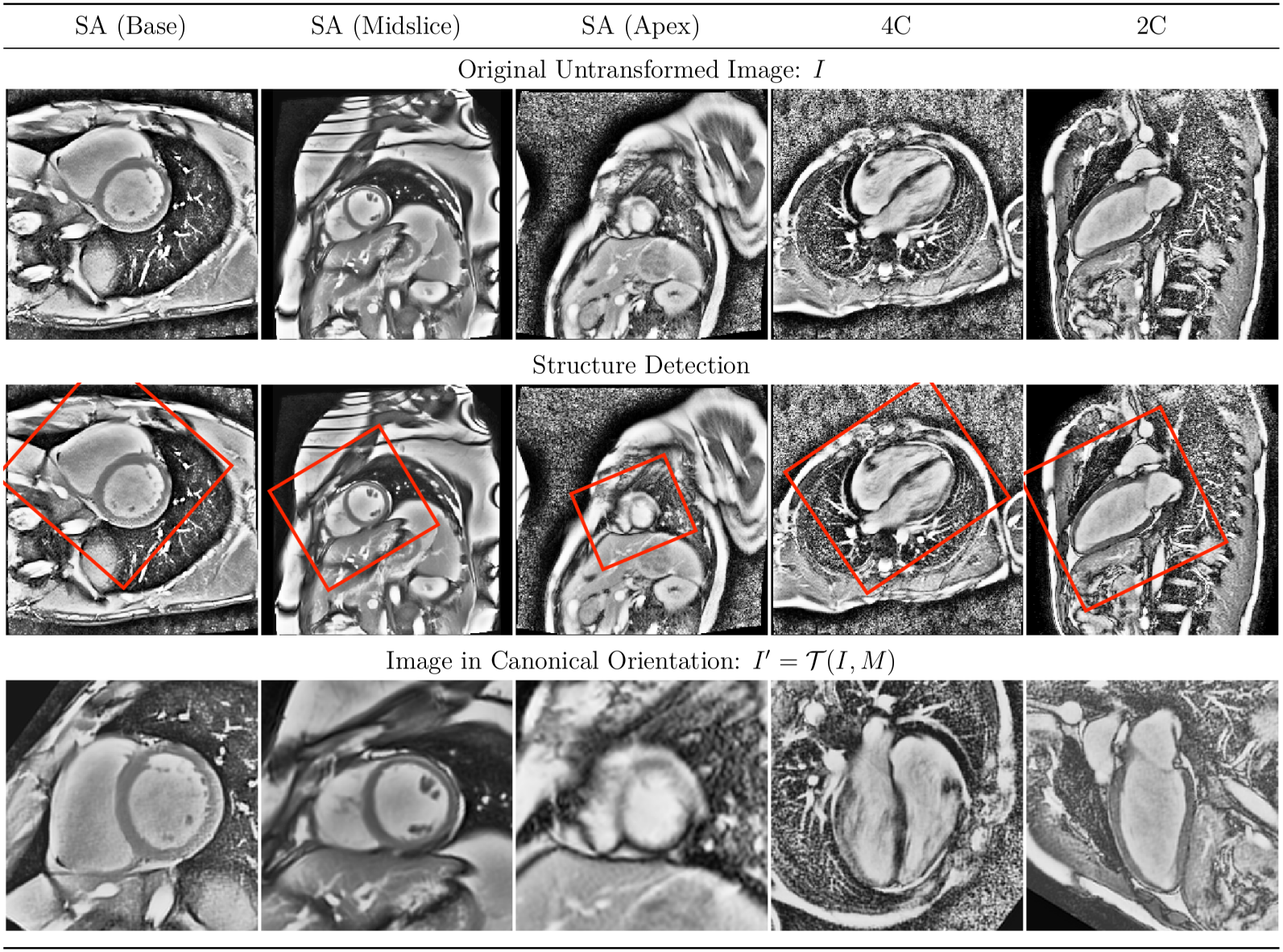


Figure 7.1: Orthogonal clinical views in canonical orientation. Representative short axis (SA), four-chamber (4C), and two-chamber (2C) images are shown as acquired (top), and having undergone rigid, affine transformation into a canonical orientation (bottom). Consistent with common clinical practice, the heart is rotated such that in the SA views, the right ventricle appears on the (radiological) right side of the image, whereas in the 4C and 2C views, the long axis of the left ventricle is oriented vertically. The heart is also centered and scaled to fill 90% of the image. Note the heterogeneity in size, orientation, and appearance of the heart in the untransformed images, which contributes to the difficulty of segmentation.

7.2 Method

The proposed model exploits a coarse-to-fine strategy for segmentation of cardiac SSFP images in an end-to-end differentiable CNNs framework, allowing for the detection, re-orientation and segmentation tasks to be codependent (Figure 7.2). It consists of three stages. *First*, the full-resolution, untransformed input image I undergoes coarse-grained segmentation using a U-Net module (§7.2.1). *Second*, the central (most downsampled) features of the aforementioned U-Net module are used to predict a rigid, affine matrix M capable of transforming I into a canonical orientation $I' = \mathcal{T}(I, M)$ (§7.2.2). *Third*, the transformed image I' is segmented using a stacked hourglass module (§7.2.3). In the following subsections, each component of the network is discussed in detail. In terms of notation, a superposed chevron (e.g., $\hat{x}$) indicates ground truth, and a superscript tick (e.g., x') indicates that the quantity pertains to the transformed data.

7.2.1 Coarse-grained Segmentation (U-Net) Module

The proposed network makes use of the U-Net architecture (Figure 7.3), which consists of a down-sampling path (left) followed by an up-sampling path (right) to restore the original spatial resolution. The downsampling path resembles the canonical classification CNN [1, 32], with two 3×3 convolutions, a rectified linear unit (ReLU) activation, and a 2×2 max pooling step repeatedly applied to the input image and feature maps. In the upsampling path, the reduction in spatial resolution is "undone" by performing 2×2 up-sampling, ReLU activation, and 3×3 convolution, eventually mapping the intermediate feature representation back to the original resolution. To provide accurate boundary localization, skip connections are used, where feature representations from the down-sampling path are concatenated with feature maps of the same resolution in the up-sampling path. Batch normalization [118], which has been shown to counteract gradient vanishing and to lead to better convergence, was performed between each pair of convolution and ReLU activation layers. The loss L_{S_U} for the U-Net module

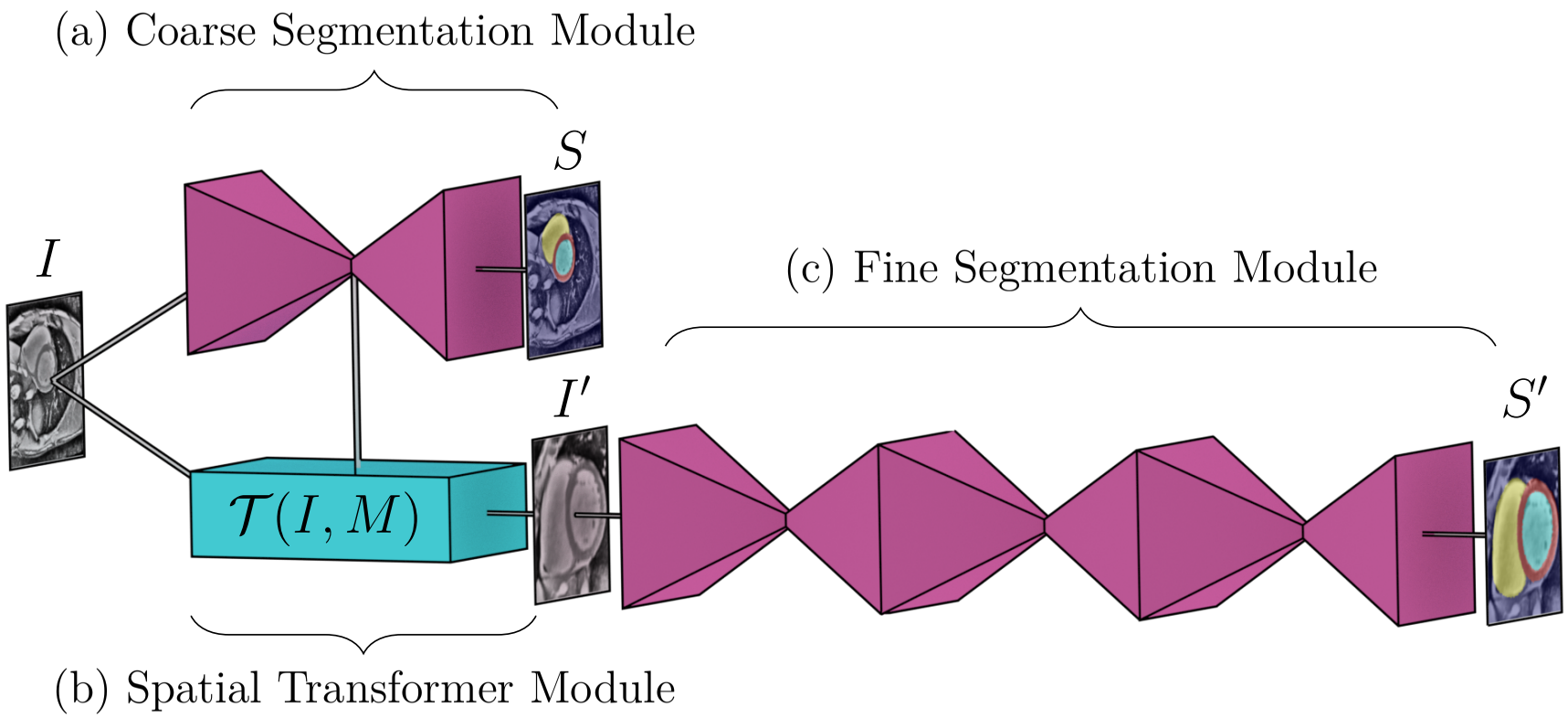


Figure 7.2: Overview of the ohm-net architecture. (a) The initial, unoriented SSFP image I is fed into a U-Net module, producing a coarse segmentation S. (b) The features from the central (most downsampled) layers of this U-Net are used by the attention module to predict the parameters M of a rigid, affine transformation and transform the input image into a cannonical orientation, $I' = \mathcal{T}(I, M)$. (c) This transformed image is fed into a stacked hourglass module to obtain a fine-grained segmentation in the canonical orientation S'. Note that, all modules shown are trained in an end-to-end way from scratch.

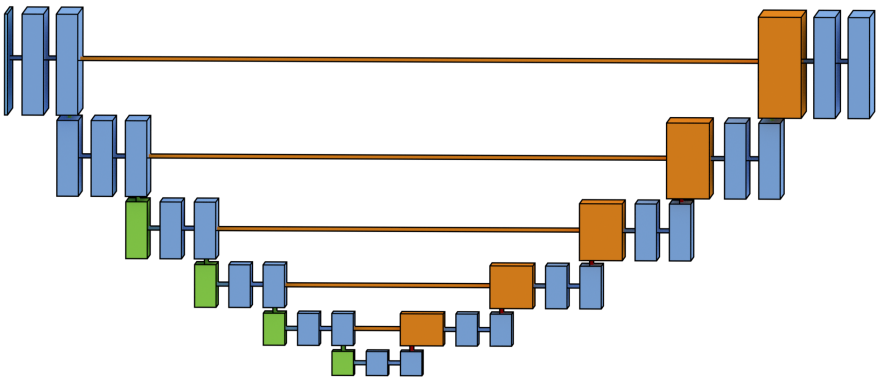


Figure 7.3: U-Net module. The input image is of size $256 \times 256 \times N$, where N is the number of channels (1 for all networks). Each blue and orange box corresponds to a multilingual feature map (orange indicates the result of a copy).

is the categorical cross entropy between the output of the softmax layer, P, and the ground truth segmentation, $\hat{S}$,

$$L_{S_U} = -\frac{1}{HW} \sum_{\forall h,w} \mathcal{H}(P_{h,w}, \hat{S}_{h,w}), \tag{7.1}$$

where

$$\mathcal{H}(x, \hat{x}) = -\hat{x} \log(x) + (1 - \hat{x}) \log(1 - x). \tag{7.2}$$

7.2.2 Geometric Transforamtion (STN)

In this section, similar idea as face alignment(§ 4.3) is used, where a localization network (LocNet) predicts a similarity transformation matrix, M; a grid generator, which implements the transform, $\mathcal{T}$; and a sampler, which implements the interpolation.

Localization network (LocNet) Intuitively, a human expert is able to provide translation, rotation, and scaling information given a rough segmentation of the heart. Based on this assumption, a small localization network (LocNet) is branched out from the layer immediately following the final max pooling step of the U-Net in order

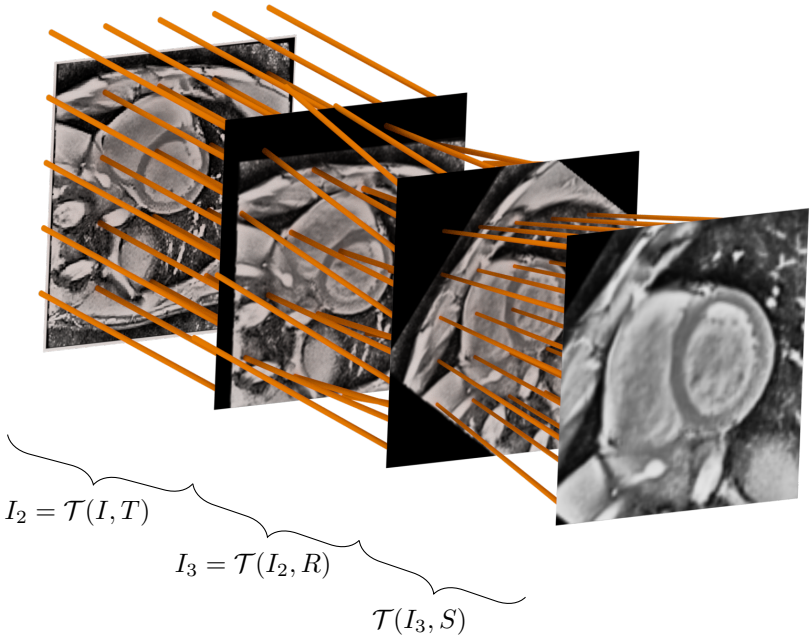


Figure 7.4: Detection with STN. Note that in the actual implementation, all transformations are performed relative to the input image I (i.e., $\mathcal{T}(I, T)$, $\mathcal{T}(I, RT)$, and $\mathcal{T}(I, SRT)$); for clarity, the transformations have been presented here as successive steps.

to predict the transformation parameters. As we have restricted the transformation to be rigid and affine, the affine matrix was decomposed into three separate matrices:

$$M = SRT,$$

where T is the translation matrix:

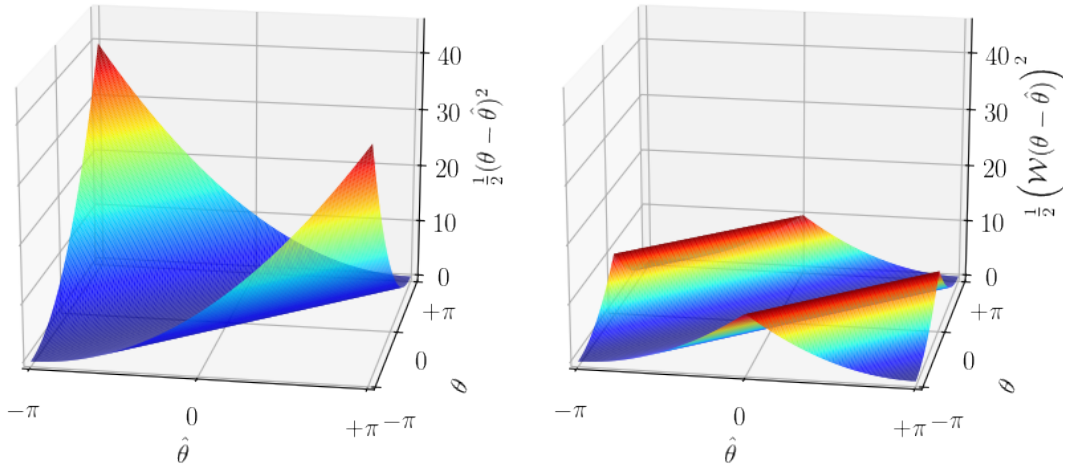
$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix};$$

R is the (counterclockwise) rotation matrix:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

and S is the (uniform) scaling matrix:

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$



(a) Mean Squared Error (MSE) **(b)** Mean Squared Wrapped Error (MSWE)

Figure 7.5: Mean squared error (MSE) vs mean squared wrapped error (MSWE).

Note that the images are defined on a normalized coordinate space $\{x, y\} \in [-1, +1]$, such that rotation and scaling occur relative to the image center.

In practice, the LocNet learns to predict only the relevant parameters, $\mathbf{m} = \begin{bmatrix} t_x & t_y & \theta & s \end{bmatrix}^\top$. During training, we explicitly provide the ground-truth transformation parameters $\hat{\mathbf{m}} = \begin{bmatrix} \hat{t_x} & \hat{t_y} & \hat{\theta} & \hat{s} \end{bmatrix}$, minimizing two types of losses, which are termed as *matrix losses* and *image losses*.

The matrix losses are regression losses between the ground truth and predicted parameters (L_{t_x}, L_{t_y}, L_θ, L_s). For scaling and translation, mean squared error (MSE) was used:

$$L_{t_x} = \frac{1}{2}(t_x - \hat{t_x})^2, \tag{7.3}$$

$$L_{t_y} = \frac{1}{2}(t_y - \hat{t_y})^2, \text{ and} \tag{7.4}$$

$$L_s = \frac{1}{2}(s - \hat{s})^2. \tag{7.5}$$

Naïve MSE is an inappropriate loss for regressing on θ given its periodicity; this can be understood intuitively by considering ground truth and predicted rotations of $\hat{\theta} = +\pi$ and $\theta = -\pi$, which yield a high MSE in spite of being synonymous. For this reason, a wrapped phase loss is used (mean squared wrapped

error MSWE, Figure 7.5), where $\theta - \hat{\theta}$ is wrapped into the range $[-\pi, \pi)$ prior to calculating the standard MSE,

$$L_\theta = \frac{1}{2}\left(\mathcal{W}(\theta - \hat{\theta})\right)^2, \tag{7.6}$$

and the wrapping operator $\mathcal{W}$ is defined as

$$\mathcal{W}(\cdot) = \mod(\cdot + \pi, 2\pi) - \pi.$$

Training the attention module based on these losses alone caused the network to overfit the training data somewhat. For this reason, we additionally regularized the model based on the MSE between the input image after translation, rotation, and scaling with the ground truth ($\hat{\mathbf{m}}$) and predicted ($\mathbf{m}$) transformation parameters:

$$L_{I_t} = \frac{1}{2}(\mathcal{T}(I, T) - \mathcal{T}(I, \hat{T}))^2, \tag{7.7}$$

$$L_{I_\theta} = \frac{1}{2}(\mathcal{T}(I, RT) - \mathcal{T}(I, \hat{R}\hat{T}))^2, \text{ and} \tag{7.8}$$

$$L_{I_s} = \frac{1}{2}(\mathcal{T}(I, SRT) - \mathcal{T}(I, \hat{S}\hat{R}\hat{T}))^2. \tag{7.9}$$

Grid generation and sampling For image sampler, we follow the same method as the original STNs paper (§ 2.4.6). The re-sampled image is then passed to the subsequent fine-grained segmentation modules.

7.2.3 Fine-grained Segmentation (Stacked Hourglass) Module

The output of the attention module, having been transformed into a canonical orientation, is then input into a stacked hourglass architecture. The hourglass consisted of $D = [1 \,..\, 3]$ U-Net modules in series with one another, each producing a segmentation $S_{H,d}$, where $d \in [1 \,..\, D]$. With reference to eq. (7.2), the categorical cross-entropy between the softmax output of the hourglass at depth d, $P^{H,d}_{h,w}$ and the (transformed) ground truth $\hat{S}'$ segmentations is calculated,

$$L_{S_{H,d}} = -\frac{1}{HW}\sum_{\forall h,w} \mathcal{H}(P^{H,d}_{h,w}\hat{S}'_{h,w}). \tag{7.10}$$

7.2.4 Summary

To summarize, we train the Ω-Net with one loss from the coarse-grained segmentation module, eq. (7.1); four matrix losses, eqs. (7.3) to (7.6), and three image losses, eqs. (7.7) to (7.9), from the attention module; and between one and three losses from the fine-grained segmentation module, eq. (7.10). Therefore, the overall loss function may be written:

$$\begin{aligned} L_\Omega = \alpha_1 L_{S_U} \\ + \alpha_2 (L_{t_x} + L_{t_y} + L_\theta + L_s) \\ + \alpha_3 (L_{I_t} + L_{I_\theta} + L_{I_s}) \\ + \alpha_4 \sum_{d=1}^{D} L_{S_{H,d}}, \end{aligned}$$

where $\alpha_1 = 100.0$, $\alpha_2 = 100.0$, $\alpha_3 = 0.1$, and $\alpha_4 = 1.0$. The architectures tested are summarized in Table 7.1.

7.3 Experiment

7.3.1 Data Preparation

The data set consisted of 63 subjects: 42 patients with overt hypertrophic cardiomyopathy (HCM) and 21 healthy control subjects. CMR was performed with a standardized protocol at 10 centers from 2009 to 2011. Nine centers used 1.5-T magnets, and one used a 3-T magnet. Where available, three SA (basal, equatorial, and apical), one 4C, and one 2C SSFP cine series were obtained.

The LV myocardium, and all four cardiac chambers were manually segmented in the SA, 4C, and 2C views (noting that not all classes are visible in the SA and 2C views). The papillary muscles and the trabeculation of the LV and RV were excluded from the myocardium.

Each volume was cropped or padded as appropriate to 256×256 pixels in the spatial dimensions, and varied from 20 to 50 frames in the time dimension. Nonuniform background illumination was corrected by dividing by an estimated illumination field, and background corrected images were histogram equalized.

Name	UNet 0	UNet 1	UNet 2	UNet 3	Parameters (Millions)
Network A	128	–	–	–	7.0
Network B	64	64	–	–	3.5
Network C	64	64	64	–	4.5
Network D	64	64	64	64	5.5

Table 7.1: CNNs architecture variants. Each layer in the UNet has same amount of channels, i.e. 64 or 128

Each individual image was normalized to zero mean and unit standard deviation before being input into the CNN.

7.3.2 Training and Cross-validation

For cross-validation, the subjects were partitioned into three folds of approximately equal size (4477, 4750, and 4625 images, respectively) such that the images from any one subject were present in one fold only. Each of the four architectures (Table 7.1) were trained on all three combinations of two folds and tested on the remaining fold.

The networks were initialized with orthogonal weights [96], and were optimized using Adam [24] by minimizing categorical cross-entropy. The learning rate was initialized to 0.001 and decayed by 0.1 every 26 epochs. To avoid over-fitting, data augmentation (translations and scaling $\pm 0.15\%$ of the image width; rotations ± 0.15 and a weight decay of 10^{-4} was used.

Besides these manual augmentation by transforming the input with small, rigid, affine transformations, it is worth noting that data augmentation is performed *implicitly* in the fine segmentation module by virtue of the fact that, in the early stages of training, the transformation parameters predicted by the LocNet are random.

7.3.3 Measure of Performance

Weighted foreground intersection-over-union (IoU) was calculated imagewise between the prediction and manual segmentations. For a binary image (one foreground class, one background class), IoU (also known as the Jaccard index) is defined for the ground truth and predicted images I_T and I_P as

$$IoU\left(I_T, I_P\right) = \frac{|I_T \cap I_P|}{|I_T \cup I_P|}, \tag{7.11}$$

noting that a small positive number should be added to the denominator in a practical implementation to avoid division by zero. To extend this concept to multiclass segmentation, IoU was calculated separately for each foreground class. A weighted sum of these five IoU values was then calculated, where the weights were given by the ratio between the relevant foreground class and the union of all foreground classes, yielding weighted, mean foreground IoU.

7.4 Result

7.4.1 Segmentation

Weighted foreground IoU was calculated separately for each image, and the median and interquartile range (IQR) of all predictions is reported. As accuracy is not necessarily the same across all clinical planes, the performance of the four networks relative to expert manual segmentation is reported for all views combined, and also for each clinical plane separately (Table 7.2).

It is instructive to examine intermediate network performance at each successive U-Net (Figure 7.6).

1) Although Network A contains the most parameters, adding the fine-grained segmentation module increases network performance *at the level of the coarse-grained U-Net* compared with Network A; i.e., the performance of the coarse-grained segmentation module U-Net (U-Net 0) is $\approx$ 0.007 higher in Networks B and C compared with Network A, and $\approx$ 0.003 higher in Network D compared with Networks B and C. 2) There is a substantial increase in performance between the coarse-grained and fine-grained segmentation U-Nets, i.e., U-Nets 0 and 1 ($\approx$ 0.016, $\approx$ 0.015, and $\approx$ 0.012 increases for Networks B, C, and D, respectively). 3) In Networks C and D, there is not a substantial increase in performance between successive U-Nets in the fine-grained segmentation module.

Name	View	U-Net 0	U-Net 1	U-Net 2	U-Net 3
Network A	All	0.834 [0.783, 0.871]	–	–	–
	SA	0.843 [0.789, 0.880]	–	–	–
	4C	0.819 [0.765, 0.855]	–	–	–
	2C	0.831 [0.788, 0.863]	–	–	–
Network B	All	0.841 [0.793, 0.876]	0.857 [0.819, 0.885]	–	–
	SA	0.848 [0.800, 0.884]	0.862 [0.820, 0.891]	–	–
	4C	0.831 [0.780, 0.861]	0.845 [0.812, 0.871]	–	–
	2C	0.832 [0.787, 0.864]	0.856 [0.822, 0.882]	–	–
Network C	All	0.841 [0.792, 0.875]	0.856 [0.816, 0.884]	0.857 [0.816, 0.885]	–
	SA	0.849 [0.797, 0.883]	0.862 [0.820, 0.890]	0.862 [0.819, 0.890]	–
	4C	0.830 [0.779, 0.861]	0.843 [0.804, 0.869]	0.844 [0.805, 0.869]	–
	2C	0.830 [0.793, 0.863]	0.855 [0.818, 0.883]	0.857 [0.819, 0.884]	–
Network D	All	0.844 [0.797, 0.877]	0.856 [0.819, 0.884]	0.858 [0.821, 0.886]	0.858 [0.821, 0.886]
	SA	0.851 [0.798, 0.886]	0.862 [0.821, 0.890]	0.863 [0.822, 0.892]	0.863 [0.822, 0.892]
	4C	0.832 [0.781, 0.861]	0.839 [0.805, 0.868]	0.843 [0.811, 0.870]	0.843 [0.811, 0.869]
	2C	0.842 [0.804, 0.869]	0.858 [0.828, 0.884]	0.860 [0.831, 0.886]	0.859 [0.830, 0.886]

Table 7.2: Network performance with reference to ground truth. Weighted foreground IoU of each network are calculated for all views combined and for each view separately. All four networks performed similarly. Network performance was highest in the SA view and lowest in the 4C view, though these differences are small.

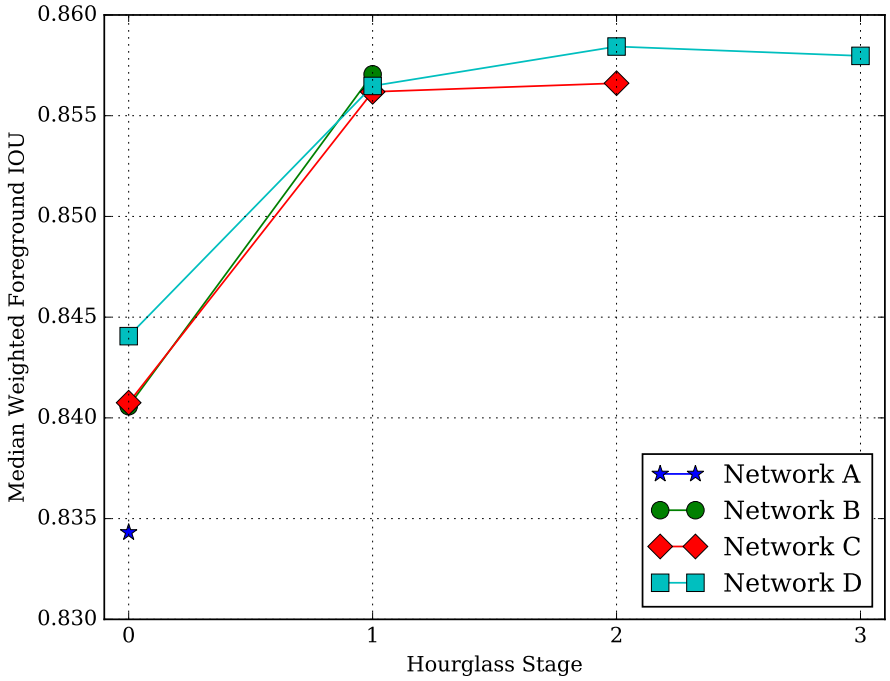


Figure 7.6: Weighted foreground IoU for architecture and depth.

As performance is likely to differ between structures, imagewise histograms of foreground IoU are plotted for the best performing network (Network D) for each structure and clinical plane (Figure 7.7). In all three clinical planes, performance is worst for the LV myocardium, best in the LV bloodpool, and intermediate in the remaining structures. Relatively poor LV myocardial segmentation performance can be understood intuitively by considering that segmentation error is concentrated primarily at the structure boundaries. Therefore, structures with a high ratio of perimeter-to-area (such as the LV myocardium, which has both an internal and external perimeter, i.e., endocardium and epicardium) are predisposed to perform poorly. A number of factors may contribute to the superior performance of LV bloodpool segmentation.

- The LV myocardium provides a high-contrast boundary along much of the perimeter of the LV bloodpool.

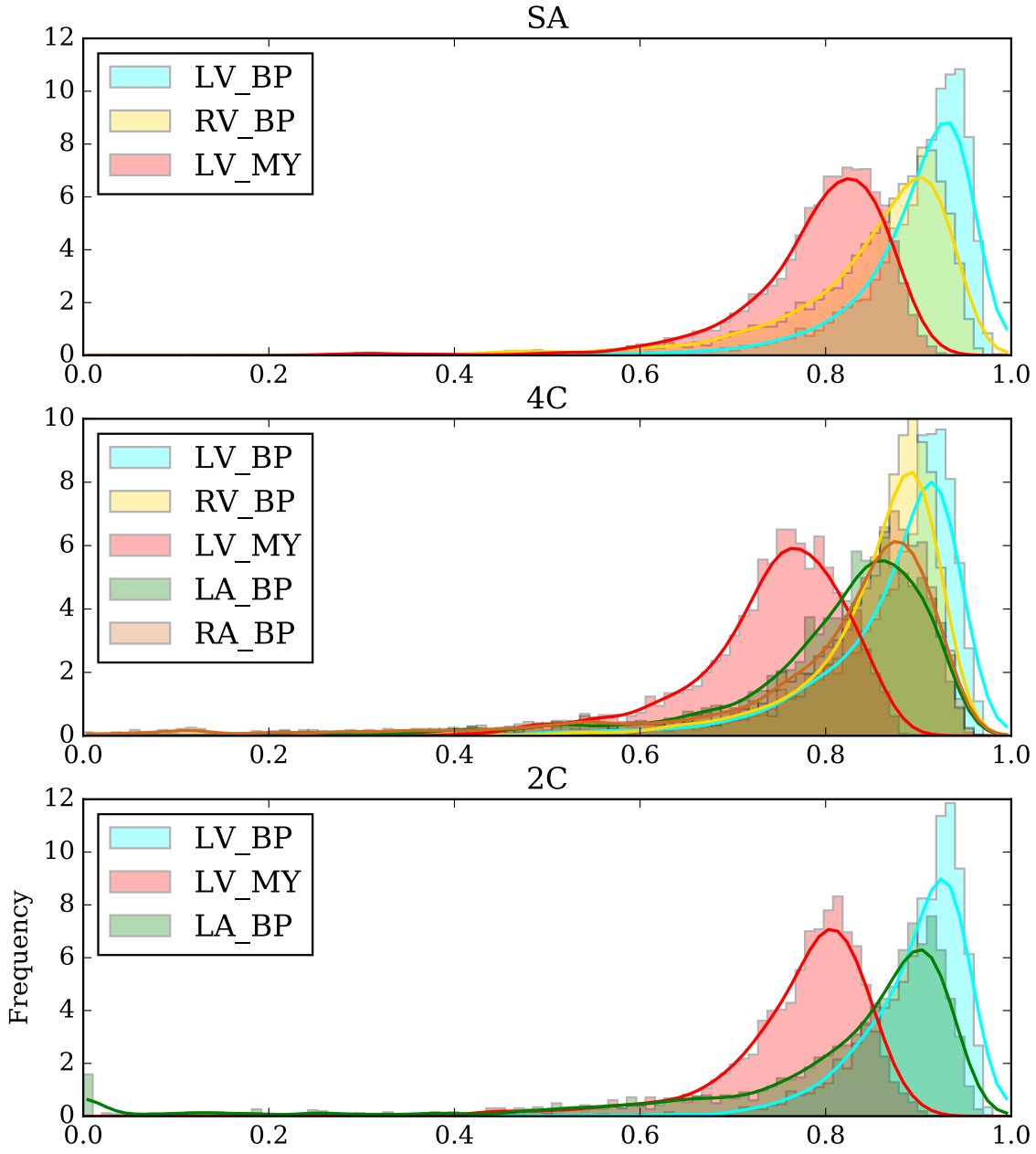


Figure 7.7: Histograms of IoU for each view and class. Performance relative to expert manual segmentation was highest in the SA view and lowest in the 4C view, though the differences are small.

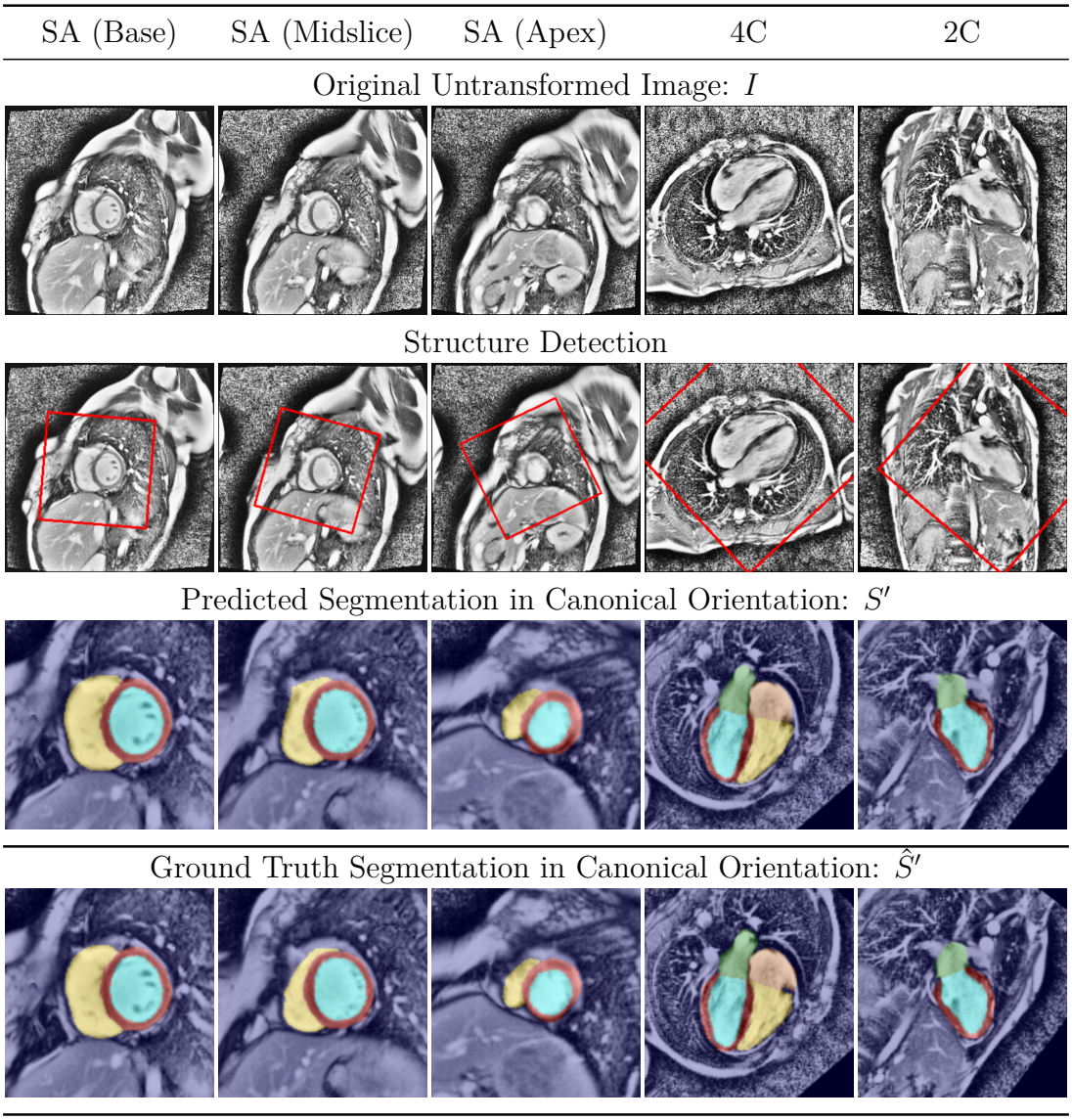


Figure 7.8: Representative segmentation results in healthy control subjects. See text for discussion.

- Compared with other cardiac chambers, the LV bloodpool has relatively less anatomical variation between subjects.
- The three orthogonal planes examined in this study are all defined relative to the left ventricle; therefore, the appearance if the LV is more consistent between subjects.

Representative segmentations produced by Network D in all views are shown for healthy control subjects in Figure 7.8 and for patients with overt HCM in Figure 7.9.

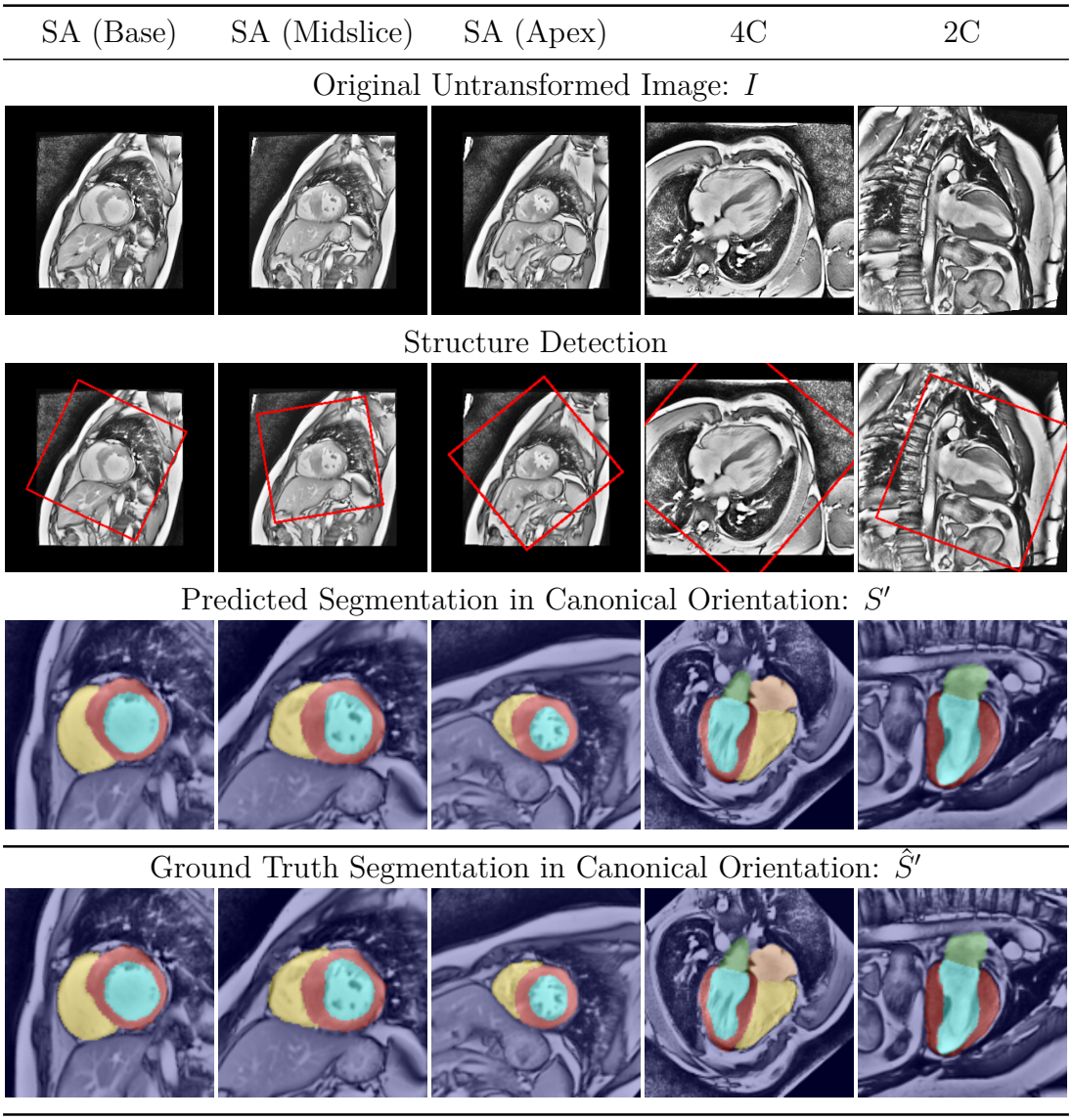


Figure 7.9: Representative segmentation results in patients with overt HCM. See text for discussion.

(Note that the ground truth segmentations have been transformed by the predicted parameters rather than the ground truth parameters in order to aid interpretation.) The network successfully transformed the images into the canonical orientation for all cases shown. Notably, the myocardial segmentation consistently excludes papillary muscles and myocardial trabeculation. Moreover, the network appears to reliably identify the atrioventricular valve plane in the long axis views, which is a useful result deserving of attention in future work.

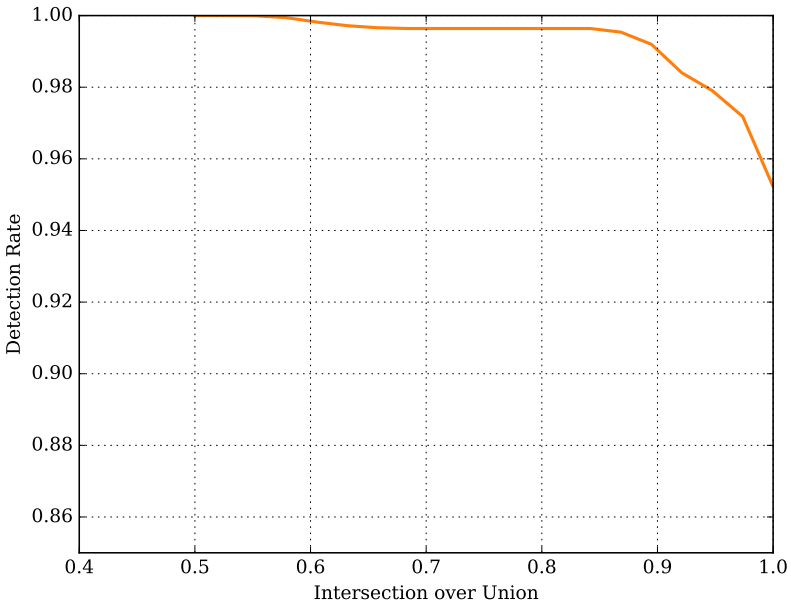


Figure 7.10: Precision-Recall curve for structure localization.

Fig. 7.10 illustrates the Ω-Net behavior on structure detection. Detection rate was defined as the percentage of cases in which weighted foreground IoU exceeded a varying threshold varying from 0.4 to 1.0, plotted on the horizontal axis. The resulting precision-recall curve had an excellent area under the curve (AUC) of 0.992, demonstrating the robustness of the Ω-Net's structure detection.

7.4.2 Transformation Parameters

Ground truth parameters were compared to those predicted by the best performing network (Network D) via correlation, and by Bland Altman plots Figure 7.11. It is notable that ground truth transformation parameters (particularly rotation and scale) were not uniformly distributed between views. Nonrandom rotation is to be expected from the fact that the positioning of the patient in the scanner, the protocol for determining imaging planes, the placement of the heart in the chest, and the relationship between imaging planes are all themselves nonrandom; nonrandom scale is likewise to be expected from the variable size of the anatomical structures visible in each view.

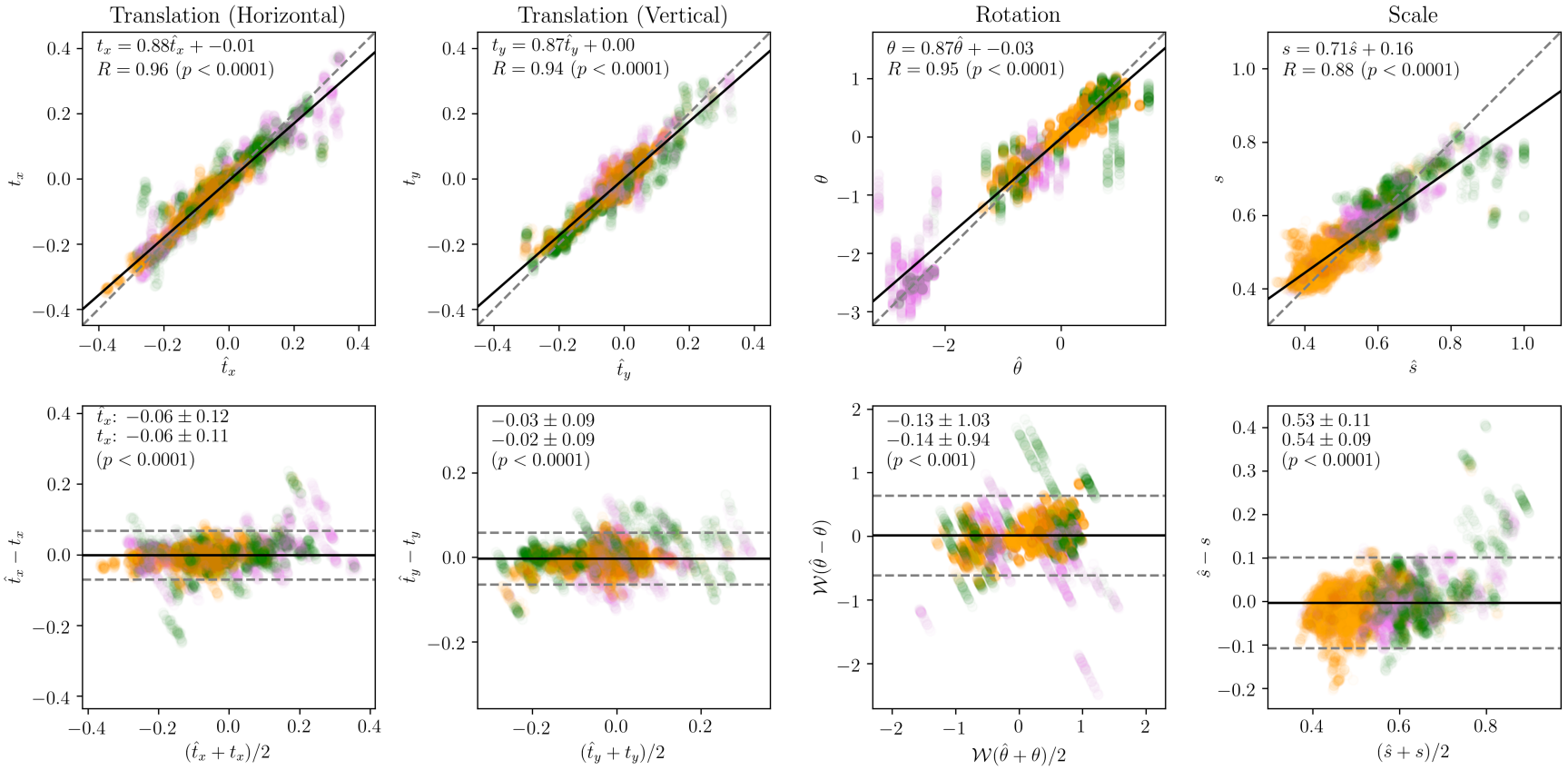


Figure 7.11: Transformation errors. Correlation (top) and Bland-Altman (bottom) plots comparing predicted and ground truth transformation parameters. SA, 4C, and 2C errors are represented by orange, green, and purple points, respectively (points have been rendered partially translucent to aid interpretation of densely occupied regions). In the correlation plots, the best-fit trendline is represented as a solid black line, and the ideal trendline ($y = x$) is represented as a dashed, gray line. The equation of the trendline and the Pearson correlation coefficient R are also given. In the Bland-Altman plots, the mean difference is represented as a solid black horizontal line, and the limits ± 1.96 standard deviations are represented as a dashed gray horizontal line.

Predicted horizontal translation, vertical translation, and rotation parameters were all highly correlated with ground truth ($R \approx 0.95$, $p < 0.0001$ for all), with the predicted parameters slightly underestimating the ground truth (slope ≈ 0.87 for all). No systematic bias was evident on visual inspection of the Bland-Altman plots; 95% of translation errors were within ± 0.07 (in normalized image coordinates), and 95% of rotation errors were within ± 0.63 (in radians). Of the 5% of cases which were outside these bounds, the vast majority were long axis (4C or 2C) views. This is perhaps unsurprising since each patient contributed three SA views, but only two long axis views.

Compared with translation and rotation, correlation between ground truth and predicted scale was slightly lower, though still good ($R = 0.88$, $p < 0.0001$); predicted scale again slightly underestimated ground truth scale ($s = 0.71\hat{s} + 0.16$). There is a marked decrease in network performance above approximately $\hat{s} = 0.7$. This may indicate the importance of context information to the network; however, it should be noted that the decrease in performance is accompanied by a sharp decrease in the frequency of cases, and so may also be the result of an insufficient number of samples in the dataset.

7.4.3 Failure Cases

Occasional failure cases were observed, a selection of which are shown in Fig. 7.12. Each of these failure cases has one or more features which could logically explain the failure. The leftmost column shows an apical SA slice from a severely hypertrophied patient. Patients with such severe disease were relatively uncommon in the dataset, perhaps causing the network to split its attention between the heart and a second "candidate structure" (the cardia of the stomach). The center-left column shows a second apical SA slice from a different subject, where the right ventricle was incorrectly segmented. The signal intensity in this image was low relative to the other patients in the cohort, resulting in a very high contrast image after histogram equalization. The center-right and rightmost columns show long axis views from a patient with a particularly high resolution scan, where the heart

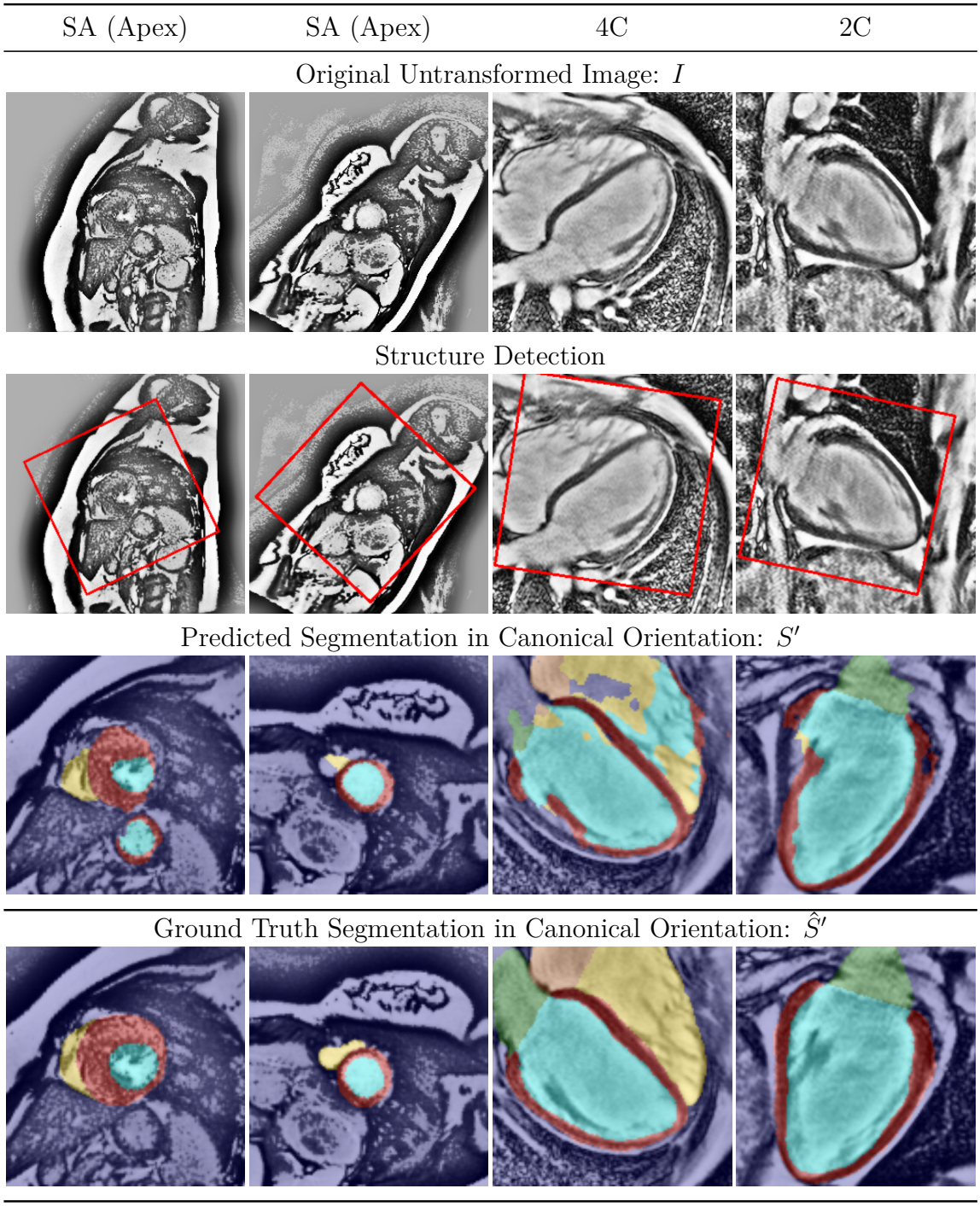


Figure 7.12: Selected failure cases from CNN segmentation. See text for discussion.

Structure	LV Bloodpool	RV Bloodpool	LV Myocardium
Jaccard Index (IoU)			
Ω-Net	**0.912**	**0.852**	0.803
[119]	0.896	0.832	**0.826**
[120]	0.869	0.784	0.775
Dice Coefficient			
Ω-Net	**0.954**	**0.920**	0.891
[119]	0.945	0.908	**0.905**
[120]	0.930	0.879	0.873

Table 7.3: Segmentation accuracy on the 2017 MICCIA ACDC dataset. Segmentation accuracy is reported as Dice coefficient in the ACDC challenge, but as IoU elsewhere in this work; therefore, both are reported here. (Note that Dice $= 2 * \text{IoU}/(1 + \text{IoU})$). Results are reported for the Network B variant of the Ω-Net; for the results by [120] published in STACOM; and for the same group's unpublished `arxiv.org` revision [119]. Boldface formatting indicates the best performing model for each foreground class.

occupies the vast majority of the image, with very little context information. In both cases, catastrophic segmentation error follows failure to properly reorient the image into a canonical orientation.

7.4.4 2017 MICCAI ACDC dataset

[120] represents the state-of-the-art network in terms of segmentation accuracy on the ACDC leaderboard; this same group has since released an unpublished revision[1] with improved results [119]. To match their methods, we retrained the Network B variant of Ω-Net from scratch using five-fold cross-validation on the provided dataset (each patient only appears in *one* fold). Single model segmentation accuracy is reported for Ω-Net, [120], and [119] in Table 7.3. Compared with [120], our results give higher IoU for all foreground classes: LV bloodpool (0.912 vs 0.869), RV bloodpool (0.852 vs 0.784), and LV myocardium (0.803 vs 0.775). Compared with [119], our results give higher IoU for LV bloodpool (0.912 vs 0.896) and RV bloodpool (0.852 vs 0.832), but lower IoU for LV myocardium (0.803 vs 0.826).

[1] https://arxiv.org/abs/1707.00587v2

7.5 Discussion

In this chapter, we have presented the Ω-Net: a novel deep convolutional neural network (CNN) architecture for detection, re-orientation, and segmentation. This network has been applied to the task of fully automatic whole-heart segmentation and simultaneous transformation into the "canonical" clinical view, which has the potential to greatly simplify downstream analyses of SSFP CMR images. Five foreground classes in three clinical views can be predicted, which is a substantially more difficult problem than has been addressed previously in the literature. To transform each view into the canonical orientation, we applied a varied version of spatial transformer network module [40], eventually, the transformed image is provided as input to several cascaded fine-grained segmentation module (U-Nets 1, 2, and 3). The network was trained end-to-end from scratch to segment five foreground classes (the four cardiac chambers plus the LV myocardium) in three views (SA, 4C, and 2C), without providing prior knowledge of the view being segmented. The dataset was highly heterogeneous from the standpoint of anatomical variation, including both healthy subjects and patients with overt hypertrophic cardiomyopathy. Data was acquired from both 1.5-T and 3-T magnets as part of a multicenter trial involving 10 institutions. In cross-validation experiments, the network performed well in predicting both the parameters of the transformation, and the cardiac segmentation. Comparing with the state-of-the-art methods, we retrained the Ω-Net from scratch using five-fold cross-validation on the public benchmarks, and we outperform previous best results on two of the three structures of interest.

8 Summary and Future Work

In this thesis, we discussed several different CNNs architecture designs for a variety of applications in computer vision and biomedical image analysis. Our research focus was on incorporating human knowledge into the Deep Learning models, either introducing differentiable layers to the model, or decomposing a complex problems into simple ones that can be solved easily.

To conclude, we summarise the contributions of this thesis and comment on some possible areas for future work. This chapter is organized according to the areas of research covered throughout Chapters 3-7.

8.1 Semantic Segmentation in Natural Images

In § 3, we presented a novel layer for learning multi-scale contextual information in image segmentation, termed Layer Recurrent Neural Networks. By introducing the within-layer recurrence, local information is aggregated together into more global contexts, and enable the model to learn the receptive field adaptively. We also show that the new layer can be initialized into any pre-trained CNNs as an identity function, and later fine-tuned as the task required, this is shown to always improve the semantic segmentation.

8.1.1 Extensions

Since the first proposal of Fully Convolutional Networks (FCNs), the performance (in terms of Mean Intersection over Union, or Jaccard Index) of semantic segmentation has been improved dramatically on the Pascal VOC 2012, from 62.2 (FCNs) to 86.9 [121]. Apart from the advancements of CNNs architectures, e.g. from AlexNet to VGGNet, to ResNet, these achievements are mainly driven by the large-scale datasets, for instance the new state-of-the-art model (DeepLabv3-JFT [121]) is pre-trained on the JFT-300M dataset [122], which contains over 300M images with noisy labels. In terms of future work, first, it would be valuable to build pipelines that enable automatic dataset collection with minimum human interaction. Second, training models with fewer supervision (e.g. only image-level top-5 annotation) would be of great interest. For instance, recent works have considered training image classification models with attentions [123], which could potentially be used as an intialization for traditional segmentation methods, e.g. graph cuts. Third, video objects segmentation [124] is gaining popularity, as it naturally poses more challenges on the algorithm efficiency and encoding temporal information.

8.2 Face Recognition

In § 4, we first presented how to achieve face in-plane alignment with only image-level annotations, secondly, we described a novel architecture for fine-grained face verification between templates, termed as Comparator Networks. The model was designed to predict if two templates (each may contain variable number of images or frames) belong to the same person or not. In addition to the global face descriptors, descriptors for local parts (eyes, noses, mouths) are pooled out by attention mechanism, and compared with local "experts". Meanwhile, face poses are also decomposed into frontal, facing left, facing right, further enabling the model to compare local parts based on face poses.

8.2.1 Extensions

In terms of future work, two directions could be interesting to explore. First, Thewlis *et al.* [80, 125] showed the possibility to learn facial landmarks in an unsupervised way. Their method could potentially be used as an alternative regularization in the comparator networks for training the local part attentions. Second, given the pre-aligned faces, simpler pooling method might be investigated, for instance, naïve horizontal or vertical pooling [126].

8.3 Microscopy Cell Counting & Detection

In § 5, we presented the first Fully Convolutional Regression Networks (FCRNs) in the biomedical image analysis community. We show that cell counting and detection can be tackled by training a mapping from raw input image to the density maps. Our presented methods achieved state-of-the-art performance on cell counting dataset at the time of publication of the respective publications. Together, it has been cited by more than 70 publications (till 19th August, 2018).

8.3.1 Extensions

Following our publication [7], several works have been proposed to further improve our FCRNs. For instance, Xie *et al.* [127] replaced our network backbone by residual networks. In terms of the future work, it could be interesting to explore interactive objects counting [83], where human interaction is introduced into the pipeline. Potentially this may improve the algorithm performance on extreme cases, such as testing data is very different from training data.

8.4 Standardized Alignment in 3D Fetal Neurosonography

In § 6, we presented a multi-task CNNs to transform 3D ultrasound fetal brain to a common reference coordinates. We achieved this by decomposing the complex transformations into several simple ones, which can be easily tackled by CNNs,

e.g. brain segmentation, eye localization, brain orientation, etc. Experiments show that the model is consistent and generalizable to a wide distribution of gestational ages. In a clinical setting, this may assist in the navigation through the brain volume and establishing coordinate locations of key cerebral landmarks, and allow neurosonographers to localize the standard biometry planes from which to extract 2D measurements for fetal growth monitoring, and to extract oblique planes for neuropathological assessment.

8.4.1 Extensions

As future work, we plan to apply this model on a very large dataset with over $8,000$ volumes. To our knowledge, this is the largest 3D ultrasound fetal brain dataset [1]. After fixing the failure cases, we plan to train an end-to-end model with the 3D Spatial Transformer Networks and directly transform the input volumes to a common reference coordinates, the resulting transformed data will then be made available for clinical research. Apart from the standardized re-orientation, we are also investigating the CNN-based registration, by that we mean, the features for registration should be learnt from data in an end-to-end way.

8.5 MR Cardiac Video Segmentation

In § 7, we proposed the Ω-Net (Omega-Net), a novel CNNs architecture trained end-to-end to tackle three important tasks: detection, transformation into a canonical orientation, and segmentation. We demonstrated that the model is capable of fully automatic segmentation of five foreground classes (LV myocardium, the left and right atria, and the left and right ventricles) in three orthogonal clinical planes (short axis, SA; four-chamber, 4C; and two-chamber, 2C), with simultaneous rigid transformation of the input into a canonical orientation based on detection. Moreover, the model is successfully trained and evaluated on a multi-center population of patients with hypertrophic cardiomyopathy (HCM),

[1] www.intergrowth21.org.uk

which shows the model can be generalizable to the highly variable appearance of the LV in these patients.

8.5.1 Extensions

As for future work, a variety of opportunities present themselves in terms of optimizing the Ω-Net architecture. For example, the network was trained to segment individual image frames, without spatial or temporal context; modifying the architecture to allow information sharing between temporal frames and spatial slices has the potential to increase accuracy and consistency. The E-Net ("Efficient Net") provides modifications to the U-Net blocks which increase computational and memory efficiency, while preserving accuracy [128]; these lessons have been applied successfully to cardiac segmentation [116], and could theoretically be applied here as well.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2012).

[2] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[3] Maxwell Stinchcombe and Kurt Hornik and Halbert White. "Multilayer Feedforward Networks Are Universal". In: *Neural Networks* (1989).

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2016).

[5] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common objects in context". In: (2014). URL: `https://arxiv.org/abs/1405.0312`.

[6] Mark Everingham, Luc Van Gool, Christopher K I Williams, and John Winn. "The PASCAL Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* (2010).

[7] Weidi Xie, J. Alison Noble, and Andrew Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks". In: *MICCAI Workshop* (2015).

[8] Weidi Xie, J. Alison Noble, and Andrew Zisserman. "Microscopy Cell Counting And Detection with Fully Convolutional Regression Networks". In: *Computer Methods in Biomechanics and Biomedical Engineering : Imaging & Visualization* (2016).

[9] Weidi Xie, J. Alison Noble, and Andrew Zisserman. "Layer Recurrent Neural Networks". In: 2016. URL: `https://openreview.net/pdf?id=rJJRDvcex`.

[10] Davis M. Vigneault, Weidi Xie, David A. Bluemke, and J. Alison Noble. "Feature tracking cardiac magnetic resonance via deep learning and spline optimization". In: *Proceedings of the International Conference on Functional Imaging and Modeling of the Heart (FIMH)* (2017).

[11] Yipeng Hu, Eli Gibson, Li-Lin Lee, Weidi Xie, Dean Barratt, Tom Vercauteren, and J. Alison Noble. "Freehand Ultrasound Image Simulation with Spatially-conditioned Generative Adversarial Networks". In: *MICCAI Workshop on Reconstruction and Analysis of Moving Body Organs (2017)* (2017).

[12] Ana I L Namburete, Weidi Xie, and J. Alison Noble. "Robust Regression of Brain Maturation from 3D Fetal Neurosonography using CRNs". In: *MICCAI Workshop on Fetal and InFant Image Analysis (2017)* (2017).

[13] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. "VGGFace2: A dataset for recognising faces across pose and age". In: *Automatic Face & Gesture Recognition (2018)*. 2018. URL: `http://arxiv.org/abs/1710.08092`.

[14] Davis M. Vigneault, Weidi Xie, Y. Carolyn Ho, David A. Bluemke, and J. Alison Noble. "Ω-Net: Fully Automatic, Multi-View Cardiac MR Detection, Orientation, and Segmentation with Deep Neural Networks". In: *Medical Image Analysis* (2018).

[15] Ruobing Huang, Weidi Xie, and J. Alison Noble. "VP-Nets : Efficient Automatic Localization of Key Brain Structures in 3D Fetal Neurosonography". In: *Medical Image Analysis* (2018).

[16] Ana I L Namburete, Weidi Xie, Mohammad Yaqub, Andrew Zisserman, and J. Alison Noble. "Fully-Automated Alignment of 3D Fetal Brain Ultrasound to A Canonical Reference Space Using Multi-task Learning". In: *Medical Image Analysis* (2018).

[17] Weidi Xie, Li Shen, and Andrew Zisserman. "Comparator Networks". In: *European Conference of Computer Vision (2018)*. 2018. URL: `https://arxiv.org/abs/1807.11440`.

[18] Weidi Xie and Andrew Zisserman. "Multicolumn Networks on Face Recognition". In: *British Machine Vision Conference (2018)*. 2018. URL: `https://arxiv.org/abs/1807.09192`.

[19] León Bottou. "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010* (2010), pp. 177–186.

[20] Ning Qian. "On the Momentum Term in Gradient Descent Learning Algorithms The Momentum Term in Gradient Descent". In: *Neural Networks* 5213.12(1) (1999), pp. 145–151.

[21] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[22] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: (2012). URL: `http://arxiv.org/abs/1212.5701`.

[23] Geoffrey Hinton and Ni Srivastava. "RMSProp Lecture Notes". In: *Lecture Notes.* ().

[24] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015).

[25] Gabriel Goh. "Why Momentum Really Works." In: *Distill* 2.4 (2017), pp. 1–24. URL: `http://distill.pub/2017/momentum`.

[26] Kunihiko Fukushima. "Neocognitron-A Self-organizing Neural Network Model For A Mechanism Of Pattern Recognition Unaffected By Shift In Position". In: *Biological Cybernetics* (1980).

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition.* 1989.

[28] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based Learning Applied To Document Recognition". In: *Proceedings of the IEEE* (1998).

[29] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. "An Analysis of Deep Neural Network Models for Practical Applications". In: (2016).

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* (2014).

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Chapel Hill, and Ann Arbor. "Going Deeper with Convolutions". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[32] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations (ICLR)* (2015).

[33] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network". In: (2013). URL: `http://arxiv.org/abs/1312.4400`.

[34] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. "Deeply-Supervised Nets". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2014).

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[36] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks". In: *International Conference on Machine Learning (ICML) workshop* (2015).

[37] Evan Shelhamer, Jonathan Long, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).

[38] Richard Zhang, Phillip Isola, and Alexei A. Efros. “Colorful image colorization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2016).

[39] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. “Context Encoders: Feature Learning by Inpainting”. In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2016).

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. “Spatial Transformer Networks”. In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2015).

[41] Sepp Hochreiter and J{\"u}rgen Schmidhuber. “Long-short Term Memory”. In: *Neural computation* (1997).

[42] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *NIPS Deep Learning and Representation Learning Workshop* (2014).

[43] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015).

[45] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2016).

[46] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *International Conference on Learning Representations (ICLR)* (2015).

[47] Ming Liang, Xiaolin Hu, and Bo Zhang. “Convolutional Neural Networks with Intra-layer Recurrent Connections for Scene Labeling”. In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2015).

[48] Pho Pinheiro and Ronan Collobert. “Recurrent convolutional neural networks for scene labeling”. In: *Proceedings of the International Conference on Machine Learning (ICML)* (2014).

[49] Shuai Zheng, Sadeep Jayasumana, B. Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H S Torr. “Conditional

Random Fields as Recurrent Neural Networks". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2015).

[50] Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross Girshick. "Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[51] Sifei Liu, Jinshan Pan, and Ming Hsuan Yang. "Learning Recursive Filters For Low-level Vision Via A Hybrid Neural Network". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2016).

[52] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. "Simultaneous Detection and Segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. (2014).

[53] Philipp Krähenbühl and Vladlen Koltun. "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2012).

[54] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition". In: *Proceedings of the British Machine Vision Conference (BMVC)* (2015).

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2015).

[56] Yaniv Taigman, Ming Yang, Marc Aurelio Ranzato, and Lior Wolf. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2014).

[57] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. "MS-Celeb-1M: A Dataset and Benchmark for Large Scale Face Recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2016), pp. 1–17.

[58] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. "Labeled faces in the wild: A database for studying face recognition in unconstrained environments". In: *Advances in Face Detection and Facial Image Analysis* 1 (2007), pp. 07–49.

[59] Lior Wolf, Tal Hassner, and Itay Maoz. "Face recognition in unconstrained videos with matched background similarity". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 529–534.

[60] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Patrick Grother, Alan Mah, and Kristen Allen. "Pushing the Frontiers of Unconstrained Face Detection and Recognition : IARPA Janus Benchmark A". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2015).

[61] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn Adams, Tim Miller, Nathan Kalka, Anil K. Jain, James A. Duncan, Kristen Allen, Jordan Cheney, and Patrick Grother. "IARPA Janus Benchmark-B Face Dataset". In: *Conference on Computer Vision and Pattern Recognition Workshops* (2017).

[62] Haoxiang Li, Gang Hua, Zhe Lin, Jonathan Brandt, and Jianchao Yang. "Probabilistic elastic matching for pose variant face verification". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2013).

[63] Omkar M Parkhi, Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, and Visual Geometry Group. "A Compact and Discriminative Face Track Descriptor". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2014).

[64] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. "Joint Face Detection and Alignment using Multi - task Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* (2016).

[65] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[66] Brianna Maze, Charles Otto, Jocelyn Adams, James A Duncan, Patrick Grother, Nathan Kalka, Tim Miller, W Tyler Niggel, Janet Anderson, and Jordan Cheney. "IARPA Janus Benchmark-C: Face Dataset and Protocol". In: *11th IAPR International Conference on Biometrics.(2018)* (2018).

[67] Hakan Cevikalp, William Triggs, Hakan Cevikalp, William Triggs, Face Recognition, Image Sets, Cvpr Ieee, Hakan Cevikalp, Bill Triggs, Laboratoire Jean Kuntzmann, and Grenoble Cedex. "Face Recognition Based on Image Sets." In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010).

[68] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. "Object Retrieval with Large Vocabularies and Fast Spatial Matching". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2007).

[69] Luan Tran, Xi Yin, and Xiaoming Liu. "Disentangled Representation Learning GAN for Pose-Invariant Face Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[70] Jiaolong Yang, Peiran Ren, Dongqing Zhang, Dong Chen, Fang Wen, Hongdong Li, and Gang Hua. "Neural Aggregation Network for Video Face Recognition". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2016).

[71] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *International Conference on Learning Representations (ICLR)* (2015).

[72] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. "Multiple Object Recognition with Visual Attention". In: *International Conference on Learning Representations (ICLR)* (2015).

[73] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of the International Conference on Machine Learning (ICML)* (2015).

[74] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. "Learning Multi-attention Convolutional Neural Network for Fine-Grained Image Recognition". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2017).

[75] Wanglong Wu, Meina Kan, Xin Liu, Yi Yang, Shiguang Shan, and Xilin Chen. "Recursive Spatial Transformer (ReST) for Alignment-Free Face Recognition". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2017).

[76] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. "A simple neural network module for relational reasoning". In: *Neural Information Processing Systems (NIPS)* (2017).

[77] Tsung Yu Lin, Aruni Roychowdhury, and Subhransu Maji. "Bilinear CNN models for fine-grained visual recognition". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2015).

[78] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. "Matching Networks for One Shot Learning". In: *Neural Information Processing Systems (NIPS)* (2016).

[79] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. "Learning to Compare: Relation Network for Few-Shot Learning". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2018).

[80] James Thewlis, Hakan Bilen, and Andrea Vedaldi. "Unsupervised learning of object landmarks by factorized spatial embeddings". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2017). arXiv: 1705.02193. URL: http://arxiv.org/abs/1705.02193.

[81] Navaneeth Bodla, Jingxiao Zheng, Hongyu Xu, Jun Cheng Chen, Carlos Castillo, and Rama Chellappa. "Deep heterogeneous feature fusion for template-based face recognition". In: *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)* (2017).

[82] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. "Learning To Detect Cells Using Non-overlapping Extremal Regions". In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2012).

[83] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. "Interactive Object Counting". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2014).

[84] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. "Detecting Overlapping Instances In Microscopy Images Using Extremal Region Trees". In: *Medical Image Analysis* (2016).

[85] Luca Fiaschi, Rahul Nair, Ullrich Koethe, and Fred a Hamprecht. "Learning to Count with Regression Forest and Structured Labels". In: *Proceedings of the International Conference on Pattern Recognition (ICPR)* (2012).

[86] D C Ciresan, A Giusti, L M Gambardella, and J Schmidhuber. "Mitosis Detection in Breast Cancer Histology Images using Deep Neural Networks". In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2013).

[87] Dc Ciresan, Alessandro Giusti, Lm Gambardella, and J Schmidhuber. "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2012).

[88] Feng Ning, Damien Delhomme, Yann LeCun, Fabio Piano, Léon Bottou, and Paolo Emilio Barbano. "Toward Automatic Phenotyping Of Developing Embryos From Videos". In: *IEEE Transactions on Image Processing* (2005).

[89] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[90] Victor Lempitsky and Andrew Zisserman. "Learning To Count Objects in Images". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2010).

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2014).

[92] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).

[93] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *International Conference on Learning Representations (ICLR)* (2014).

[94] Tomas Pfister, James Charles, and Andrew Zisserman. "Flowing ConvNets for Human Pose Estimation in Videos". In: *Proceedings of the International Conference on Computer Vision (ICCV)* (2015).

[95] Yuanpu Xie, Fuyong Xing, Xiaoshuang Shi, Xiangfei Kong, Hai Su, and Lin Yang. "Efficient and robust cell detection: A structured regression approach". In: *Medical Image Analysis* (2017).

[96] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *International Conference on Learning Representations (ICLR)* (2014).

[97] S Panda-Jonas, J B Jonas, and M Jakobczyk-Zmija. "Retinal Pigment Epithelial Cell Count, Distribution, And Correlations In Normal Human Eyes". In: *Americal Journal of Ophthalmology* (1996).

[98] Geisa M. Faustino, Marcelo Gattass, Stevens Rehen, and Carlos J. P. de Lucena. "Automatic embryonic stem cells detection and counting method in fluorescence microscopy images". In: *Proceedings of the International Symposium on Biomedical Imaging (ISBI)* (2009).

[99] Phililp Went, S. Mayer, N. Oberholzer, and S. Dirnhofer. "Plasma cell quantification in bone marrow by computer-assisted image analysis". In: *Histology and Histopathology* (2006).

[100] Aravindh Mahendran and Andrea Vedaldi. "Understanding Deep Image Representations by Inverting Them". In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[101] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. "Densely Connected Convolutional Networks". In: *Proceedings of the Computer Vision and Pattern Recognition (CVPR)* (2016).

[102] P.H.S. Torr and A. Zisserman. "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry". In: *Computer Vision and Image Understanding* (2000).

[103] Ana I L Namburete, Richard V. Stebbing, Bryn Kemp, Mohammad Yaqub, Aris T. Papageorghiou, and J. Alison Noble. "Learning-based prediction of gestational age from ultrasound images of the fetal brain". In: *Medical Image Analysis* (2015).

[104] Chiung Hsin Chang, Chen Hsiang Yu, Fong Ming Chang, Huei Chen Ko, and Hsi Y. Chen. "The assessment of normal fetal brain volume by 3-D ultrasound". In: *Ultrasound in Medicine and Biology* (2003).

[105] Nanette M. Roelfsema, Wim C J Hop, Simona M E Boito, and Juriy W. Wladimiroff. "Three-dimensional sonographic measurement of normal fetal brain volume during the second half of pregnancy". In: *American Journal of Obstetrics and Gynecology.* (2004).

[106] Joo Hyun Song, Gary E. Christensen, Jeffrey A. Hawley, Ying Wei, and Jon G. Kuhl. "Evaluating image registration using NIREP". In: *International Workshop on Biomedical Image Registration* (2010).

[107] A. Monteagudo, I. E. Timor-Tritsch, and P. Mayberry. "Three-dimensional transvaginal neurosonography of the fetal brain: 'Navigating' in the volume scan". In: *Ultrasound in Obstetrics and Gynecology* (2000).

[108] Mohammad Yaqub, Sylvia Rueda, Anil Kopuri, Pedro Melo, A. T. Papageorghiou, Peter B. Sullivan, Kenneth McCormick, and J. Alison Noble. "Plane Localization in 3-D Fetal Neurosonography for Longitudinal Analysis of the Developing Brain". In: *IEEE Journal of Biomedical and Health Informatics* (2016).

[109] Dario Paladini, Gustavo Malinger, Ana Monteagudo, Gianluigi Pilu, Ilan Timor-Tritsch, and Ants Toi. "Sonographic examination of the fetal central nervous system: Guidelines for performing the 'basic examination' and the 'fetal neurosonogram'". In: *Ultrasound in Obstetrics and Gynecology* (2007).

[110] Peng Peng, Karim Lekadir, Ali Gooya, Ling Shao, Steffen E. Petersen, and Alejandro F. Frangi. "A review of heart chamber segmentation for structural and functional analysis using cardiac magnetic resonance imaging". In: *MAGMA* (2016).

[111] Li Kuo Tan, Yih Miin Liew, Einly Lim, and Robert A. McLaughlin. "Cardiac left ventricle segmentation using convolutional neural network regression". In: *IEEE EMBS Conference on Biomedical Engineering and Sciences* (2016).

[112] Rudra P K Poudel, Pablo Lamata, and Giovanni Montana. "Recurrent Fully Convolutional Neural Networks for Multi-slice MRI Cardiac Segmentation". In: *MICCAI RAMBO Workshop* (2016).

[113] Li Kuo Tan, Yih Miin Liew, Einly Lim, and Robert A. McLaughlin. "Convolutional neural network regression for short-axis left ventricle segmentation in cardiac cine MR sequences". In: *Medical Image Analysis* (2017).

[114] Gongning Luo, Ran An, Kuanquan Wang, Suyu Dong, and Henggui Zhang. "A Deep Learning Network for Right Ventricle Segmentation in Short-Axis MRI". In: *Computing in Cardiology* (2016).

[115] Phi Vu Tran. "A Fully Convolutional Neural Network for Cardiac Segmentation in Short-Axis MRI". In: (2016). URL: `http://arxiv.org/abs/1604.00494`.

[116] Jesse Lieman-Sifry, Matthieu Le, Felix Lau, Sean Sall, and Daniel Golden. "Fastventricle: Cardiac segmentation with ENet". In: *Proceedings of the International Conference on Functional Imaging and Modeling of the Heart (FIMH)* (2017).

[117] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2016).

[118] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the International Conference on Machine Learning (ICML)* (2015).

[119] Fabian Isensee, Paul F. Jaeger, Peter M. Full, Ivo Wolf, Sandy Engelhardt, and Klaus H. Maier-Hein. "Automatic cardiac disease assessment on cine-MRI via time-series segmentation and domain specific features". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2018).

[120] Fabian Isensee, Paul F. Jaeger, Peter M. Full, Ivo Wolf, Sandy Engelhardt, and Klaus H. Maier-Hein. "Automatic cardiac disease assessment on cine-MRI via time-series segmentation and domain specific features". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2018).

[121] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. "Rethinking Atrous Convolution for Semantic Image Segmentation". In: (2017). URL: `http://arxiv.org/abs/1706.05587`.

[122] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: (2017). URL: `http://arxiv.org/abs/1707.02968`.

[123] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. "Residual Attention Network for Image Classification". In: 1 (2017). URL: `http://arxiv.org/abs/1704.06904`.

[124] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. "The 2017 DAVIS Challenge on Video Object Segmentation". In: (2017), pp. 1–4. URL: `http://arxiv.org/abs/1704.00675`.

[125] James Thewlis, Hakan Bilen, and Andrea Vedaldi. "Unsupervised learning of object frames by dense equivariant image labelling". In: *Proceedings of the Neural Information Processing Systems Conference (NIPS)* (2017).

[126] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. "AlignedReID: Surpassing Human-Level Performance in Person Re-Identification". In: (2017). URL: `http://arxiv.org/abs/1711.08184`.

[127] Yuanpu Xie, Fuyong Xing, Xiaoshuang Shi, Xiangfei Kong, Hai Su, and Lin Yang. "Efficient and robust cell detection: A structured regression approach". In: *Medical Image Analysis* (2017).

[128] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation". In: (). URL: `http://arxiv.org/abs/1606.02147`.