Describing Obstetric Ultrasound Video Content Using Deep Learning



Yuan Gao

Wolfson College

University of Oxford

Supervised by

Professor Alison Noble

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2019

Abstract

In recent years, advances in ultrasound technology have made devices cheaper and portable thus making the technology more accessible in both High Income Country (HIC) and Low and Middle Income Country (LMIC) settings. Meanwhile, there is an increasing amount of ultrasound scans that might not necessarily be performed by experienced sonographers. Automatic recognition of patterns in such scans can be difficult for traditional machine learning models trained with hand-crafted features, due to its high variability in terms of image quality and anatomies appearance. This doctoral thesis presents deep learning based methods for the automation of fetal structure recognition in free-hand obstetric ultrasound video.

First, we demonstrate the feasibility of training deep convolutional neural networks (CNNs) for ultrasound image classification. It is worth noting that the challenge faced in this case is overfitting caused by limited training data. We show that the over-fitting of deep CNNs can be prevented by: (i) tailoring the architecture, for example by removing fully-connected layers; (ii) introducing data augmentation during training; and (iii) careful regularization. We also visualize the high level CNN features to understand the classification results which suggests that standard CNN architectures are not good enough for learning discriminative representations of complicated anatomy, for example the fetal heart, that shows high variability in terms of anatomical appearance and scale.

Next, we address the challenge of fetal heart recognition by learning deep representations of ultrasound video that take into account temporal information. We inflate the standard CNN by adding a motion detection stream to the spatial stream. This novel two-stream CNN model demonstrates: (i) its capability of detection and localization of the fetal heart; (ii) significantly superior fetal heart recognition than standard CNNs; and (iii) the capability of describing fetal cardiac motion.

Finally, we expand the capability of object detection to other important fetal structures of interest, such as the fetal head and abdomen. We present a hybrid model, consisting of CNNs and recurrent neural networks (RNNs), which can localize the target structures in short video sequences. In this model, we do not have object level annotation (e.g. bounding boxes) so the localization is achieved by class activation mapping (CAM). Additionally, a soft-attention mechanism is introduced into the representation learning to produce a spatial-temporal saliency

map that is shown to be useful to highlight the object of interest, suggesting the potential as a video navigation cue.

The methods described in this thesis contribute to the ultrasound video image analysis literature, and also understanding of how to design image analysis algorithms for potential use by minimally trained users of ultrasound devices in HIC and LMIC settings.

Acknowledgements

First of all, I would like to sincerely acknowledge my supervisor, Prof. Alison Noble, who is my life mentor continuously providing me with support and advice throughout the PHD. I could not have done what I achieved in this thesis without her constant guidance.

I would like to thank my colleagues who make the Lab a inspiring and supportive environment. It was such an enjoyable and memorable four years in my life.

I would also like to show my gratitude to my funding body, Chinese Scholarship Council, who financially support my study and living in Oxford.

Last but not least, I am grateful to my family, especially my beloved parents who give me their selfless love and support me pursuing the degree spiritually in general.

Contents

List of Figures **xi**

List of Abbreviations **xix**

List of Notations **xxiii**

I Introduction **1**

I.1 Motivation . . . 1
I.2 Thesis Outline . . . 4
I.3 Publications . . . 5

II Literature Review **7**

II.1 Introduction . . . 8
II.2 Ultrasound Imaging in Obstetrics . . . 9
II.2.1 Standard Biometry Measurement . . . 10
II.2.2 Fetal Presentation . . . 12
II.2.3 Ultrasound in Limited-Resources . . . 13
II.3 Convolutional Neural Networks (CNNs) . . . 14
II.3.1 Convolution . . . 15
II.3.2 Sparse Connectivity . . . 17
II.3.3 Parameter Sharing . . . 17
II.3.4 Non-linearity . . . 18
II.3.5 Pooling . . . 21
II.4 Training Deep CNNs . . . 23
II.4.1 Cost Function . . . 24
II.4.2 Back-Propagation . . . 25
II.4.3 Optimization . . . 27
II.4.4 Parameter Initialization . . . 29
II.4.5 Batch Normalization . . . 32
II.4.6 Dropout . . . 35
II.5 Advances in Deep Learning . . . 37
II.5.1 Large-scale Image Classification . . . 37
II.5.2 Video Recognition (Learning Spatio-temporal Features) . . . 40

II.6 Obstetric Ultrasound Analysis . . . 41
II.6.1 Shallow Learning Approaches . . . 42
II.6.2 Deep Learning Approaches . . . 44

III Classification of Fetal Ultrasound Images with CNNs 47
III.1 Introduction . . . 48
III.2 Dataset . . . 49
III.2.1 Pre-processing . . . 50
III.2.2 Annotation . . . 50
III.3 Learning Deep Ultrasound Representations from Scratch . . . 54
III.4 Transfer Learning and Domain Adaptation . . . 58
III.5 Visualization: Deep Inside CNN . . . 61
III.5.1 Deconvolution . . . 61
III.5.2 Class-Specific Saliency Visualization . . . 63
III.5.3 t-Stochastic Neighbour Embedding (t-SNE) . . . 65
III.6 Experiments and Results . . . 68
III.6.1 Implementation Details . . . 69
III.6.2 Quantitative Evaluation . . . 72
III.6.3 Visualization . . . 81
III.7 Discussion and Conclusion . . . 86

IV Weakly-Supervised Detection, Localization and Characterization of Fetal Heartbeat 89
IV.1 Introduction . . . 90
IV.2 Dataset . . . 91
IV.2.1 Sequence Annotation . . . 92
IV.2.2 Preprocessing . . . 95
IV.3 Learning to Extract Motion from Videos in CNN . . . 95
IV.3.1 CNN Architecture . . . 95
IV.3.2 Multi-scale Learning . . . 99
IV.4 Motion-Appearance Fusion: Two-Stream CNN . . . 100
IV.5 Experiments and Results . . . 102
IV.5.1 Implementation Details . . . 103
IV.5.2 Classification Results . . . 104
IV.5.3 Localization Results . . . 106
IV.5.4 Characterization of Heartbeats . . . 111
IV.6 Discussion and Conclusion . . . 112

V Learning Deep Spatial-Temporal Features for Detection of Fetal Structures in Free-Hand Ultrasound Videos **117**
V.1 Introduction . . . 118
V.2 Dataset . . . 120
V.2.1 Preprocessing . . . 121
V.2.2 Annotation . . . 122
V.3 Delving into CNN for Learning Ultrasound Video Representations . . . 123
V.3.1 Convolutional LSTM . . . 123
V.3.2 Attention-gated ConvLSTM . . . 126
V.3.3 Recurrent Convolutional Networks . . . 127
V.4 Learning Weakly Supervised Classifiers for Localization . . . 128
V.4.1 Cross-Entropy Loss with Doubly Stochastic Regularization . . . 131
V.4.2 Adam Optimizer . . . 132
V.5 Experiments and Results . . . 134
V.5.1 Implementation Details . . . 134
V.5.2 Evaluation of Classification Performance . . . 135
V.5.3 Visualization of Attention . . . 144
V.5.4 Evaluation of Localization Performance . . . 146
V.5.5 Evaluation of running time . . . 150
V.6 Discussion and Conclusion . . . 150

VI Summary and Future Work **155**
VI.1 Contributions and Limitations . . . 156
VI.2 Directions for future work . . . 159
VI.2.1 Portable ultrasound for point-of-care . . . 159
VI.2.2 Learning and Describing Actions . . . 161
VI.2.3 Automation with Reinforcement Learning . . . 162

References **165**

List of Figures

I.1 **Fetal Anomaly Scans.** Green Box: Standard fetal planes. Red Box: Non-standard fetal Planes. UV: Umbilical Vein; SB: Stomach Bubble; SP: Spine; LV: Left Ventricle; RV: Right Ventricle; LA: Left Atrium; RA: Right Atrium. 2

I.2 Why deep learning is powerful? Deep neural networks are end-to-end trainable which learn features directly from data eliminating the need of feature engineering; and the features are robust and generalisable to different tasks. 3

II.1 **World Health Organization (WHO) map of global neonatal mortality 2009.** 9

II.2 **Example fetal abdomen structures at various gestational ages (17, 28 and 38 weeks).** The first column contain highly occluded examples, the middle column contains examples where key structures are partially occluded, and the last column contains high quality images. Adapted from [21] 10

II.3 **Standard fetal biometry measurements.** (a) CRL should be measured the longest straight line from the top of the fetal head to the rump. (b) The correct plane for measuring BPD, HC, and OFD. (c) The correct plane for measuring of the APAD, TAD and AC. (c) The correct plane for measuring the femur length. **CRL**: Crown-rump Length. **BPD**: Biparietal Diameter. **HC**: Head Circumference. **OFD**: Occipito-Frontal Diameter. **APAD**: Anterio-Posterior Abdominal Diameter. **TAD**: Transverse Abdominal Diameter. **AC**: Abdominal Circumference. 11

II.4 **Fetus in cephalic presentation (left) and breech presentation (right).** The **red dotted arrow** indicates the ultrasound sweep path. The presentation of the fetus in the womb is referred to as the relative position. **Cephalic presentation** is the most common case and refers to a head-down position. **Breech presentation** is the opposite of cephalic and indicates a head-up position. Adopted from [21] 13

II.5 Multilayer perceptrons (MLP) vs Convolutional Neural Network (CNN). Red tensor: input; Blue tensor: linear activation; Green tensor: non-linear activation; Yellow tensor: Pooled activation. . . . 15
II.6 Cross-correlation vs Convolution. 15
II.7 **Dense Connectivity** (Fully-Connected) vs **Sparse Connectivity**. Solid line: directly connected. Dash line: indirectly connected. Green lines: input to hidden transition; Red lines: hidden to hidden transition. 16
II.8 **Separate Parameters** vs **Sharing Parameters**.Solid lines: weights. Different colours represent different weights. Dash window: receptive filed of convolution with zeros padding for the boundary elements. Input size: $N_1 \times N_2$, Receptive field Size: 3×3. 18
II.9 **Sigmoid** vs. **Tanh**. **Left**: Sigmoid non-linearity squashes real numbers to range between [0,1]. **Right**: The tanh non-linearity squashes real numbers to range between [-1,1]. 19
II.10 **Rectified Linear Units (ReLU)** vs. **Parametric ReLU** [35]. For **PReLU**, the coefficient of the negative part is not constant and is adaptively learned. 20
II.11 Max pooling introduces **translation invariance**. **Left** shows the output of max pooling, with stride of two pixels and a square pooling region (blue box) width of two pixels. **Right** shows the input has been shifted to the right by one pixel. Every value in the input has changed, but the output stays the same, because the max pooling units are only sensitive to the maximum value in the neighborhood, not its exact location (red box). 22
II.12 Training a simple CNN with **backprop**. Blocks shown in purple: forward propagation. Blocks shown in green: back propagation. . . 27
II.13 Stochastic gradient descent with momentum (adopted from [34]. **Contour lines**: example of a quadratic loss function. 30

III.1 **A simple free-hand scan protocol.** A straight ultrasound video sweep is obtained from the cervix up to the top of the maternal abdomen. Three sample frames are shown from various points in the ultrasound video sweep illustrating "Other Structures"(left), "Fetal Skull" (centre) and "Fetal Abdomen" (right). 50
III.2 **Dataset Preprocessing.** (a) Original frame (b) Frame is masked to keep the data at the ultrasound's "field of view" only. This is mainly to remove the text that appears on the top left and right hand sides of the image (c)Frame is downsampled to half it's original size for computational efficiency (d) Output frame 51

III.3 **Scan Fetal Brain with various probe's position and angle.** The appearance of anatomical structures depends on the position and angle of the ultrasound probe. Here three different planes of the fetal brain are illustrated, where different anatomies are visible on each plane. **(a) Transventricular plane; (b) Transthalamic plane; (c) Transcerebellar plane.** 51

III.4 Annotation criteria for fetal skull. The only necessary criterion is presence of a semi-elliptical shape with or without the other standard key landmarks. The criteria are set loosely and independent of the presence of key anatomical structures. 53

III.5 Annotation criteria for fetal abdomen. The only necessary criterion is presence of at least one of the key landmarks i.e. stomach bubble and umbilical vein. 53

III.6 **Test accuracy against the number of trainable parameters.** **(a)** Deeper models tend to perform better than shallow models. Fully convolutional models tend to perform better than fully connected models. (adopted from [98]);**(b)** States-of-the-art deep CNNs trained on classification of fetal ultrasound images in this chapter. Deeper architecture tends to perform worse than shallow one. 56

III.7 A schematic of the CNN used in this paper. 96-256-384-384-256 are the number of feature maps generated in each convolutional layer. 55-27-13-13-13 are the sizes of the feature maps. There are 4096 neurons in each of the fully connected (FC) layers (2 FC layers). There are 4 neurons in the output layer i.e. the last layer. 57

III.8 Dictionary bases learned from: (Left) Natural images, (Right) US images. (Adopted from [6]) 59

III.9 Overview of transfer learning framework. **Top row**: The TCNN-A is pre-trained on the ImageNet database for classification. The labelled purple rectangles (e.g. W_{c1}) represent the weights learned by that layer. **Bottom row**: In the TCNN-B, the first l weight layers (in this example, l = 3) are copied from the base network A. The remaining layers are randomly initialized, and then the entire network is trained on ultrasound datasets with the transferred parameters locked (frozen) or unlocked (fine-tuned). 60

III.10 An illustration of the unpooling operation, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the CNN. The black/white bars are negative/positive activations within the feature map. (Adopted from [58]) 62

III.11 Feature visualization by deconvolution that reconstructs a high-level feature map in the pixel space. 63
III.12 Effect of data augmentation and dropout. **(a)** Training Objective (Loss), **(b)** Validation Objective. 70
III.13 Effect of data augmentation and dropout. **(a)** Training Error, **(b)** Validation Error. 71
III.14 class-specific Error. **standard**: without any regularization (corresponding red curves in Fig.III.13); **dropout**: 50% dropout to fully-connected layers (corresponding brown curves in Fig.III.13); **dropout**: 50% dropout to fully-connected layers plus data augmentation (corresponding green curves in Fig.III.13); 73
III.15 Effect of reducing network size and batch normalization (BN). BN was applied to every single convolution layer and FC layer. **(a)** Training Objective (Loss), **(b)** Validation Objective. 74
III.16 Effect of reducing network size and batch normalization (BN). BN was applied to every single convolution layer and FC layer. **(a)** Training Error, **(b)** Validation Error. 75
III.17 Error break down into classes. 77
III.18 Convolutional Kernels learned by the first layer: (a) without any regularization (b) dropout+ data augmentation (c) small network (half the number of trainable parameters) + strong regularization (data augmentation, dropout, batch normalization. 78
III.19 Classification performance of transfer learning models. 80
III.20 Misclassified abdomen and heart images. Abdomen samples misclassified as heart (enclosed by cyan box); Heart samples misclassified as abdomen (enclosed by yellow box). 82
III.21 t-SNE visualisation. (a) Raw training data. (b) Raw testing data. (c) Extracted features of the 7th fully connected layer from training dataset. (d) Extracted features of the 7th fully connected layer from validation dataset. Blue: Skull, Cyan: Abdomen, Yellow: Heart, Red: Other. 83
III.22 Raw validation images embedding. 83
III.23 **Class-specific Saliency Maps.** Four columns from left to right. In each column: original images (Left); corresponding saliency maps on which hot spots highlight the class-specific pixels (Right). **S: Skull**, **A: Abdomen**, **H: Heart**, **O: Other**. 85
III.24 Visualization of mid-high level features: showing the toppest activation of feature maps in the 5th layer (layer before fully-connection) across the validation data, projected down to pixel space using deconvolution approach. 87

IV.1 Localization confidence maps for fetal heart detection of our weakly-supervised two-stream ConvNet. Note our model is robust to variations in fetal heart appearance, scale, position and image contrast. 91

IV.2 **Acquisition of fetal cardiac planes (adapted from [104]).** The five axial views for optimal fetal heart screening are illustrated: **(a) Four-chamber** view of the fetal heart, showing the right and left ventricles (RV, LV) and atria (RA, LA), foramen ovale (FO) and pulmonary veins (PV) to the right and left of the descending aorta (dAo). **(b) Left ventricular outflow tract view**, showing the aortic root (Ao), LV, RV, LA, and RA and a cross-section of the dAo. **(c) Right ventricular outflow tract** view showing the main pulmonary artery (MPA) and the bifurcation into the right (RPA) and left (LPA) pulmonary arteries and cross-sections of the ascending aorta (Ao) and dAo aorta. **(d) Three-vessel view.** This view best demonstrates the relationship between the pulmonary artery (PA), aorta (Ao) and superior vena cava (SVC) in the upper mediastinum. The pulmonary artery, to the left, is the largest of the three and the most anterior, whereas the SVC is the smallest and most posterior. **(e)Three vessels and trachea view** showing the superior vena cava (SVC), pulmonary artery (PA), ductus arteriosus (DA), transverse aortic arch (from proximal Ao to dAo) and trachea (Tr). IVC, inferior vena cava; **L, Abdominal left; R, Abdominal right.** 92

IV.3 Annotation criteria for fetal heart. (a) Four chamber view. (b) Left ventricular Outlow Tract (LVOT). (c) Three Vessel View. (d) Non-standard View. The only necessary criterion is presence of a heart with or without the other standard key landmarks. The criteria are set loosely and independent of the presence of key anatomical structures. 93

IV.4 Intraclass Statistics e.g. Heart. FCV: Four Chamber View, 3VV: Three Vessel View, LVOT: Left Ventricular Outflow Tract, N-S: Non-Standard Views. 94

IV.5 CNN motion detector. The CNN takes raw pixels of video frames as input, and outputs a dense optical flow at quarter its resolution. The network consists of 2 convolutional layers, 2 max-pooling layer, and 2 pixelwise weights (1×1 convolutions). 96

IV.6 Conv1 filter patterns. 96

IV.7 Conv2 filter patterns. 98

IV.8 Multi-scale learning for motion detection. **Concat.** denotes stacks the feature maps at the same spatial locations. 99
IV.9 Network Architecture. The numbers in each layer (e.g. 64 and 3×3 in Conv1 refer to the number of kernels and their size, /1 denotes a stride of 1,and the size of max-pooling window is equivalent to its stride (e.g./2 denotes window size 2×2, pooling with a stride of 2). **concat.** denotes stacks the feature maps at the same spatial locations. **add.** denotes the sum of the feature maps at the same spatial locations. 101
IV.10 Coarse to Fine Augmentation. The network enclosed by red box applied on multiple down-sampled versions of input frames. 102
IV.11 Intermediate layer's activation. It shows the progressive evolution of activation of features when feeding-forward through the motion and appearance network respectively, and finally after convolutional fusion i.e. Conv5 a class activation map is obtained. 107
IV.12 Localization saliency (Spatial CNN vs. Two stream CNN) of test heart examples. 1st Col: **Raw input**. 2nd Col: **Single stream CNN**. 3rd Col: **Two-stream CNN (as proposed)**. First row shows the standard planes (i.e. three vessel, four chamber, and LVOT), the rest of examples are all non-standard planes. 109
IV.13 Characterization of fetal heart motion (optical flow is color coded). 111
IV.14 Threshold and zoom into the flow map. 112
IV.15 Localization saliency of fetal hearts at test time. Red arrow points the cardiac valve which gradually disappears. 113
IV.16 Indexed flow maps for characterizing the fetal cardiac motion. . . . 114
IV.17 Failure example of localizing the fetal heart. 115

V.1 Class-specific features learned at different scales. Row-wise (top-down): skull, abdomen, heart and other (none of structures of interest). Column-wise (left-right): last frame in a sequence, Conv1, Conv2, Conv3, Conv4, Conv5 i.e. indices of convolution layers . . . 119
V.2 Doubly Stochastic Attention. Examples of 10 consecutive frames on the left-hand side each column and their respective attention dynamics on ride-hand side. The attention maps show a progressive change of focus spatially over the course of generation, and localisation are progressively corrected. Best viewed in Adobe Reader. All videos should play automatically. 120
V.3 Data and Annotation. 121

V.4 Steps for Preprocessing. Frame Generation: Frames are masked to keep the *field of view* only by removing the texts on the top and scale marks near the right boundary of frame. Sequence Generation: Sampling consecutive frames with Tϵ {3, 5, 10} frames in our experiments. 122

V.5 Annotation Tool. Annotation tool used to annotate the bounding boxes of the structures of interest. Annotation is done on the last frames of test video clips. 123

V.6 (a) Inside the convolutional LSTM. The input-to-hidden and hidden-to-hidden transitions are done locally with convolution operations. (b) Convolutional LSTMs are applied to a local spatial neighbourhood repeatedly over successive frames. 124

V.7 Inside the attention cell. 126

V.8 Network Overview. CNN feature maps are fed to the Attention Module to generate an attention map at each time step, then the attention-warped (element-wise product) feature maps are fed to the Recurrent Module to update the hidden states. Inside the Attention Gated LSTM: the input-to-hidden and hidden-to-hidden transitions are done locally with convolution operations. 128

V.9 Weakly Supervised Classification Layers. (a) **Classifier A.**: An adaptation layer (1×1 convolution) is applied to map the hidden representation (at last time step) to Class Activation Maps, which are then fed into a Global Average Pooling (GAP) or Global Max Pooling (GMP) layer to obtain a classification score vector. (b) **Classifier B.**: The hidden representation (at the last time step) is first fed into a GAP or GMP layer to obtain a global feature embedding (vector) then a dense layer is used to map the global feature vector to classification scores. 129

V.10 Skull Class Activation Maps (CAMs). Skull examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization. 136

V.11 Abdomen Class Activation Maps (CAMs). Abdomen examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization. 137

V.12 Heart Class Activation Maps (CAMs). Heart examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization. 138
V.13 Other Class Activation Maps (CAMs). Other examples (background frames) of the high level CAMs generated from the four base-line models (see Section III.A.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization. 139
V.14 Multi-scale CAMs for Fine-grained Detection. Fetal abdominal examples of CAMs learnt at different feature scale. Column-wise (left-right): Last frames in the abdominal video-clips, and CAMs (from low to high feature levels) at Conv1, Conv2, Conv3, Conv4, Conv5, respectively. Red boxes: umbilical vein, Blue boxes: stomach bubble. 141
V.15 Fetal Heart Detection. Fetal heart examples of CAMs generated from different models. Row-wise (top-down): examples of heart frames, **two-stream ConvNet** [83] and **AD-GMP**, respectively. 142
V.16 Examples of CAMs generated by AD-GAP and attention gated AD-GAP. Row-wise (top-down): example frames, **AD-GAP** and **AG-AD-GAP**, respectively. 144
V.17 Spatial-temporal Attention. **Row 1**: examples videos. **Row 2 to 4**: attention maps at **Conv3 to Conv5** levels, respectively. Best viewed in Adobe Reader. All videos should play automatically. . . . 145
V.18 Effect of temporal size. **Row 1 to 3**: Examples of skull, abdomen, and heart videos, respectively. **Column 1, 3 and 5**: attention learnt with a temporal scale of **3, 5 and 10** frames, respectively. **Column 2, 4 and 6** are CAMs generated at the last frame of the videos. Best viewed in Adobe Reader. All videos should play automatically. . . . 146
V.19 Examples of predicted bounding boxes of target anatomy. **Row 1 to 3**: **Skull**, **Abdomen**, and **Heart** examples, respectively. **Red and blue boxes** are generated from **Conv4 and Conv5 CAMs** with different thresholds. **White boxes** are the predicted boxes that achieve the best **Intersection of Union (IOU)** against the **ground truth boxes (green boxes)**. 147
V.20 Categorical Mean Intersection Over Union(IOU). 148
V.21 The average and standard deviation of critical parameters 149

VI.1 Portable ultrasound probe compatible with smart phones. 160
VI.2 Ultrasound probe movement. 161
VI.3 Reinforcement learning setting in Obstetric Ultrasound. 164

List of Abbreviations

1D, 2D, 3D . . One- or two-dimensional, referring in this thesis to spatial dimensions in an image.

US . . . . . Ultrasound.

BoVW . . . . . Bag of Visual Words.

SIFT . . . . . Scale Invariant Feature Transform.

SVM . . . . . Support Vector Machine.

FV . . . . . Fisher Vector.

ISUOG . . . . International Society of Ultrasound in Obstetrics and Gynecology Education Committee.

CSP . . . . . Cavum Septi Pellucidi.

CP . . . . . Choroid Plexus.

PHV . . . . . Posterior Horn of Ventricle.

AH . . . . . Anterior Horn.

C . . . . . Cerebellum.

V . . . . . Vermis.

SB . . . . . Stomach Bubble.

SP . . . . . Spine.

UV . . . . . Umbilical Vein.

AC . . . . . Abdominal Circumference.

DA . . . . . Descending Aorta.

CHD . . . . . Congenital Heart Disease.

RV, LV . . . . Right and Left Ventricles.

RA, LA . . . . Right and Left Atria.

FO . . . . . Foramen Ovale.

PV . . . . . Pulmonary Veins.

AO . . . . . Aortic Root.

dAO . . . . . Descending Aorta.

MPA . . . . . Main Pulmonary Artery.

LPA . . . . . Left Pulmonary Artery.

RPA . . . . . Right Pulmonary Artery.

LVOT . . . . . Left ventricular outflow Tract.

RVOT . . . . . Right ventricular outflow Tract.

FCV . . . . . Four Chamber View.

3VV . . . . . Three Vessel View.

N-S . . . . . Non-standard View.

SVC . . . . . superior vena cava.

Tr . . . . . Trachea.

IVC . . . . . Inferior Vena Cava.

CRL . . . . . Crown-rump Length.

BPD . . . . . Biparietal Diameter.

HC . . . . . Head Circumference.

OFD . . . . . Occipito-Frontal Diameter.

APAD . . . . . Anterio-Posterior Abdominal Diameter.

TAD . . . . . Transverse Abdominal Diameter.

AC . . . . . Abdominal Circumference.

CNN, ConvNet Convolutional Neural Networks.

TCNN . . . . . Transfer Learning CNN.

DCNN . . . . Direct Learning CNN.

FCN . . . . . Fully Convolutional Network.

ReLU . . . . . Rectified linear units.

PReLU . . . . Parametric ReLU.

MP . . . . . Max Pooling.

GMP . . . . . Global Max Pooling.

GAP . . . . . Global Average Pooling.

FC . . . . . Fully Connected.

Backprop . . . Back Propagation.

SGD . . . . . Stochastic gradient descent.

BN . . . . . Batch Normalization.

t-SNE . . . . . t-Stochastic Neighbour Embedding.

KLD . . . . . Kullback-Leibler divergences.

RNN . . . . . Recurrent Neural Network.

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

CAM . . . . . Class Activation Mapping.

IOU . . . . . Intersection Over Union.

Adam . . . . . Adaptive Moment Estimation.

AD-GAP . . . Adaptation with Global Average Pooling.

AD-GMP . . . Adaptation with Global Max Pooling.

AG-AD-GAP Attention-Gated Adaptation with Global Average Pooling.

AG-AD-GMP Attention-Gated Adaptation with Global Max Pooling.

CAM-GAP . . Class Activation Mapping with Global Average Pooling.

AG-CAM-GAP Attention-Gated Class Activation Mapping with Global Average Pooling.

CAM-GMP . Class Activation Mapping with Global Max Pooling.

AG-CAM-GMP Attention-Gated Class Activation Mapping with Global Max Pooling.

List of Notations

I	Two-dimensional Image.
K	Convolutional Kernel.
F	Feature Map.
W	Trainable Weights (Parameters).
$N_1 \times N_2$	Spatial Size of Feature Maps.
C	Total Number of Categories (Classes).
J	Loss Function.
X	CNN Feature Maps (ChapterV).
A	Attention Map (ChapterV).
N_c	Number of Feature Channels.
P	Probability Distributions.
G	Gradient.
N	Number of Training Samples.
H	A Mini-batch of linear activations (ChapterII) or Hidden States (ChapterV).
SM	Saliency Maps.
CB	Class-balanced Weighting Factor (ChapterV).
CE	Cross-entropy Loss.
M	Number of Samples in a Batch.
B	A Mini-Batch.
N_m	Number of Sub-models.
S	Score Maps.
O	Number of Feature Channels (ChapterV).
L	Number of Layers.
i, j	Spatial Index or General index.

l . . . . . Layer Index.

g . . . . . Non-linear Activation Function.

z . . . . . Output of A Linear Transformation.

x . . . . . High Dimensional Datapoints (t-SNE) or Neural Network Input.

. . . . . y Low Dimensional Datapoints (t-SNE).

b . . . . . Bias.

s . . . . . Slope (Leaky ReLU).

k . . . . . Class (Categorical) Index.

bm . . . . . binary mask.

c . . . . . Feature Channel Index.

f . . . . . General Functions.

μ,σ . . . . . Mean, Standard Deviation.

γ,β . . . . . Scale, Shift Factor (Batch Normalization).

σ . . . . . Sigmoid Function.

ϵ . . . . . Learning Rate.

α . . . . . Momentum Rate.

δ . . . . . Small Positive Value (10^{-8}).

r_1,r_2 . . . . . 1st and 2nd Moment (Adam).

ρ_1,ρ_2 . . . . . Momentum Rates (Adam).

λ . . . . . Regularization Factor (ChapterV).

γ . . . . . Positive Integer (ChapterV).

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain...

— Cicero's *de Finibus Bonorum et Malorum*

I

Introduction

Contents

I.1 Motivation 1
I.2 Thesis Outline 4
I.3 Publications 5

I.1 Motivation

Medical ultrasound is the most widely used clinical imaging modality worldwide but is very different from its medical imaging counterparts of X-ray, CT (X-ray Computed Tomography), MRI and nuclear medicine imaging (PET, SPECT) which provide anatomical and functional image that are relatively easy to discern by (the radiologist's) eye. Ultrasound (US) stands out on its own in that it is an interactive imaging technology where the user (sonographer) interrogates a volume of interest with a hand-held probe and interpretation of the images is normally done "on-the-fly" with a record of key findings. When compared to the other medical imaging modalities, ultrasound is significantly economical and benefits from point-of-care for real-time imaging and visualization [1]. More importantly,

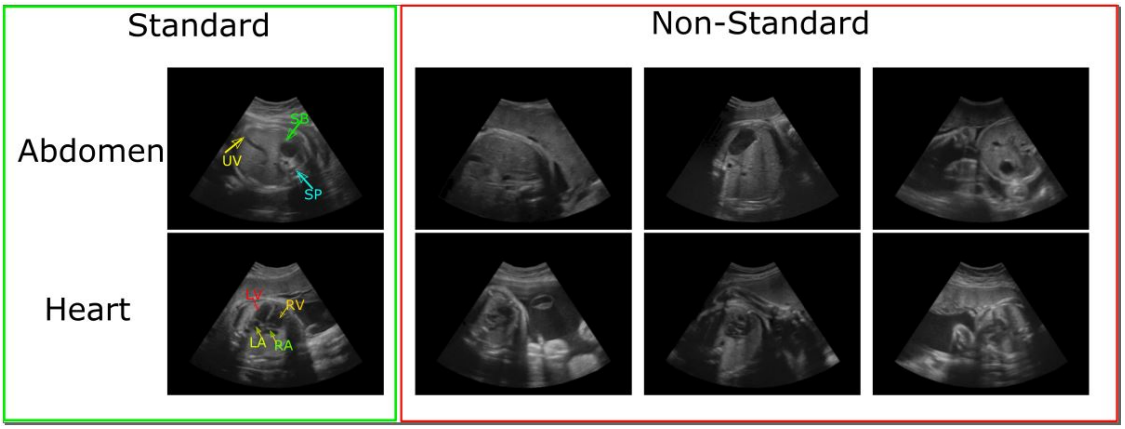


Figure I.1: Fetal Anomaly Scans. Green Box: Standard fetal planes. Red Box: Non-standard fetal Planes. UV: Umbilical Vein; SB: Stomach Bubble; SP: Spine; LV: Left Ventricle; RV: Right Ventricle; LA: Left Atrium; RA: Right Atrium.

an acceptable safety profile makes the ultrasound widely used in current prenatal care, where it accounts for roughly 40% of ultrasound's total use in medicine [2]. Its applications in prenatal care ranges from estimating of gestational age to identifying pregnancy related complications and diagnosis of fetal abnormalities [3]. However, despite its benefits, obstetric ultrasound requires a high level of skill to navigate an ultrasonic volume or sequence and interpret the sonographical patterns for decision-making, which can be highly challenging for inexperienced sonographers. There is a demand of developing capabilities that go beyond the sonographers eye to simplify US image interpretation and ultrasonic pattern recognition in order to make the technology more accessible to non-experts.

In order to detect fetal abnormalities and growth restriction, the fetal anomaly scan is recommended as an essential routine of obstetric ultrasound screening [4]. During an anomaly scan, a number of 2D images, enclosed by green box in Fig.I.1, are acquired by a skilled sonographer following standardized clinical protocols [5], which requires the presence of certain anatomical structures. The standard 2D planes are examined to identify any structural abnormalities and are also used for biometry measurement, for example, the standard abdomen plane is used to measure the fetal abdominal circumference. However, for a non-expert, it is very unlikely to acquire the standard planes with an acceptable quality, instead, there is

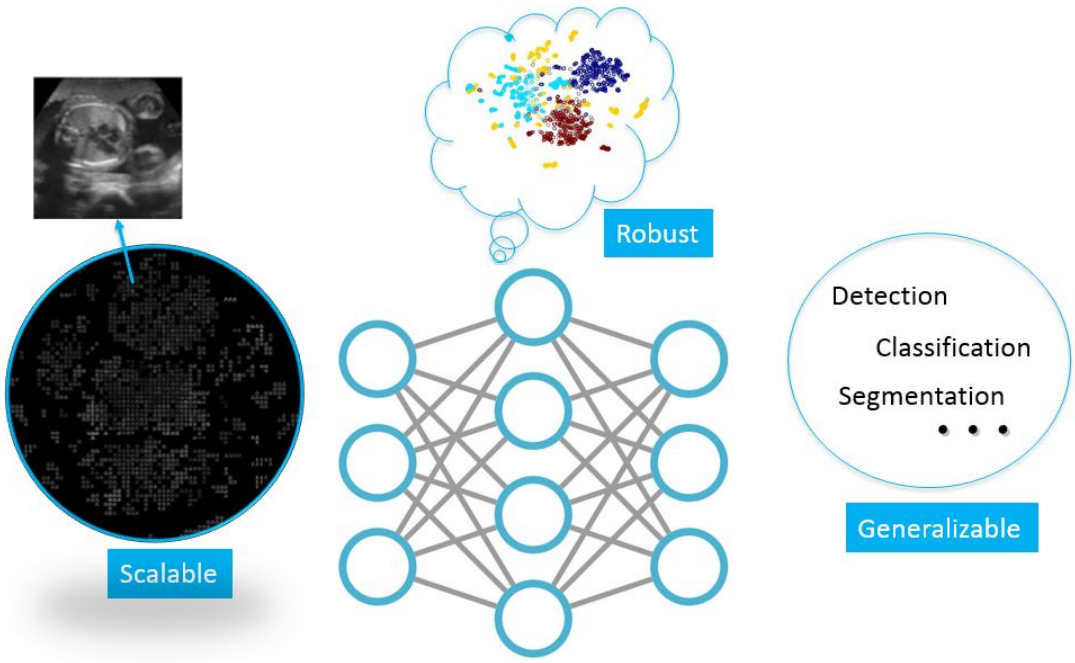


Figure I.2: Why deep learning is powerful? Deep neural networks are end-to-end trainable which learn features directly from data eliminating the need of feature engineering; and the features are robust and generalisable to different tasks.

significantly more non-standard planes than standard ones are acquired during the scan, illustrated in Fig.I.1. As can be seen, the key anatomical structures, such as fetal heart and stomach bubble, are highly varied in terms of appearance, which can be very challenging to interpret for a novice sonographer or non-expert. Most stat-of-the-art studies, such as [6] and [7], only considers the recognition of the standard planes, while the non-standard planes are usually discarded. However, the non-standard planes can be of great value to study the transition between the standard and non-standard sequences. It is therefore a need of developing smart machines to interpret the non-standard plane sequences, which could not only provide useful information guiding non-experts to interpret the scan contents, but also help the skilled sonographers to improve the efficiency of fetal screening. Moreover, this could potentially facilitate the use of ultrasound for poin-of-care, which is of great importance for the developing worlds where medical resources are very limited.

This thesis aims to automate content description of 2D fetal US video sequences by developing deep learning based methods for detecting structures and patterns

of structural changes over scan duration, that are not necessarily reliably found by sonographer's eye. Compare to traditional machine learning approaches, deep learning has many advantages as illustrated in Fig.I.2: (1) deep learning models, such as Convolutional Neural Networks (CNNs) are scalable, which particularly fits well on a large amount of training dataset; (2) deep CNNs learn hierarchical feature representations end-to-end which eliminates the need of feature engineering; (3) deep CNNs are able to learn semantically meaningful feature presentations that are robust to variations in the data; (4) deep learning models are generalizable to multitasks which only needs to fine-tune a base model with different objective function i.e. loss function; 5) deep learning is distributable which makes it possible to deploy AI functionality to mobile ultrasound devices for point-of-care. However, despite the merits, it is still of big challenge to apply deep learning in medical image analysis. One of the most important reasons is it is usually a laborious and expensive process to collect and label the medical image dataset. Therefore, in this thesis, we also demonstrate the feasibility of applying deep learning in ultrasound image analysis with limited amount of data and weak annotation.

I.2 Thesis Outline

Chapter II describes the clinical application of ultrasound imaging in fetal health screening and discusses its current limitations and challenges. This chapter also provides a review of basics and advances in deep learning, and discusses current state-of-the-art methods that are related to our work.

Chapter III explores how the stat-of-the-art CNN architecture, such as AlexNet [8], should be tailored to fit the fetal ultrasound image classification. This chapter also presents a simple transfer learning design that is employed to investigate the generality of CNN features. Furthermore, this chapter presents state-of-the-art approaches to visualize the deep CNNs learned for fetal image classification.

Chapter IV investigates localization and characterization of fetal heartbeat via weakly-supervised classification. This chapter presents a two-stream CNN model that demonstrates the importance of incorporating motion into representation learning for detecting a fetal heart sequence.

Chapter V demonstrates the superiority of exploring temporal context information for fine-grained detection. This chapter presents a spatial-temporal model for detection of general fetal anatomical structures, which looks into the patterns learned at different feature level. Furthermore, the effect of incorporating soft-attention into spatial-temporal representation learning has also been studied.

Chapter VI concludes the thesis contributions and discusses its limitations. Future direction of research will also be discussed.

I.3 Publications

This thesis contributes to the pioneer research of deep learning in ultrasonic pattern recognition towards computer guided fetal screening. The works have been published (or submit to) the following conferences and journal:

Chapter III **Y Gao**, MA Maraci, JA Noble. "Describing ultrasound video content using deep convolutional neural networks". Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI), 2016.

Chapter IV **Y Gao** and JA Noble. "Detection and Characterization of the Fetal Heartbeat in Free-hand Ultrasound Sweeps with Weakly-supervised Two streams Convolutional Networks". International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI), 2017.

Chapter V Y Gao and JA Noble. "Learning and Understanding Deep Spatio-Temporal Representations from Free-Hand Fetal Ultrasound Sweeps". In-

ternational Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI), 2019.

Y Gao and JA Noble. "Learning Deep Spatial-Temporal Features for Detection of Fetal Structures in Free-Hand Ultrasound Videos". IEEE Transactions on Medical Imaging (TMI), 2019. (Under Review)

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [9]

II

Literature Review

Contents

II.1 Introduction **8**
II.2 Ultrasound Imaging in Obstetrics **9**
II.2.1 Standard Biometry Measurement 10
II.2.2 Fetal Presentation 12
II.2.3 Ultrasound in Limited-Resources 13
II.3 Convolutional Neural Networks (CNNs) **14**
II.3.1 Convolution 15
II.3.2 Sparse Connectivity 17
II.3.3 Parameter Sharing 17
II.3.4 Non-linearity 18
II.3.5 Pooling 21
II.4 Training Deep CNNs **23**
II.4.1 Cost Function 24
II.4.2 Back-Propagation 25
II.4.3 Optimization 27
II.4.4 Parameter Initialization 29
II.4.5 Batch Normalization 32
II.4.6 Dropout 35
II.5 Advances in Deep Learning **37**
II.5.1 Large-scale Image Classification 37
II.5.2 Video Recognition (Learning Spatio-temporal Features) 40
II.6 Obstetric Ultrasound Analysis **41**
II.6.1 Shallow Learning Approaches 42
II.6.2 Deep Learning Approaches 44

This chapter presents a review of literature relevant to this thesis. It starts with introducing the research motivation and background in Section II.1. Section II.2 provides a brief introduction to ultrasound in obstetrics and discusses its challenges and limitations. Section II.5 describes the advances in deep learning that inspire the novelty in this thesis. Finally, Section II.6 will discuss the traditional approaches (Section II.6.1), referred to as shallow learning, and also current trends i.e. deep learning approaches (Section II.6.2) for obstetric ultrasound analysis.

II.1 Introduction

Ultrasound is a non-radioactive and real time imaging modality that is generally preferred in routine examination of fetal health and growth measurements. Regular ultrasound examination is of great importance to identify potential pregnancy complications and facilitate pregnancy management. However, recent studies [10], [11], [12], [13], [14] and [15] have shown that maternal mortality is unacceptably high. It was estimated that in 2015, roughly 303,000 women died during and following pregnancy and childbirth. Almost all of these deaths occurred in low-resource (resource-limited) settings that for examples, in rural and remote regions (particularly in developing countries), there are limited diagnostic imaging facilities [16] and lack of sonography education and ultrasound-trained physicians [17]. Newborn health are closely linked to maternal health. It was estimated that globally 2.5 million children died in the first month of life in 2017. There are approximately 7,000 newborn deaths every day, amounting to 47% of all child deaths under the age of 5-years, up from 40% in 1990 [14]. Almost all (99%) of such incidents occur in developing countries, where not all pregnancies are cared effectively and complications of the simplest kind during pregnancy may result in fatal outcomes [11]. Figure II.1 shows the map of neonatal mortalities around the

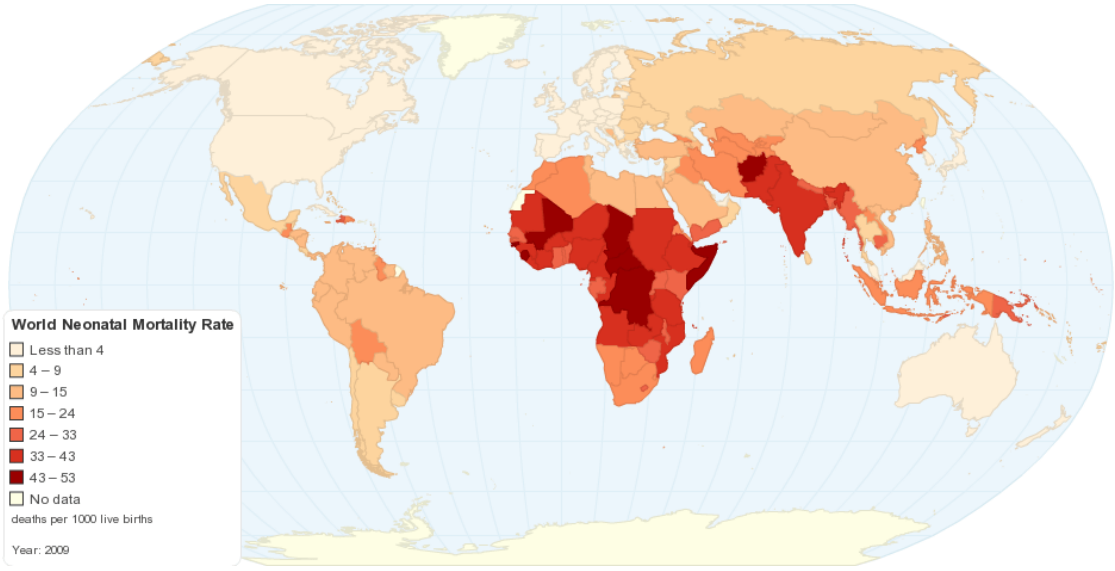


Figure II.1: World Health Organization (WHO) map of global neonatal mortality 2009.

world in 2009, where it should be noted the developing countries (darker colours) suffering from higher mortality rates.

One of the main causes of the poor care is fetal ultrasound screening requires highly experienced sonographers. This makes it difficult for a wider adoption of clinical ultrasound for pregnancy care, in the developing world, where there is a substantial lack of experienced sonographers [18], [19] leading to delays in diagnosis and treatment. Even in developed countries, such as UK, [20] reported a sonographer vacancy rate of 18.1% in 2014, across 59 departments, and it also suggested the main reasons for vacancies was that departments are not able to recruit qualified applicants. The shortage of skilled sonographers worldwide make it necessary to develop automated analytic tools that could potentially help in training inexperienced operators, standardization and quality control in obstetric scans.

II.2 Ultrasound Imaging in Obstetrics

A pregnancy usually lasts for about 40 weeks, which are usually divided into three trimesters. Each trimester lasts for 12 to 13 weeks. Figure II.2 illustrates fetal

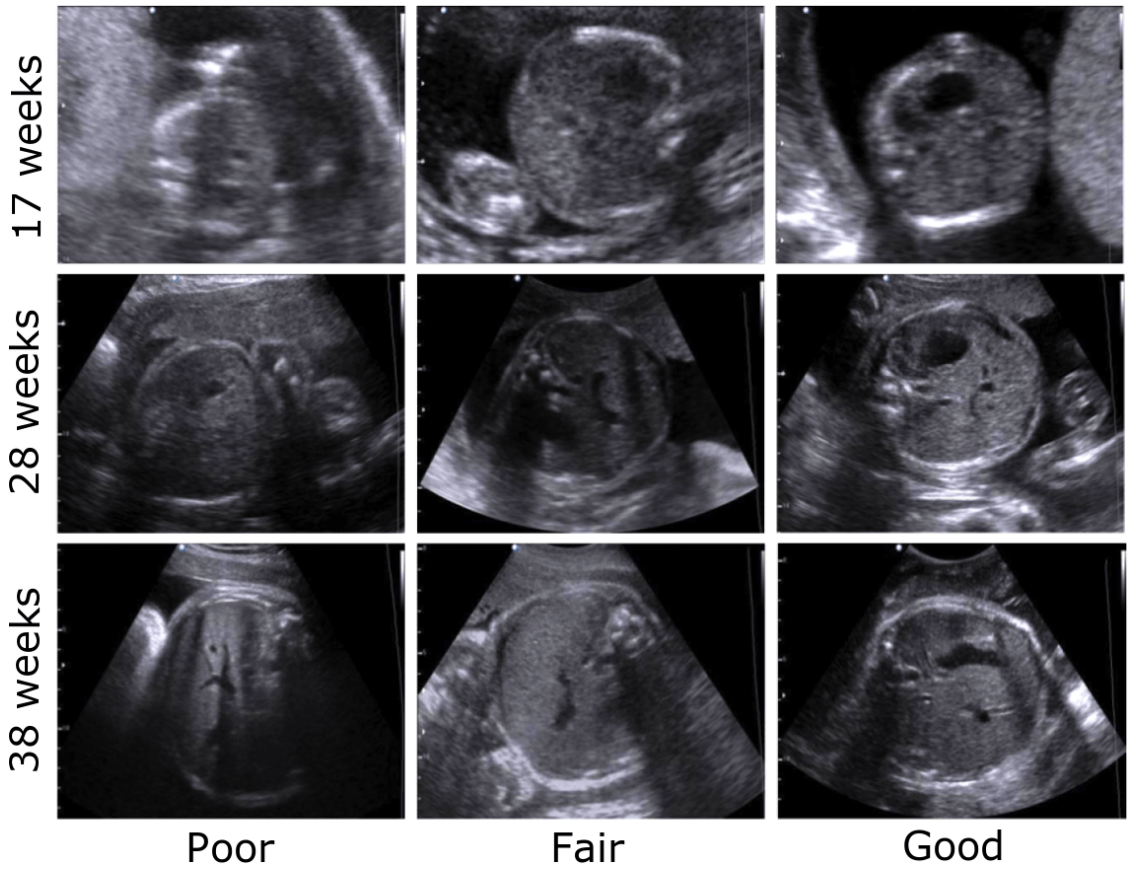


Figure II.2: Example fetal abdomen structures at various gestational ages (17, 28 and 38 weeks). The first column contain highly occluded examples, the middle column contains examples where key structures are partially occluded, and the last column contains high quality images. Adapted from [21]

abdominal images of poor, fair and good image qualities at gestational age of 17 weeks, 28 weeks, and 38weeks respectively. It should be noted that although the anatomical structures, such as stomach bubble and umbilical vein, are better established and benefit from better acoustic contrast in later gestational ages, the possibility of occluded structures and artefacts also increases with advancing pregnancy. In this thesis, the third trimester scans i.e. 28 weeks+ has been studied that the dataset used will be introduced in the following chapters.

II.2.1 Standard Biometry Measurement

Fetal biometry is the measurements of the fetus and various segments of the fetal anatomy in which fetal head, abdomen, and femur measurements are the

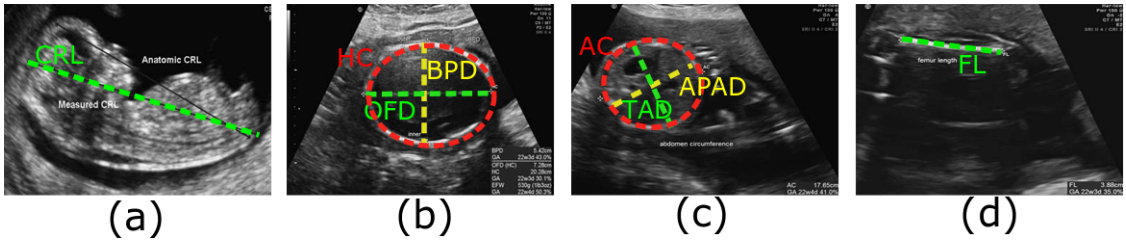


Figure II.3: Standard fetal biometry measurements. (a) CRL should be measured the longest straight line from the top of the fetal head to the rump. (b) The correct plane for measuring BPD, HC, and OFD. (c) The correct plane for measuring of the APAD, TAD and AC. (c) The correct plane for measuring the femur length. **CRL**: Crown-rump Length. **BPD**: Biparietal Diameter. **HC**: Head Circumference. **OFD**: Occipito-Frontal Diameter. **APAD**: Anterio-Posterior Abdominal Diameter. **TAD**: Transverse Abdominal Diameter. **AC**: Abdominal Circumference.

most commonly used. These biometric measurements can be used to estimate gestational age and fetal weight, evaluate interval fetal growth, and identify fetuses who are either growth restricted or macrosomic. These measurements may influence antepartum and intrapartum management and may be used to predict peripartum outcomes [22]. Biometry is therefore of great value and importance for obstetrical practice.

A pregnancy is divided into trimesters: the first trimester is from week 1 to the end of week 12; the second trimester is form week 13 to the end of week 26; and the third trimester is from week 27 onwards to the end of the pregnancy. The INTERGROWTH-21st Project produced an international standard for early fetal size and ultrasound dating of pregnancy based on Crown-Rump Length (CRL) measurement [23]. During the first trimester, the fetus is typically very small and the CRL measurement is carried out from 9 to 13 weeks of gestation. CRL, as illustrated in Fig.II.3(a), not truly measure the anatomic CRL but measures the longest straight line measurement from the top of the fetal head to the rump [24]. Standard ultrasound fetal biometry measurements are typically taken at the second and third trimester from the fetal head, abdomen and femur. According to the international fetal growth standards [25], the five primary ultrasound measures of fetal growth, i.e. Head Circumference (HC), Biparietal Diameter (BPD), Occipito-Frontal Diameter (OFD), Abdominal Circumference (AC) and Femur Length (FL),

are obtained from 14 weeks to 42 weeks of gestation. Regarding the fetal head, as illustrated in Fig.II.3(b), the Biparietal Diameter (BPD) is measured from the outer border of the parietal bones at the widest part of the skull; the Occipito-Frontal Diameter (OFD) is measured across the long axis of the skull from the outer border of the occipital and frontal edges; the Head Circumference (HC) is measured at the outer perimeter of the skull not including the fetal skin. Similarly, standard measurements taken from the fetal abdomen, as illustrated in Fig.II.3(c), include the Anterio-Posterior Abdominal Diameter (APAD) measured from the outer borders of the body outline covering the spine to the anterior abdominal wall, Transverse Abdominal Diameter (TAD) measured from the outer borders of the body outline across the abdomen at the widest point perpendicular to the APAD and Abdominal Circumference (AC). As illustrated in Fig.II.3 (d), femur length (FL) is measured perpendicular to the axis of the bone, not including the epiphyses or ossification centers of the bone to metaphysis. The distal femoral epiphysis is typically not seen before 28 weeks gestation [26].

II.2.2 Fetal Presentation

Knowing the fetal presentation prior to delivery is of great importance of effective management and planning of labour. Fetal presentation can be briefly categorized into cephalic and breech as illustrated in Fig.II.4. Breech presentation occurs rarely (only accounting for 1-3% of all deliveries [27] is defined as a fetus in a longitudinal presentation with the buttocks or feet closest to the cervix. Previous studies [28], [29], [27] and [30] have shown the breech presentation is associated with an increased risk of pregnancy complications such as difficult delivery, growth restriction and even death. In addition, a breech presentation is very likely to resulting in a preterm delivery. The percentage of breech deliveries decreases with advancing gestational age from 22% of births prior to 28 weeks gestation to 7% of births at 32 weeks gestation and 1-3% of births at term [30]. Therefore, detection of the order of fetal

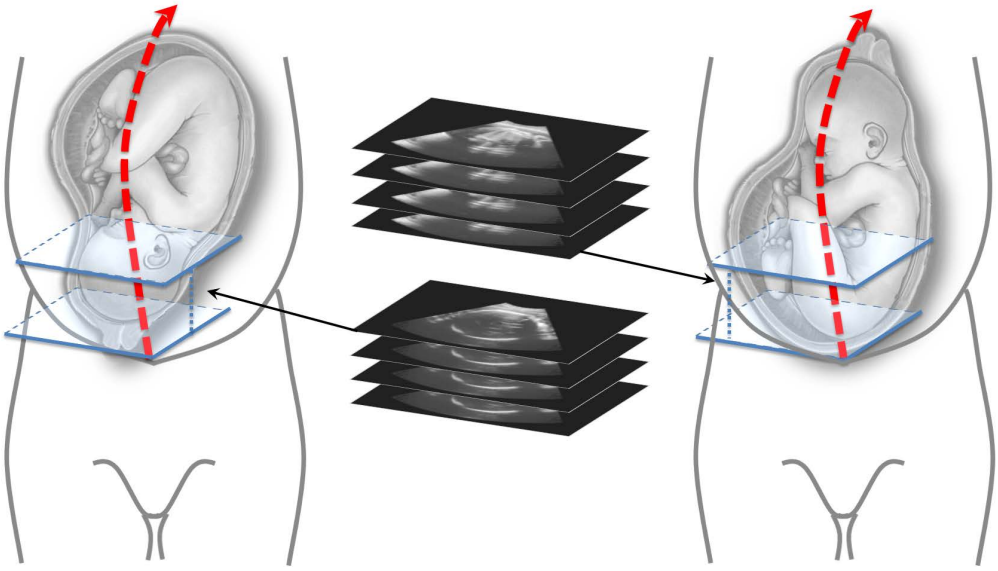


Figure II.4: Fetus in cephalic presentation (left) and breech presentation (right). The **red dotted arrow** indicates the ultrasound sweep path. The presentation of the fetus in the womb is referred to as the relative position. **Cephalic presentation** is the most common case and refers to a head-down position. **Breech presentation** is the opposite of cephalic and indicates a head-up position. Adopted from [21]

structures (for instance, fetal head is the last structure popping up in a scan with breech presentation) can be of great importance to reducing such adverse outcomes.

II.2.3 Ultrasound in Limited-Resources

As introduced early this section, standard biometry measurements typically need to acquire a number of standardized scan planes, which have a constrained (similar) appearance and contain certain anatomical structures. For instance, abdominal circumference (AC) measurement, it is required to acquire a standard fetal abdomen plane on which the stomach bubble, umbilical vein and the spine are clearly visible [31]. Acquisition of the standard scans requires skilled sonographers who are specifically trained to be able to meet the standards, as training programmes have previously shown to improve the quality and consistency of measurements [32]. However, one of the challenges, mentioned earlier in this chapter, there is lack of qualified sonographers worldwide, especially in the developing areas. For wider adoption of clinical ultrasound in limited-resource settings, it is required to

define much simpler scanning protocols that can be performed by non-experts (e.g. healthcare workers) and novice sonographer as well as qualified sonographers in terms of measurements reproducibility [33]. In this thesis, using of the simplest protocol (will be introduced in next chapter) was proposed and was named a **predefined free-hand** acquisition protocol. The question is then what useful diagnostic information can be automatically acquired and analyzed from such data.

II.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a specialized kind of neural network for processing data that has a known, grid-like topology. For example, image data can be thought of as a 2D grid of pixels. A regular neural network i.e. multi-layer perceptrons, as illustrated in Fig.II.5 receives an input feature vector, and transforms and activates it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully-connected to all neurons in the previous layer, and where neurons in a single layer function are completely independent and do not share any connections. In contrast, a CNN receives 2D images as input, and employs convolution (a linear operation) in place of matrix multiplication to transform the input into a 3D tensor (feature representation) with neurons arranged in 3 dimensions: width, height, depth (referring to the feature dimensionality). Moreover, each neuron is only connected to a small region of the layer before it, which is referred to as sparse local connectivity, instead of all of the neurons in a fully-connected manner. The local connectivity is shared across different spatial locations. A typical layer of a CNN consists of three stages. In the first stage the layer performs several convolutions in parallel, across different spatial locations, to produce a set of linear activations. The second stage is sometimes called the detector stage; each linear activation is run through a nonlinear activation function, such as the most commonly used, the rectified linear activation unit (ReLU) [8]. The third stage applies a pooling function, normally a max pooling, to down-sample the activated feature maps.

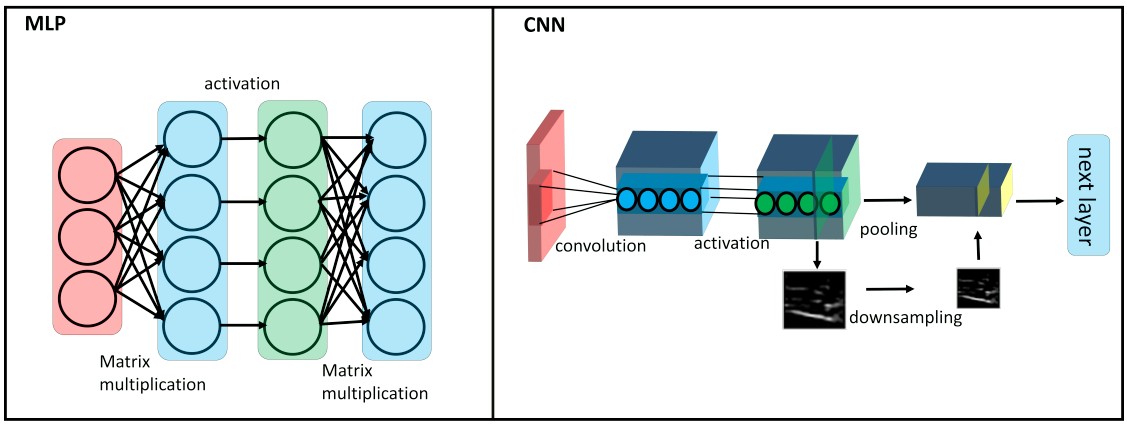


Figure II.5: Multilayer perceptrons (MLP) vs Convolutional Neural Network (CNN). Red tensor: input; Blue tensor: linear activation; Green tensor: non-linear activation; Yellow tensor: Pooled activation.

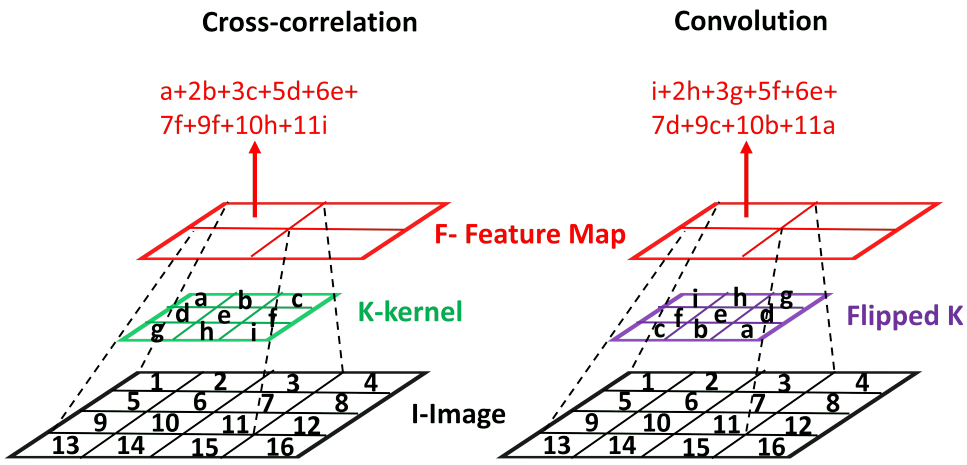


Figure II.6: Cross-correlation vs Convolution.

II.3.1 Convolution

In machine learning imaging applications, the input to a CNN is usually a multidimensional array of data, referred to as a tensor, for example, the input is a two-dimensional image **I** with spatial index (i,j), and **K** is also a two-dimensional tensor representing a small convolutional kernel of size 3×3 pixels. The discrete **convolution** is then defined as:

$$F(i,j) = (I * K)(i,j) = \sum_{m=-3}^{3} \sum_{n=-3}^{3} K(m,n) I(i-m, j-n) \qquad \text{(II.1)}$$

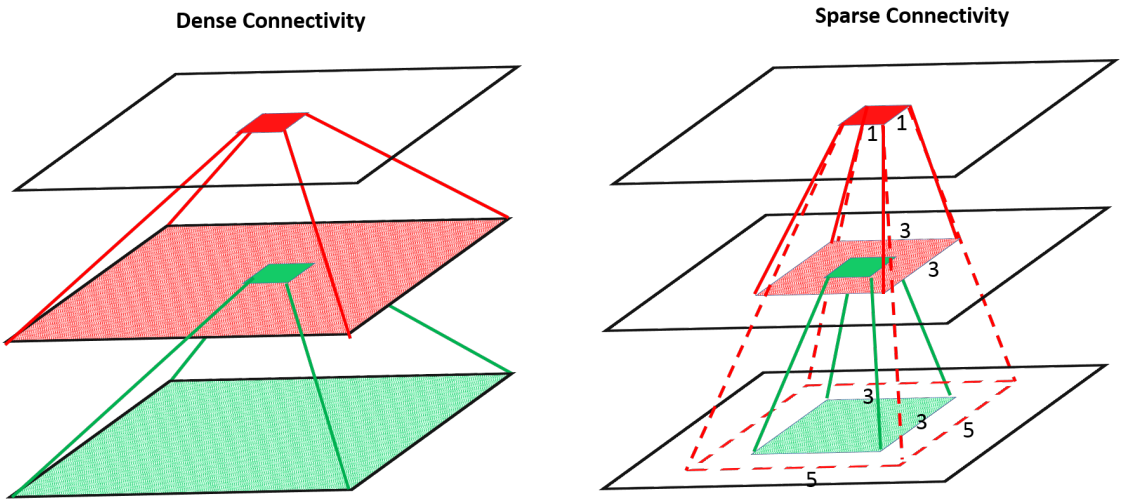


Figure II.7: Dense Connectivity (Fully-Connected) vs **Sparse Connectivity**. Solid line: directly connected. Dash line: indirectly connected. Green lines: input to hidden transition; Red lines: hidden to hidden transition.

It should be noted that the kernel is flipped relative to the input patch for the convolution operation, as illustrated in Figure II.6, in the sense that as m increases, the index into the input increases, but the index into the kernel decreases. Instead, in our work, the **cross-correlation** was implemented, which is the same as convolution but without flipping the kernel as:

$$F(i,j) = (I * K)(i,j) = \sum_{m=-3}^{3} \sum_{n=-3}^{3} K(m,n) I(i+m, j+n) \tag{II.2}$$

Theoretically, convolution is just a linear transformation on a signal with a filter, whereas correlation is a measure of similarity between two signals. When implementing CNNs in the thesis, cross-correlation is employed because it does template matching hierarchically that each individual filter serves as a template. Many machine learning libraries, like Tensorflow, Keras implement cross-correlation but some others like Theanos implement convolution. It does not make any difference when comes to implement a CNN because filters are initialised with Gaussian kernels which are symmetrical.

II.3.2 Sparse Connectivity

Traditional neural network layers use matrix multiplication by a matrix of parameters with separate parameters describing the interaction between each input unit and each output unit, which means every output unit is densely connected with every input unit, as depicted in Figure II.7 left. However, CNNs typically have sparse connectivity by making the kernel several orders of magnitude smaller than the input. For instance, when processing an image, the input image might have thousands or millions of pixels, but the low level kernels can detect meaningful features such as edges that occupy only tens or hundreds of pixels. By the sparse connectivity, there are fewer parameters to store. This both reduces the memory requirements of the model and improves its statistical efficiency. It also significantly improves the computational efficiency as fewer computational operations are required. In a deep convolutional network, units in the deeper layers may indirectly interact with a larger area of the input. As shown in Figure II.7 right, each neuron in a given layer has a receptive field of 3×3 pixels of the previous layer, the neuron labelled in red indirectly interacts with the neurons in the input layer with a larger receptive filed of 5×5. This allows the network to efficiently describe complicated interactions between many variables by constructing such interactions from simple building blocks that each describe only sparse interactions.

II.3.3 Parameter Sharing

In a traditional neural net, each output neuron has a unique set of weights, as illustrated in Fig.II.8 left. Each element of the weight matrix is used exactly once when computing the output of a layer. In a CNN, the same kernel is used at every position of the input, as illustrated in Figure II.8 right. The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, only one set needs to be learned. Suppose the input has N_1N_2 elements, it requires $N_1N_2 3^2$ parameters to map the input to the output

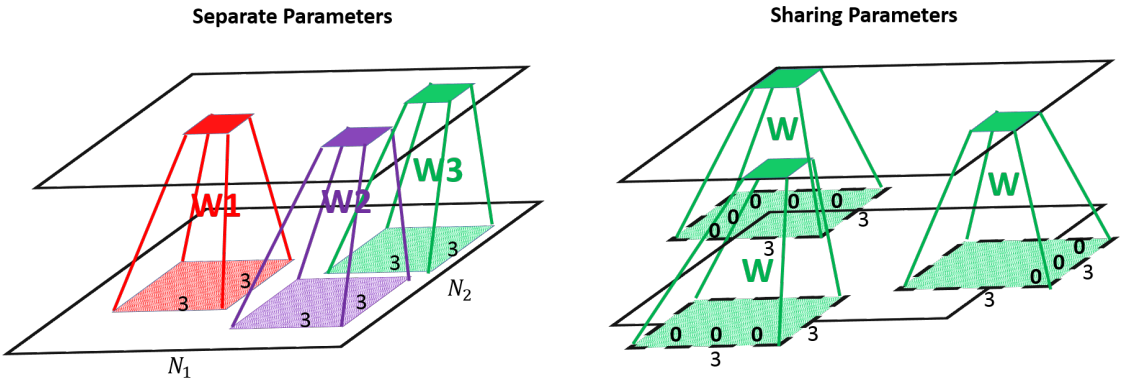


Figure II.8: Separate Parameters vs **Sharing Parameters**.Solid lines: weights. Different colours represent different weights. Dash window: receptive filed of convolution with zeros padding for the boundary elements. Input size: $N_1 \times N_2$, Receptive field Size: 3×3.

with same number of elements. However, a convolution with a shared kernel of size 3×3 only requires a small number of parameters i.e. 3^2 (3<<N_1 or N_2 in a CNN). Convolution is thus dramatically more efficient than dense matrix multiplication in terms of memory requirements and statistical efficiency.

II.3.4 Non-linearity

To make the neural network expressive, it is necessary to apply a non-linear activation function, denoted by **g**, right after the linear transformation, denoted by **z**. A possible choice of the non-linearity is the logistic sigmoid activation function;

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{II.3}$$

or the hyperbolic tangent activation function;

$$g(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{II.4}$$

It should be noted in Fig.II.9 on the left that the **sigmoid** function saturates across most of its domain: it is easily saturated to a high value 1 when **z** is very positive; saturate to a low value 0 when **z** is very negative; and is strongly sensitive to its input when z is near 0. This is a very undesirable property of the sigmoid for a neuron's activation saturate at either tail, Since the gradient at these regions is

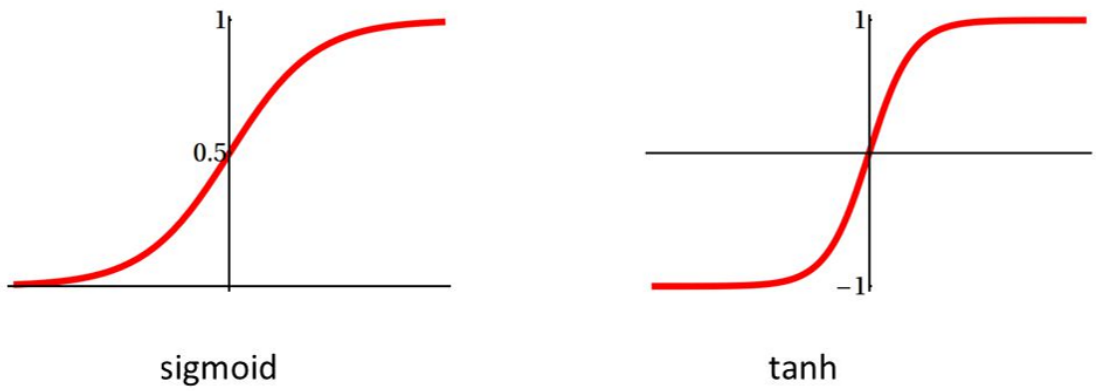


Figure II.9: Sigmoid vs. **Tanh**. **Left**: Sigmoid non-linearity squashes real numbers to range between [0,1]. **Right**: The tanh non-linearity squashes real numbers to range between [-1,1].

almost zero during backpropagation, and almost no error signal will flow through the neuron to its weights and recursively to its input. Initializing the weights of sigmoid neurons has to be done with care to prevent saturation. For example, if the initial weights are too large then most neurons would become saturated at an early stage of training and the network will then barely learn. Additionally, sigmoid outputs are not zero-centered, which is also an undesirable property since neurons in later layers would be receiving input that is not zero-centered. This could cause undesirable zig-zagging dynamics in the gradient updates for the weights, because the input is always positive i.e. ($\mathbf{x}>\mathbf{0}$ element-wise in $\mathbf{z}=\mathbf{W}^T\mathbf{x}+\mathbf{b}$). In this case, the gradient on the weights $\mathbf{W}$ will become either all positive, or all negative, during backpropagation, depending on the gradient on the $\mathbf{z}$. As an alternative, the **tanh** non-linearity is shown in Figure II.9 on the right. Like the sigmoid neuron, its activation saturates, but unlike the sigmoid neuron its output is zero-centered. In practice, this makes the tanh preferrable to the sigmoid. It resembles the identity function more closely, in the sense that $\mathbf{tanh}(0)=0$ while $\sigma(0) = 0.5$. Because **tanh** is similar to the identity function near 0, training a deep neural network resembles training a linear model as long as the activations of the network can be kept small. Due to the saturation issue, **sigmoid** and **tanh** are discouraged in deep feed-forward network design. Instead, they are more commonly used in settings, such as recurrent networks [34], requiring the ability to rule out the use of piecewise

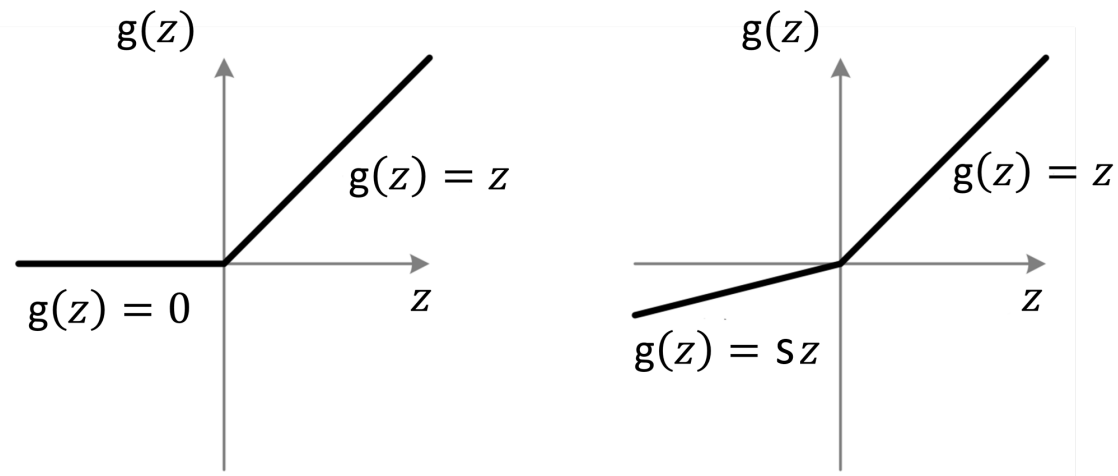


Figure II.10: Rectified Linear Units (ReLU) vs. **Parametric ReLU** [35]. For **PReLU**, the coefficient of the negative part is not constant and is adaptively learned.

linear activation functions. A **Rectified linear units (ReLU)** is a popular choice for deep CNNs. A **ReLU** is defined as the activation function:

$$g(z) = max\{0, z\} \tag{II.5}$$

ReLUs are easy to optimize because they are so similar to linear units. The only difference between a linear function and a ReLU is that a ReLU outputs zero across half its domain, as shown in Fig.II.10 on the left. This make the derivatives through a rectified linear unit remain large and consistent whenever the unit is active. Compared to sigmoid/tanh neurons that involve expensive operations i.e. exponential, the **ReLU** can be simply implemented by thresholding activations at zero. However, one drawbacks of the **ReLU** is that it is unable to learn via gradient-based methods on examples for which their activation is zero.

Several generalizations of rectified linear units exist that perform comparably to rectified linear units and occasionally better. Some of these generalizations, as shown in Fig.II.10 on the right are based on using a non-zero slope **s** when **z<0**:

$$g(z, \alpha) = max(0, z) + smin(0, z) \tag{II.6}$$

Absolute value rectification fixes s=-1. This has been proposed for object recognition from natural images [36], where it makes sense to seek features that

are invariant under a polarity reversal of the input illumination. A **leaky ReLU** [37] fixes **s** to a small value like 0.01 while a **Parametric ReLU(PReLU)** [35] treats **s** as a learnable parameter. **Maxout units** [38] further generalize rectified linear units by dividing **z** into groups of n values, then applies a max-pooling across the groups to output the maximum element;

$$g(z) = \max_{j\epsilon[1,n]} z_j \tag{II.7}$$

A **maxout** unit is able to learn the activation function itself rather than just fixed relationship between units. It can learn a piecewise linear convex function, and with a large n, it can learn to approximate any convex function with arbitrary fidelity. In particular, a **maxout** layer with two pieces i.e. n=2, can learn to approximate the same function of the input z as a traditional layer using the rectified linear activation function, absolute value rectification function, or the leaky or parametric ReLU, or can learn to implement a totally different function altogether. However, unlike the ReLU neurons, **maxout** doubles the number of parameters for every single neuron, leading to a high total number of parameters, so it typically needs more regularization than a rectified linear unit. They can work well without regularization if the training set is large and the number of pieces per unit is kept low [39]. In this thesis, ReLU was chosen to use as the non-linearity considering the risk of overfitting using **maxout** units.

II.3.5 Pooling

A pooling function replaces the output of the detector stage at a certain location with a summary statistics of the nearby outputs. For example, the max pooling operation [40] spots the maximum output within a rectangular neighborhood. Other popular pooling functions include the average of a rectangular neighbourhood, the L^2 norm of a rectangular neighbourhood, or a weighted average based on the distance from the central pixel [34]. Only the max and average pooling was used in this thesis.

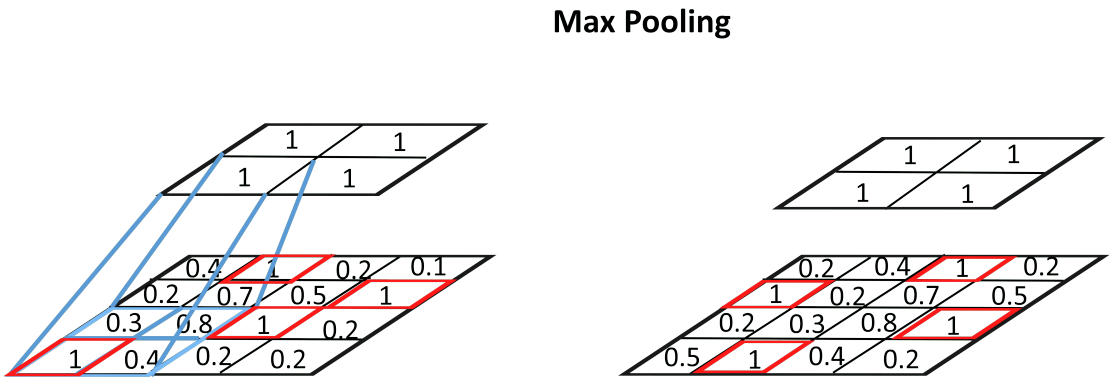


Figure II.11: Max pooling introduces **translation invariance**. **Left** shows the output of max pooling, with stride of two pixels and a square pooling region (blue box) width of two pixels. **Right** shows the input has been shifted to the right by one pixel. Every value in the input has changed, but the output stays the same, because the max pooling units are only sensitive to the maximum value in the neighborhood, not its exact location (red box).

In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if the input is translated by a small amount, the values of the pooled outputs do not change, as illustrated in Figure II.11. Because pooling summarizes the responses over a whole neighbourhood, it is possible to use fewer pooling units than detector units, by reporting summary statistics for pooling regions spaced **n** pixels apart rather than 1 pixel apart (in Figure II.11, **n=2**). This improves the computational efficiency of the network because the next layer has roughly **n** times fewer inputs to process. When the number of parameters in the next layer is a function of its inputs size, such as when the next layer is fully connected and based on matrix multiplication, this reduction in the input size can also result in improved statistical efficient and reduced memory requirements for storing the parameters.

Pooling is essential to build a fully convolutional network for handling varying spatial size of inputs. For example, if classifying images of variable size is wanted, the problem is that the input to the classification layer must have a fixed size. This is managed in practice by varying the size of an offset between pooling regions so that the classification layer always receives the same number of summary statistics regardless of the input size. For example, in some applications, before feeding into

the classification layer, a global pooling layer may be used to output a feature vector summarizing the global statistics, regardless of the image size. Actually, in this thesis, a pooling operation was very often used to generate a global feature descriptor for classification tasks.

II.4 Training Deep CNNs

Training a neural network is very different from training other machine learning models, such as SVMs. Neural networks are usually trained by using iterative, gradient-based optimizers that drive the cost function to a very low value, rather than the linear equation solvers used to train linear regression or the convex optimization with global convergence guarantees used to train logistic regression or SVMs. There is no guarantee that a deep neural network will converge at global minima because the non-linearity causes most interesting loss functions to become non-convex. With non-convex functions, nearly any deep model is essentially guaranteed to have an extremely large number of local minima. In the context of deep learning, it is generally acceptable to converge at a local minima as long as it corresponds to significantly low values of the cost functions. Additionally, stochastic gradient descent applied to a non-convex loss function is sensitive to the values of the initial parameters. It is important to initialize all weights to small random values. In the following subsections, the iterative gradient-based optimization algorithms used to train deep CNNs will be discussed in detail, with parameter initialization. Moreover, regularization has been found of great importance in training deep CNNs. In the context of deep learning, most regularization strategies are based on regularizing estimators. Regularization of an estimator works by trading increased bias for reduced variance. It will be discussed in Section III.6 that the regularization of an estimator is very often dealt. An effective regularization reduces variance significantly while not overly increasing the bias, which is to take a model from the overfitting regime to match the true data generating process. But in practice, it almost never has access to the true data generating process, which means controlling

the complexity of the model is not a simple matter of finding the model of the right depth, with the right number of parameters. In stead, and indeed in practical deep learning scenarios, it almost always does find the best fitting model, in the sense of minimizing generalization error, is a large model that has been regularized appropriately. In this section, regularization methods will be discussed in detail, such as dropout that is employed in training deep CNNs.

II.4.1 Cost Function

To apply gradient-based learning, it is a must to choose an appropriate cost function. Cross-entropy is used between the training labels and the CNN's predictions as the cost function. CNNs define a distribution $\mathbf{P(y|x;W)}$ (x-input, W-trainable parameters), which are trained using maximum likelihood. This is simply to minimize the negative log-likelihood, described as the cross-entropy between the training data and the model distribution. The cost function is given by:

$$J(W) = -logP(y|x;W) \tag{II.8}$$

The negative log-likelihood helps to avoid the saturation of the objective (loss becomes very flat). Many output units involve an **exp** function that can saturate when its argument is very negative. The log function in the negative log-likelihood cost function undoes the **exp** of some output units.

The choice of how to represent the output determines the form of the cross-entropy function. In this thesis, a linear layer i.e. fully-connected layer is used as output to predict unnormalized log probabilities $\mathbf{z}$ (score of the k^{th} class):

$$z = W_o^T h + b_o \tag{II.9}$$

where W_o is the output layer's weights, b_o is the output layer's bias, h is the input to the output layer. The softmax function is then applied to exponentiate and normalize $\mathbf{z}$ to produce a probability distribution $\hat{y}$ over $\mathbf{C}$ categorical labels for classification ($\mathbf{C=4}$, in our tasks), with $\hat{y_k} = P(y = k|x)$ ($k\epsilon\{1, ..., C\}$). It is

required not only that each element of $\hat{y}_k$ be between 0 and 1, but also that the entire vector sums to 1 so that it represents a mathmatically valid probability distribution. Formally, the softmax function is given by:

$$P(y = k|x) = \frac{exp(z_k)}{\sum_{j=1}^{C} exp(z_j)} \tag{II.10}$$

The goal is to maximize logP(y=k|x). Defining the softmax in terms of exp is natural because the log in the log-likelihood can undo the exp of the softmax:

$$logP(y = k|x) = z_k - log \sum_{j=1}^{C} exp(z_j) \tag{II.11}$$

The first term of equation II.11 shows that the input z_k directly contributes to the cost function. Because this term cannot saturate, the learning can thus proceed even if the contribution of z_k to the second term of equation II.11 becomes very small. When maximizing the log-likelihood, the first term tends to enlarge the z_k, while the second term tends to reduce all values of $\mathbf{z}$. To gain some intuition, the second term of equation II.11 $log \sum_{j=1}^{C} exp(z_j)$ can be roughly approximated by $max_j(z_j)$ based on the idea that exp(z_j) is insignificant for any z_j that is noticeably less than $max_j(z_j)$. Intuitively, this approximation suggests the negative log-likelihood cost function always strongly penalizes the most active incorrect prediction. If the correct answer has the largest input to the softmax, then the $-z_k$ term and the $log \sum_{j=1}^{C} exp(z_j) \approx max_j(z_j) = z_k$ term will roughly cancel. This example will then contribute little to the overall training cost, which will be dominated by other examples that are not yet correctly classified.

II.4.2 Back-Propagation

It has already known that CNNs take the data x_0 (in our case, images) propagating through the hidden layers and generate a probability distribution $\hat{y}$ for classification. This process is called **forward propagation**. Let x_1,x_2,...,x_L be the outputs of each layer in the network (L layers), and let x_0=**x** denote the network input. Each intermediate output at layer l, x_l= $f_l(x_{l-1};W_l)$, is computed from the previous

output x_{l-1} by applying the function f_l with parameters w_l. During training, forward propagation continues onward until it produces a scalar cost $\mathbf{J(W)}$. The **back-propagation** algorithm [41] allows the information from the cost to then flow backwards through the network, in order to compute the gradient $\bigtriangledown_W J(W)$.

In an image-based CNN, each $x_l \epsilon \mathbb{R}^{N_1^l \times N_2^l \times N_c^l}$ is a 3D tensor where the first two dimensions N_1^l (height) and N_2^l (width) are spatial dimensions. The third dimension N_c^l is the number of feature channels. Therefore, the tensor x_l represents a $N_1^l \times N_2^l$ field of N_c^l-dimensional feature vectors, one for each spatial location. Each layer is a function y= f(x). The derivative of the function **f** contains the derivative of each output component $y_{i'j'c_i'}$ with respect to each input x_{ijc_i}, where **i**, **j** are spatial coordinates, and c_i denotes the index of the feature channel, for a total of $N_1' \times N_2' \times N_c' \times N_1 \times N_2 \times N_c$ elements naturally arranged in a 6D tensor. Instead of expressing derivatives as tensors, tensors are converted into vectors, with each element of a tensor arranged in lexicographical order: $vec(x) = \begin{bmatrix} x_{111} \\ x_{211} \\ \cdots \\ x_{N_1 11} \\ x_{121} \\ .. \\ x_{N_1 N_2 N_c} \end{bmatrix}$

By stacking both input and output, each layer **f** can be seen as vector function **vec f**, whose derivatives forms the conventional Jacobian matrix: $\frac{\mathrm{d}vec(f)}{\mathrm{d}(vec(x))^T} = \begin{bmatrix} \frac{\partial y_{111}}{\partial x_{111}} & \frac{\partial y_{111}}{\partial x_{211}} & \cdots & \frac{\partial y_{111}}{\partial x_{N111}} & \frac{\partial y_{111}}{\partial x_{121}} & \cdots & \frac{\partial y_{111}}{\partial x_{N1N2Nc}} \\ \frac{\partial y_{211}}{\partial x_{111}} & \frac{\partial y_{211}}{\partial x_{211}} & \cdots & \frac{\partial y_{211}}{\partial x_{N111}} & \frac{\partial y_{211}}{\partial x_{121}} & \cdots & \frac{\partial y_{211}}{\partial x_{N1N2Nc}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{N1'11}}{\partial x_{111}} & \frac{\partial y_{N1'11}}{\partial x_{211}} & \cdots & \frac{\partial y_{N1'11}}{\partial x_{N111}} & \frac{\partial y_{N1'11}}{\partial x_{121}} & \cdots & \frac{\partial y_{N1'11}}{\partial x_{N1N2Nc}} \\ \frac{\partial y_{111}}{\partial x_{111}} & \frac{\partial y_{121}}{\partial x_{211}} & \cdots & \frac{\partial y_{121}}{\partial x_{N111}} & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{N1'N2'Nc'}}{\partial x_{111}} & \frac{\partial y_{N1'N2'Nc'}}{\partial x_{211}} & \cdots & \frac{\partial y_{N1'N2'Nc'}}{\partial x_{N111}} & \frac{\partial y_{N1'N2'Nc'}}{\partial x_{121}} & \cdots & \frac{\partial y_{N1'N2'Nc'}}{\partial x_{N1N2Nc}} \end{bmatrix}$

In this chapter, simple CNNs are mainly dealt that have the single stream configuration trained with **backpropagation**, as illustrated in Fig.II.12. The goal of **backpropagation** is to compute the gradient of the loss value **J** (output) with respect to each network parameter W_l:

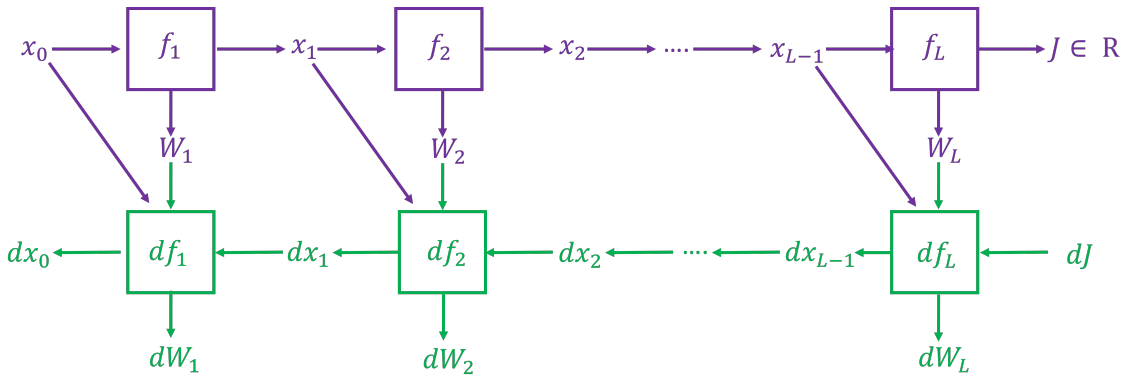


Figure II.12: Training a simple CNN with **backprop**. Blocks shown in purple: forward propagation. Blocks shown in green: back propagation.

$$\frac{\mathrm{d}J}{\mathrm{d}(vec(W_l))^T} = \frac{\mathrm{d}}{\mathrm{d}x(vec(W_l))^T}[f_L(\cdot; W_L) \circ ... \circ f_L(\cdot; W_2) \circ f_L(x_0; W_1)] \tag{II.12}$$

By applying the chain rule and by using the matrix notation introduced above, the derivative can be written as:

$$\frac{\mathrm{d}J}{\mathrm{d}(vec(W_l))^T} = \frac{\mathrm{d}vec(f_L(x_{L-1}; W_L))}{\mathrm{d}(vec(x_{L-1}))^T} \times ... \times \frac{\mathrm{d}vec(f_{l+1}(x_l; W_{l+1}))}{\mathrm{d}(vec(x_l))^T} \times \frac{\mathrm{d}vec(f_l(x_{l-1}; W_l))}{\mathrm{d}(vec(W_l))^T} \tag{II.13}$$

II.4.3 Optimization

As introduced above, the parameters W_l of an image classification CNN is estimated by maximum likelihood, when viewed in log space, decompose into a sum over each training example:

$$W_{ML} = \underset{W}{argmin} J(W) = \underset{W}{argmin} \sum_{i=1}^{N} -logP(y^{(i)}|x^{(i)}; W) \tag{II.14}$$

Optimization algorithms that use the entire training set are called batch methods, because they process all of the training examples simultaneously in a large batch. Computing the gradient $\bigtriangledown_W J(W)$ with a **batch** method is very expensive because it requires evaluating the model on every example in the entire dataset. In practice, a **minibatch** method is actually used to compute the gradient by randomly sampling a subset of examples from the training dataset, then taking the average over only those examples. One consideration in mini-batches is redundancy in the training

set, the worst scenario would be all M samples are identical to each other, but in practice, it is unlikely to have this situation, but larger numbers of examples make very similar contributions to the gradient. Another consideration is the memory limitation for typical hardware setups, it is almost impossible to train on the entire dataset with batch method. Small batches can also offer a regularizing effect [42]. The stochastic method is often used to train deep CNNs in this thesis, is **Stochastic Gradient Descent** [43], as illustrated in algorithm 1. A crucial hyper-parameter

Algorithm 1 Stochastic gradient descent (SGD)

Require: Learning rate ϵ_i
Require: Initial parameter W
1: **while** stopping criterion not met **do**
2: Sampling a minibatch of M examples from the training set $\{x^{(1)}, ..., x^{(N)}\}$ with corresponding targets $y^{(i)}$.
3: Computing gradient estimate: $\hat{G} \leftarrow +\frac{1}{N} \bigtriangledown_W \sum_i J(f(x^{(i)}; W), y^{(i)})$
4: Apply update: $W \leftarrow W - \epsilon\hat{G}$
5: **end while**
6: **return**

for SGD is the learning rate denoted by ϵ_i. In practice, it is necessary to gradually decrease the learning rate over time. This is because the SGD gradient estimator introduces a source of noise (the random sampling of m training examples) that does not vanish even when approaching and reaching a minimum. It is common to decay the learning rate linearly until iteration τ:

$$\epsilon_i = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \tag{II.15}$$

with $\alpha = \frac{i}{\tau}$. It is crucial to choose an appropriate initial learning rate ϵ_0. In practice, if it is too large, the learning curve will show violent oscillations, with the cost function often increasing significantly. If it is too low, learning proceeds slowly, and may get stuck with a high cost value [34].

While SGD remains a very popular optimization strategy, learning with it can sometimes be slow [43]. So the SGD with momentum algorithm is actually employed as given in algorithm 2, to train deep CNNs. The method of momentum [44] is designed to accelerate learning, especially in the face of high curvature, small but

Algorithm 2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ_i, momentum rate α.
Require: Initial parameter **W**, initial velocity **v**.
1: **while** stopping criterion not met **do**
2: Sampling a minibatch of M examples from the training set $\{x^{(1)}, ..., x^{(N)}\}$ with corresponding targets $y^{(i)}$.
3: Computing gradient estimate: $\hat{G} \leftarrow +\frac{1}{N} \bigtriangledown_W \sum_i J(f(x^{(i)}; W), y^{(i)})$
4: Computing velocity update: $v \leftarrow \alpha v - \epsilon \hat{G}$
5: Apply update: $W \leftarrow W + v$
6: **end while**
7: **return**

consistent gradients, or noisy gradients. The momentum algorithm introduces a variable **v**, which can be interpreted as the velocity that it is the direction and speed at which the parameters move through parameter space. The velocity accumulates an exponentially decaying moving average of the past negative gradient and continues to move in their direction. A hyperparameter $\alpha\epsilon[0, 1)$ determines how quickly the contributions of previous gradients exponentially decay. The greater α, the more previous gradients affect the current direction. For SGD, the size of the step was simply the norm of the gradient multiplied by the learning rate i.e. $\epsilon \|G\|$. For SGD with momentum, the size of the step also depends on the momentum rate i.e. $\frac{\epsilon\|G\|}{1-a}$. The momentum rate α is set to 0.9 in our experiments that corresponds to multiplying the step size by 10 relative to the standard SGD. It effectively stabilizes the gradient descent, as illustrated in Fig. II.13, the red path cutting across the contours indicates the path followed by the momentum learning, and the arrows indicate the step that SGD would take at that point without momentum. It can be seen that **momentum** traverses the long narrow valley, while SGD steps back and forth across the narrow axis of the valley.

II.4.4 Parameter Initialization

Training deep CNNs is a sufficiently difficult task that optimization algorithms are strongly affected by the choice of initialization. The initial point can determine whether the algorithm converges at all, with some initial points being so unstable

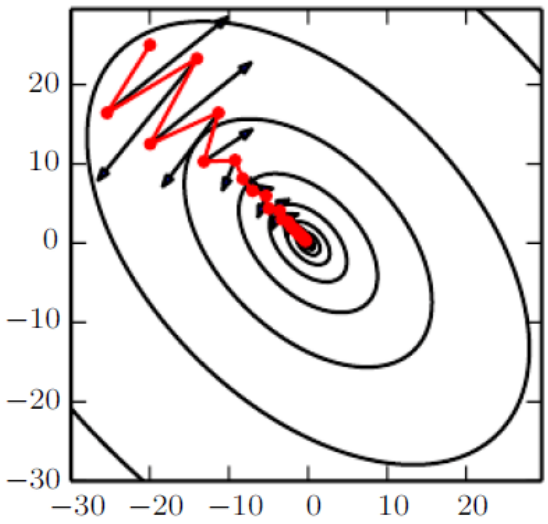


Figure II.13: Stochastic gradient descent with momentum (adopted from [34]. **Contour lines**: example of a quadratic loss function.

that the algorithm encounters numerical difficulties and fails altogether. Also, points of comparable cost can have wildly varying generalization error, and the initial point can affect the generalization as well. In practice, all the weights in the model are initialized to values drawn randomly from a Gaussian or uniform distribution. The scale of the initial distribution, however, does have an effect on both the outcome of the optimization procedure and on the ability of the network to generalize.

Some heuristics are available for choosing the initial scale of the weights. One heuristic is to initialize the weights of a Convolutional layer with the **filter size 3×3**, the **input feature dimensionality** N_c^{in} and the **output feature dimensionality** N_c^{out} by sampling each weight from $U(-\frac{1}{3\sqrt{N_c^{in}}}, \frac{1}{3\sqrt{N_c^{out}}})$, Glorot and Bengio [45] suggest using the normalized initialization;

$$W \sim U(-\sqrt{\frac{6}{3^2(N_c^{in}+N_c^{out})}}, \sqrt{\frac{6}{3^2(N_c^{in}+N_c^{out})}}) \tag{II.16}$$

This normalized initialization, known as **Xavier initialization**, is designed to compromise between the goal of initializing all layers to have the same activation variance and the goal of initializing all layers to have the same gradient variance. However, this initialization is derived using the assumption that the network consists only of linearity, with no non-linear activations. In practice, a theoretically more

sound initialization [35] is adopted by taking ReLU/PReLU into account. For a Convolutional layer, a response is;

$$z_l = W_l x_l + b_l \tag{II.17}$$

Here, $\mathbf{x}$ is a $3^2 N_c^{in}$-by-1 vector, $\mathbf{W}$ is a N_c^{out}-by-$3^2 N_c^{in}$ matrix representing the weights of a filter. $\mathbf{b}$ is a $N_c^{out} - by - 1$ vector of biases, and $\mathbf{z}$ is the response at a pixel of the output map. $\mathbf{l}$ is the layer index. $x_l = g(z_{l-1})$ where $\mathbf{g}$ is the activation. W_l is initialized with Gaussian distribution. As in [45], it assumes that the elements in x_l are mutually independent and share the same distribution, and x_l and W_l are independent of each other. Then it has;

$$Var[z_l] = 3^2 N_c^{in} Var[W_l x_l] \tag{II.18}$$

W_l is initialized with zero mean. Then the variance of the product of independent variables gives;

$$Var[z_l] = 3^2 N_c^{in} Var[W_l] E[x_l^2] \tag{II.19}$$

Here $E[x_l^2]$ is the expectation of the square of x_l. Note that $E[x_l^2] \neq Var[x_l]$ unless x_l has zero mean. In this thesis, the ReLU activation is used, this leads to $E[x_l^2] = \frac{1}{2} Var[z_{l-1}]$. Substituting this into Eqn. II.19, it obtain;

$$Var[z_l] = \frac{1}{2} 3^2 N_c^{in} Var[W_l] Var[z_{l-1}] \tag{II.20}$$

With L layers put together, it has;

$$Var[z_L] = Var[z_1] \prod_{l=2}^{L} \frac{1}{2} 3^2 N_c^{in} Var[W_l] \tag{II.21}$$

As in [45], all layers are initialized to have the same activation variance. A sufficient condition is;

$$\frac{1}{2} 3^2 N_c^{in} Var[W_l] = 1, \forall l \tag{II.22}$$

This leads to a zero-mean Gaussian distribution whose standard deviation is $\sqrt{\frac{2}{3^2 N_c^{in}}}$;

$$W_l \sim N(0, \sqrt{\frac{2}{3^2 N_c^{in}}}) \tag{II.23}$$

This is the way of initialization used in this thesis. b_l is initialized to 0. Above derivation, it only takes the forward-propagation into account, it also wants to consider the backprop that gradients should also have the same variance, this leads to a initialization $W_l \sim N(0, \sqrt{\frac{2}{3^2 N_c^{out}}})$. In our experiments, it found both initialization works very well.

II.4.5 Batch Normalization

Training Deep CNNs is difficult by the fact that the distribution of each layer's inputs varies significantly during training, as the parameters of the previous layers change. Therefore, it requires careful initialization as discussed in the last subsection to ensure the variance of activation and gradient are approximately the same in all layers. A good choice of initial learning rate is also required, as SGD is used with momentum (discussed above) so that the learning rate is fixed without adaptive to the importance of features. To cope with the difficulties, **Batch Normalization** [46] is incorporated in building the convolutional layers in CNNs. Batch normalization is actually a method of adaptive reparameterization acting as a regularizer, which allows us to use higher learning rates and be less careful about initialization. In practice, an update to the parameters for one layer actually depends very strongly on all of the other proceeding layers. The reparameterization provided by batch normalization significantly reduces the problem of coordinating updates across many layers. Let $\mathbf{H}$ be a minibatch of linear activations of a convolutional layer to normalize, arranged as a design matrix, with the activations for each example appearing in a row of the matrix. To normalize $\mathbf{H}$, it is replaced with;

$$H' = \frac{H - \mu}{\sigma} \tag{II.24}$$

where μ is a vector containing the mean of each unit and σ is a vector containing the standard deviation of each unit. H' is obtained via broadcasting the vector μ and the vector σ to be applied to every row of the matrix $\mathbf{H}$. Within each row, the arithmetic is element-wise, so H_{ij} is normalized by subtracting μ_j and dividing

by σ_j. The rest of the network then operates on $H^{'}$ in exactly the same way that the original network operated on **H**. At training time,

$$\mu = \frac{1}{M}\sum_i H_i \tag{II.25}$$

and,

$$\sigma = \sqrt{\delta + \frac{1}{M}\sum_i (H-\mu)_i^2} \tag{II.26}$$

where δ is a small positive value such as 10^{-8} imposed to avoid encountering the undefined gradient of $\sqrt{z}$ at z=0. Importantly, it backpropogates through the operations for computing the mean and the standard deviation, and by applying them to normalize **H**. As a result, the normalization removes the effect of such an action and zeros out its component in the gradient so that the gradient will never propose an operation acting simply to increase the standard deviation or mean of h_i. This is a major benefit of the batch normalization approach. At test time, μ and σ are replaced by running averages that were collected during training time. This allows the model to be evaluated on a single example, without needing to use definitions of μ and σ that depend on an entire minibatch.

However, simply normalizing the mean and standard deviation of units can reduce the expressive power of the model. In other words, it may change what the normalized layer can represent. For instance, normalizing the inputs if a sigmoid would constrain them to the linear regime of the nonlinearity. To restore the expressive power, two trainable parameters γ and β are introduced to scale and shift the normalized value: $\gamma H^{'} + \beta$. The training and inference process are illustrated in algorithm 3. A convolutional layer is implemented that consists of an affine transformation followed by an element-wise ReLU: g(Wx+b) where W and b are learned parameters, g represents the ReLU nonlinearity. A fully-connected layer is formulated the same way. The batch normalization is applied immediately before the nonlinearity, by normalizing z=Wx+b. It could have also normalized the layer inputs x, but since x is the output of ReLU, its distribution are non-Gaussian and less amendable to batch normalization. In contrast, Wx+b is more likely to

have asymmetric, non-sparse distribution, that is more Gaussian [47] to produce activations with a stable distribution. Of note, since Wx+b is normalized, the bias b can be ignored since its effect will be canceled by the subsequent mean subtraction. Therefore, g(Wx+b) is replaced with g(BN(Wx)) where **BN** represents the batch normalization applied independently to each dimension of Wx, with a separate pair of learned parameters $\gamma^{(c_i)}$, $\beta^{(c_i)}$ per dimension. The batch normalization is

Algorithm 3 Batch Normalization (BN).

Require: Feature values of x over a mini-batch: $B = \{x_1, x_2, ..., x_M\}$ $x \epsilon R^{N_c}$. Initial Parameters to be learned: layer weights **W**, γ, β.
Ensure: $y_i = BN_{\gamma,\beta}(x_i)$
Training BN.
for t= 1 to M **do**
  for k= 1 to N_c **do**
    Computing mini-batch mean $\mu_B \leftarrow \frac{1}{M}\sum_{i=1}^{M} x_i$
    Computing mini-batch variance $\sigma_B^2 \leftarrow \frac{1}{M}\sum_{i=1}^{M}(x_i - \mu_B)^2$
    Normalizing $\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
    Scaling and Shifting $y_i \leftarrow \gamma \hat{x_i} + \beta$
    //For clarity, $x \equiv x^{(c)}$, $x \equiv y^{(c)}$, $\mu_B \equiv \mu_B^c$, $\sigma_B \equiv \sigma_B^c$
  end for
end for
Train network to optimize the parameters $W \cup \{\gamma, \beta\}$
Inference with learned parameters.
for k= 1 to N_c **do**
  Computing moving average of mean over training batches: $E[x] \leftarrow E_B[\mu_B]$
  Computing moving average of variance over training batches: $Var[x] \leftarrow \frac{M}{M-1} E_B[\sigma_B^2]$
  Computing BN: $y = \frac{\gamma}{\sqrt{Var[x]+\epsilon}} \cdot x + (\beta - \frac{\gamma E[x]}{\sqrt{Var[x]+\epsilon}})$
  //For clarity, $x \equiv x^{(c)}$, $x \equiv y^{(c)}$, $\gamma \equiv \mu_B^c$, $\gamma^c \equiv \beta^c$
end for

implemented to obey the convolutional property i.e. parameter sharing so that different elements of the same feature map, at different locations, are normalized in the same way. To achieve this, all activations are jointly normalized in a mini-batch, over all locations. In algorithm 3, **B** represents a set of all values in a feature map across both the elements of a mini-batch and spatial locations. It learns a pair of parameters $\gamma^{(c_i)}$ and $\beta^{(c_i)}$ per feature map, rather than per activation.

II.4.6 Dropout

Dropout [48] is applied to regularize the learning of deep CNNs. Dropout is a computationally inexpensive regularization technique that can be applied to deterministic feedforward architectures, like CNNs that predict an output **y** given input vector **x**. CNNs contain a series of hidden layers **l**= $\{1, ..., L\}$. Dropout trains an ensemble of models consisting of the set of all models that contain a subset of the variables in both **x** and **l**. The same set of parameters W is used to parameterize a family of distributions $P(y|x; W, bm)$ where **bm** is a binary mask determining which variables to include in the model. Each time an example is loaded into a minibatch, a different binary mask is randomly sampled to apply to all of the input **x** and fully-connected layers in the network. The mask for each unit is sampled independently from all of the others. The probability of sampling a mask value of one (causing a unit to be included) is a hyperparameter fixed before training begins, which is set to be 0.5. For many parameterizations of **P** (such as most multilayer perceptrons) the instantiation of different sub-models $P(y|x; W, bm)$ can be obtained by elementwise multiplication of **x** and **l** with the mask **bm**. Dropout training is similar to bagging [49], where many different models are trained on different subsets of the data. Dropout training differs from bagging in that each model is trained for only one step and all of the models share parameters. For this training procedure to behave as if it is training an ensemble rather than a single model, each update must have a large effect, so that it makes the sub-model induced by that **bm** fit the current input **x** well.

To make a prediction, an ensemble must accumulate votes from all of its members. Specifically, each model i produces a probability distribution $P^{(i)}(y|x; W)$. The prediction of the ensemble is given by the arithmetic mean of all of these distributions:

$$\frac{1}{N_m} \sum_{i=1}^{N_m} P^{(i)}(y|x; W) \tag{II.27}$$

In the case of dropout, each sub-model defined by mask vector **bm** defines a probability distribution $P(y|x; W, bm)$. The arithmetic mean over all masks is given by:

$$\sum_{bm} P(bm)P(y|x; W, bm) \tag{II.28}$$

where $P(bm)$ is the probability distribution that was used to sample **bm** at training time. It is not that obvious how to do arithmetic mean because the sum includes an exponential number of terms. Therefore, the geometric mean is used rather than the arithmetic mean of the ensemble members' predicted distributions. To guarantee that the result is a probability distribution, we impose the requirement that none of the sub-models assigns probability 0 to any event, and the resulting distribution is renormalized. The unnormalized probability distribution defined directly by the geometric mean is given by:

$$\tilde{P}_{ensemble}(y|x; W) = \sqrt[2^d]{\prod_{bm} p(y|x; W, bm)} \tag{II.29}$$

where d is the number of units that may be dropped. To make predictions, the ensemble must be re-normalized:

$$P_{ensemble}(y|x; W) = \frac{\tilde{P}_{ensemble}(y|x; W)}{\sum_{y'} \tilde{p}_{ensemble}(y'|x; W)} \tag{II.30}$$

A key insight [50] involved in dropout is that $P_{ensemble}(y|x; W)$ can be approximated by evaluating $P(y|x; W) = softmax(x^T W_o + b)$ in one model: the model with all units, but with the weights going out of unit i multiplied by the probability of including unit i. The motivation for this modification is to capture the right expected value of the output from that unit. Because an inclusion probability of 0.5 is used, the weights are divided by 2 at the end of training, and then using the model as usual for testing. So $P_{ensemble}(y|x; W)$ is actually approximated by $P(y|x; W) = softmax(x^T W_o/2 + b)$ This is to make sure that the expected total input to a unit at test time is roughly the same as the expected total input to that unit at train time, even though half the units at train time are missing on average. The W/2 method has not been characterized mathematically, but performs well in practice.

II.5 Advances in Deep Learning

Traditional hand-crafted approaches for US pattern recognition usually require multiple procedures, which makes the tools inefficient and sometimes incompatible in clinical applications. Alternatively, the rise of deep learning techniques, such as CNNs (Convolutional Neural Networks), makes it possible to develop end-to-end trainable integrated frameworks for recognition tasks. Deep learning is based on multi-layer neural networks, including several different techniques, such as auto-encoder, DBM (Deep Boltzman Machine), CNNs (Convolutional Neural Networks), etc. This thesis focus on the applications of CNNs in image classification, video recognition, and object detection. The idea (CNNs) has been long existed [51], [52], and significantly raised the interest of the computer vision community recently. CNNs comprise several layers of non-linear feature extractors, which are more sophisticated than traditional hand crafted feature encoders. CNNs naturally integrate low/mid/high level feature and classifiers in an end-to-end hierarchical fashion, and features in each level can be enriched by the number of stacked layers (depth) [53]. This also explains why CNNs have outperformed traditional hand-crafted approaches by a large margin. Two major factors are responsible for the rising of CNNs: (1) the availability of much larger training datasets, such as ImageNet ILSVRC [54], with millions of labelled natural images; (2) Powerful GPU implementations make the training of very large models practical.

II.5.1 Large-scale Image Classification

Over the past few years, the advances of deep CNNs have achieved a great success on large scale natural image classification [8], [55], [56], [53], [57]. Krizhevsky et al. [8] proposed a deep CNN, known as AlexNet, which has 60 million parameters, consisting of five convolutional layers (some of which are followed by local response normalization, ReLU (Rectified Linear Units), and Maxpooling), and three fully-connected layers with a 1000-way softmax classifier. Notably, the AlexNet achieved

beating performance in 2012 ImageNet classification challenge with top-5 error rate of 16.4% compared to 2nd place result of 26.1% [58]. To understand why CNNs performed so well, [58] uses deconvolutional networks, proposed by Zeiler et al. [59], to visualise the activation of intermediate features layers in the input pixel space. It has shown that using large kernel and stride in the first Conv layer, such as AlexNet, results in a mix of extremely high and low frequency filter patterns in the 1st Conv layer and aliasing artefacts in the 2nd layer. Therefore, it proposed to use smaller kernel and stride i.e. 7×7 and 2 in the 1st Conv layer to keep finer low level information, which results in more distinctive features and fewer dead patterns in the low layers. More importantly, this modified model also outperforms the AlexNet by 1.7% in terms of ILSVRC-2012 test top-5 error rate. Inspired by AlexNet, [60] came up with three CNN architectures (Fast, Medium, and Slow in terms of training speed), namely CNN-F, CNN-M and CNN-S. CNN-F is similar to AlexNet that also takes 4 pixel stride in the 1st Conv layer to ensure a fast processing. The main difference between the CNN-F and AlexNet is the dense connectivity between convolutional layers ([8] used sparse connections to enable training on two GPUs). CNN-M is characterised by the decreased stride and smaller receptive field of the first Conv layer, which is similar to the one used by Zeiler and Fergus [16]. Unlike CNN-M, the stride in Conv2 is smaller (1 pixel), but larger max-pooling window (3×3) is used in Conv1 and Conv5 to compensate the increased spatial resolution. It has shown that both CNN-M and CNN-S outperform the CNN-F and [58] by a significant margin about 3% in terms of ILSVRC-2012 top-5 error.

[56], [53] and [57] demonstrated that network depth is of great importance. Simonyan and Zisserman addressed an important aspect of CNN architecture design i.e. depth in [56]. Rather than using relatively large receptive filed in Conv1 (such as [58] and [8]), they use small 3×3 convolution filters and fix convolution stride i.e. 1 pixel throughout the whole network. They came up with five architectures with an increase of depth. The depth of network is increased steadily by stacking more convolutional layers. The idea of stacking 3×3 Conv layers is to force a large (e.g. 5×5 or 7×7) layer decomposing through the small 3×3 filters

with non-linearity injected in between, which makes the decision function more discriminative. GoogleNet in [53], which is also based on very deep ConvNets (22 weight layers), introduced the Inception architecture to approximate an optimal local sparse structure. The Inception module consists of 1×1, 3×3, 5×5 convolutional layers and 3×3 maxpooling layers, whose outputs are concatenated into a single output vector forming the input of next layer. 1×1 convolutions are also used in the Inception module to reduce feature dimensionality before the expensive 3×3 and 5×5 convolutions. As mentioned in [56], this is also to increase non-linearity of decision function. The Inception allows for increasing both depth and width of networks without an uncontrolled blow-up in computational complexity. [56] and [53] achieved competitive results (top-5 error 6.8% vs 6.7%) in ILSVRC classification, but both substantially outperform the previous generation of shallow models mentioned in [58], [8], [60]. Deep CNN architectures, such as VGGVD and GoolgeNet, are prone to vanishing of gradients when back-propagating, which hamper the convergence of training. In [53], Szegedy and Ioffe addressed the problem by adding auxiliary classifiers connected to intermediate layers to train the network at different depth. Thereafter, [46] introduced an intermediate normalisation layer, known as batch normalisation, which enables deeper networks to converge. With batch normalisation, the performance of GooglNet is significantly improved (achieved top-5 error 4.9% compared to 6.7% in [53]). K.M. He et al. [57] has found that with an increased depth, networks performance gets saturated then degrades substantially. To address the problem, they introduced a deep residual learning framework. The idea is to learn a residual mapping through a stack of convolutional layers (e.g. 3×3 stack in VGG-Net) instead of directly leaning a desired mapping. This is achieved by shortcut connections which simply perform identity mapping, and their outputs are added to the outputs of the stacked convolutional layers. It is worth noting that they perform downsampling directly by convolutional layers that have a stride 2. Their 152-layer residual network achieved 4.49% top-5 validation error in ILSVRC 2015.

II.5.2 Video Recognition (Learning Spatio-temporal Features)

Encouraged by the great success of CNNs in static image classification, several studies [61], [62], [63] have recently proposed to apply CNNs to video classification tasks. [61] applied AlexNet to investigate approaches for fusing information over temporal dimension, including early fusion (the first layer filters convolve with a stack of consecutive frames so that first layers learns spatio-temporal features), slow fusion (consecutively increasing the temporal receptive field as approaching high level), late fusion (fusing fully connected layers of separate networks (sharing weights) that operate on temporally distant frames). The slow fusion model gradually collapses temporal information, which slightly outperforms the early fusion model (collapsing temporal information at very beginning i.e. 1st layer). Late fusion achieved the worst performance as it barely encodes any temporal features. They also found that a purely spatial network achieved the similar performance as their temporal fusion networks. This indicates the learnt temporal fusion networks poorly capture the motion features from evolving scenes. [62] demonstrated that 3D ConvNets (C3D) can well capture both appearance and motion information. In 3D ConvNets, convolution and pooling are performed spatio temporally (e.g. slow fusion model in [61]) while 2D convolution applied on a stack of frames treats them as different channels (e.g. early fusion model in [61]). In this work, they experimented on two types of 3D convolution i.e. homogeneous temporal depth (kernels size in temporal dimension is same) and varying temporal depth (kernel size in temporal dimension either increases or decreases as increased layers). As a result, $3 \times 3 \times 3$ is turned to be the best kernel choice for 3D ConvNets. They have shown that C3D performs significantly superior than both fusion models (in [61]) and the best hand-crafted features, namely improved dense trajectories (iDT) [64]. [63] proposed a two-stream 2D ConvNets architecture for video recognition. The method first decomposes a video clip into spatial component i.e. RGB frames, and temporal component i.e. stacking of optical flows. These components are fed into the separate

streams to extract appearance and motion features from the video clip. Each stream performs classification on its own and their softmax scores are combined by averaging fusion. They also experimented on different measures of stacking optical flow e.g. trajectory stacking, bi-directional stacking with different lengths. Their results demonstrated that the optimum stacking length is 10 and the difference between different stacking techniques is marginal. However, [65] points out that the two-stream architecture proposed in [63] has two main drawbacks: (1) The pixel-wise correspondences between spatial and temporal features cannot be learned since fusion is only performed on the classification scores, and (2) Spatial stream has limited temporal scale, which only operates on singe frame. To address (1), [65] investigated fusing features at different levels or multiple layers simultaneously by using different approaches, such as sum fusion, max fusion, conv fusion etc. They modified both spatial and temporal streams to be 3D ConvNets, which are operated on multiple temporal chunks sampled from a video. This allows temporal scale to be changed flexibly, which effectively addresses the limitation (2). Their results indicate that Max and Concatenation fusion perform considerably lower than Sum and Conv fusion, and Conv fusion performs the best. Moreover, their architecture significantly outperform the previous [61], [62] and [63]. To date, [65] is the state-of-the-art approach of applying CNNs to video recognition.

II.6 Obstetric Ultrasound Analysis

Before the success of deep learning, feature encoding has been demonstrated to provide a powerful approach to object detection from Ultrasound (US) videos. For instance, Maraci et al. [66] proposed a framework where images are initially pre-processed to create intensity invariant structures using local phase based feature symmetry maps, followed by learning a Bag of Visual Words (BoVW) encoding of densely computed SIFT features and an SVM classification. Instead of using SIFT features, [67] uses Fisher Vector (FV) encoding. That work also demonstrated

that using structured random forest edge detection instead of feature symmetry significantly outperforms previously published methods.

However, convolutional neural networks (CNNs) are currently transforming object detection in natural images so a natural thing is to consider how this class of method can be used to advantage for ultrasound image analysis. As introduced in Section II.3, CNNs are highly complex and non linear model which has powerful feature representation capacity. The capacity can be controlled by varying their depth and breadth, and they also use strong prior knowledge of images, for example, locality of pixel dependencies. Deep CNNs have achieved a great success on large scale natural image classification [8], [55], [56]. One of the key reasons is that it is now possible to collect labelled datasets with millions of natural images. For example, ImageNET [68] consists of over 15 million labelled natural images in about 22,000 categories. Medical Imaging, however, at least currently, is not able to provide this amount of "Big Imaging" information. This is likely to change with time, but current data ethics and data protection rules limit the ability to build large scale publicly available healthcare imaging resources.

II.6.1 Shallow Learning Approaches

Feature encoding has been demonstrated to provide a powerful approach to US image classification. Maraci et al. proposed a framework [69] where initially images are pre-processed to obtain their feature symmetry maps by using local phase based filters. Then, SIFT features [70] are created with a regular dense grid on the phase-symmetry maps (so called LP-SIFT) and vectorized into bag of visual words (BoVW) [71]. The visual words of the vocabulary are constructed by quantizing the feature space using k-means, where the cluster centres are the visual words and the number of clusters is defined by users. For each tile of a spatial filter, frequency of each visual word is represented in a histogram. The midlevel feature vectors are then created from these histograms and used to train an SVM classifier. Alternative to local phase feature symmetry maps, [67] proposed to use Structure Random

Forest (SRF) for edge detection for a more accurate structure representation. SIFT features computed on edge maps are then encoded with Fisher Vector (FV) [72], [73], which aims to reduce the loss of information caused by the hard quantization step in BoVW. It is worth noting that this work uses PCA to reduce the SIFT feature dimension and squared root the feature vectors (also known as RootSIFT) [74]. FV encoding, in this paper, fits a Gaussian Mixture Model (GMM) to the features and then encodes the derivatives of the log-likelihood of the model with respect to its parameters. First and second order differences between each descriptor and its Gaussian cluster mean are accumulated, appropriately weighted by the Gaussian soft-assignments and covariance, leading to a high-dimensional feature representation. SVM classification is trained on the encoded FV. It has shown that FV encoding significantly outperforms the BoVW encoding in terms of accuracy.

In contrast to single frame classification, detecting sequence of interest (SOI) in fetal ultrasound videos is more interested in the context of clinical diagnosis. To analyze the temporal evolution of dynamic scenes, traditional pipeline takes use of Linear Dynamic Systems (LDSs) to model the dynamic context of a sequence of frames. A recognition framework is set up then to learn the LDS parameters corresponding to a SOI. Finally, a metric is defined between the system parameters to compare the models for sequence classification. For example, [75] proposed to use dynamic texture (DT) model [76], in which, the appearance of each video frame is determined through the observed variable, and the motion and dynamics in the video over a given time is determined through the hidden-state variables sampled from a Gauss-Markov process. In such a model, the observed frame at any given time can be constructed from a linear combination of the hidden state variables. It is worth noting that a kernelized version of the DT model, known as KDT, is used where the evolution of the hidden states of the model are still kept linear, but the output matrix is replaced by a non-linear kernel function so as to capture the dynamics of the video. A distance metric is defined to assess similarity between any two sub-sequence models. This work uses the Binet-Cauchy singular value kernel [75], which has shown performing significantly better than the Martin Distance [77].

SVM classification is then trained on the distance metric to detect SOI. Although, the dynamic texture model has shown a great ability to a wide variety of complex patterns of motion and appearance into a simple spatio-temporal model. Maraci et al. [78] applied dynamic texture modelling with hand-crafted rotation-invariant feature for detection of a fetal heartbeat. Bridge et al. [79] proposed a framework based on Sequential Bayesian Filtering to predict visibility, position and orientation of fetal heart in consecutive frames. However, one major limitation of the dynamic texture framework is it is not good at modelling video where there are multiple regions of distinct motion [80]. For example, in many of the fetal US videos collected in my study, heart appears at different regions of abdomen, which is highly variant in terms of spatial and motion patterns (various scales, speeds, views and orientations). Therefore, there is a great need of more powerful encoder which can extract spatiotemporal representations from multiple regions of consecutive frames.

II.6.2 Deep Learning Approaches

Recently, deep learning has become popular for analyzing 2D fetal ultrasound sweeps, e.g. [6], [81], [82], [83], [84] and 3D fetal ultrasound [85], [86]. Chen et al. [6] constructed a domain transferred CNN for standard abdominal plane detection, which was the very first attempt of transfer leaning in ultrasound image analysis. However, one of the drawbacks of that work is that it requires feeding forward subimages through the CNN for multiple times to compute final classification scores. Gao et al. [81] has extensively studied transfer learning for the classification of fetal ultrasound images, in which the authors proposed an experimental setting to quantitatively explore the transferability of features learnt from natural images to ultrasound images recognition. [7] introduced a Fully Convolutional Network (FCN), called SonoNet, for the classification of 13 standard fetal planes. This work reports good classification results, however, it performs poorly in differentiating cardiac views. [87] and [81] also suffer the same problem.

The above models have exhibited state-of-the-art performance in still image tasks such as classification or detection. However, such models discard temporal information providing important cues in videos. Consequently, Gao et al. [83] proposed a two stream ConvNet, learning spatial-temporal representations to detect the fetal heartbeat in ultrasound videos, which demonstrated a substantial improvement in correctly identifying heart frames. Recurrent neural networks (RNNs) have recently been introduced for learning ultrasound video representation in [87] and [88]. Chen et al. [87] proposes a recurrent network for detection of standard fetal ultrasound planes, in which they apply LSTMs built on whole image features that completely discard spatial correspondence between frames. It should be noted that the improvement is marginal with respect to detection, compared to their previous work in [6]. A closely related work to ours is Huang et al. [88] who incorporates a convolutional recurrent layer working at a local region level of the frames. However in contrast to this thesis, the recurrent design is only applied on the coarsest feature map extracted from the CNN. The argument is that this design is more likely to focus on global appearance changes and is not suited for capturing fine temporal local details, which is non-trivial in our case. This thesis is different to [88] in two distinct ways; firstly a general detection network is proposed for detecting different anatomy with varied appearance, scale and location in free hand videos, whilst they only detect the fetal heart in a very constrained setting i.e. fetal echocardiography. Secondly, localisation in this thesis is learnt from image-level labels only, whilst in their scenario the position of target anatomy was annotated in each frame.

To the best knowledge of the author, the proposed work is the first to introduce a gated mechanism for visualizing spatial-temporal attention. Schlemper et al. [89] also proposed a soft attention-gated network for improving ultrasound scan plane detection, which is actually an improved SonoNet [82], however, this model focused on spatial attention only which did not exploit temporal information. In [89] the authors implemented three aggregation strategy, in contrast, in this thesis it only exploits the simplest and most stable aggregation approach i.e. training separate

classifiers at each scale. In addition, building weakly-supervised classifies have also been investigated extensively with the adaptation approach [90] of aggregating the final class score maps, and class activation mapping (CAM) approach [91] that fit a fully-connected (FC) layer for mapping the class scores.

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [9]

III

Classification of Fetal Ultrasound Images with CNNs

Contents

III.1 Introduction **48**
III.2 Dataset **49**
III.2.1 Pre-processing 50
III.2.2 Annotation 50
III.3 Learning Deep Ultrasound Representations from Scratch **54**
III.4 Transfer Learning and Domain Adaptation **58**
III.5 Visualization: Deep Inside CNN **61**
III.5.1 Deconvolution 61
III.5.2 Class-Specific Saliency Visualization 63
III.5.3 t-Stochastic Neighbour Embedding (t-SNE) 65
III.6 Experiments and Results **68**
III.6.1 Implementation Details 69
III.6.2 Quantitative Evaluation 72
III.6.3 Visualization 81
III.7 Discussion and Conclusion **86**

In this chapter, we address the task of image classification in obstetric ultrasound using deep Convolutional Neural Networks (CNNs). The state-of-the-art CNN models have been studied extensively and the regularization methods, such as dropout and data augmentation and their effects on reducing overfitting for training

deep CNNs. Moreover, a transfer learning based design is also presented to study the transferability of features learnt from natural images to ultrasound image recognition which on the surface is a very different problem. It shows the transfer learning and its variants can bridge the gap between deep CNNs and medical applications caused by the limited training dataset. An early version of the work presented in this chapter has been reported in:

Y Gao, Maraci, JA Noble. "Describing ultrasound video content using deep convolutional neural networks". Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI), 2016 [81].

III.1 Introduction

A number of recent studies [92], [93], and [94] have demonstrated that transfer learning provide a powerful way to reduce overfitting by using the knowledge of a pre-trained network to solve a target task which has limited number of data. The question are therefore interested is "Do features learnt from natural images generalize to be useful for US image machine learning tasks?" The answer is not necessarily obvious, as the appearance of ultrasound images and videos is quite different from natural images - acoustic ridges, ultrasonic texture patterns as well as shadows are characteristic in this case. Specifically, in this chapter, CNNs are first trained on a small-scale database of thousands of labelled US images directly to identify four type of frames, which contain the fetal skull, abdomen, heart and "other" structures. A transfer learning approach is then proposed to experimentally quantify the transferability of features learnt from ImageNET to ultrasound video frame recognition. A number of transfer learning models are compared with direct learning models in terms of their classification performance.

Visualization of image classification models, learnt using deep convolutional networks was also considered. Three visualization techniques were implemented. One is to visualize the high-dimensional feature vectors via dimensionality reduction,

the other two are to visualize the class-specific input stimuli via backpropagation. Visualization helps to gain insight into the internal operation and behavior of these complex deep models, to understand how they achieve good or bad performance. It is of great importance to help identifying the strengths and limitations of the models so that the models can be improved accordingly if necessary.

III.2 Dataset

The clinical data used in the study consisted of 323 fetal ultrasound sweeps acquired to a freehand sweep protocol. The clinical data was acquired using a mid-range US machine (Phillips HD9 with a V7-3 transducer) by five experienced obstetricians following a simple predefined scanning protocol, as illustrated in figure III.1. The obstetricians were informed of the objectives of the study and were asked to perform the scan moving the ultrasound probe from the cervix up to the top of the maternal abdomen. Each sweep takes about 6 to 8 seconds. The participants included in this study were healthy pregnant volunteers at 28^{+} weeks of gestation, who were taking part in the International Fetal and Newborn Growth Consortium for the 21 Century project (INTERGROWTH-21^{ST}) [25], [95], [96]. The INTERGROWTH-21^{ST} was designed to develop new standards for defining normal fetal growth, preterm growth and newborn nutritional status. It was conducted in collaboration with eight different sites with geographically diverse populations around the world. The study only included participants with a low risk impairment of fetal growth or risk factors for IUGR (alcohol or smoking) and clinical history (diabetes or chronic illness). Full study demographics can be found from Villar et al. [96]. Moreover, ultrasound video sweeps for this thesis were acquired from some of the participants in the Oxford site of the INTERGROWTH-21ST project. These participants were selected at random and the only selection criteria was a gestational age of 28 weeks or over.

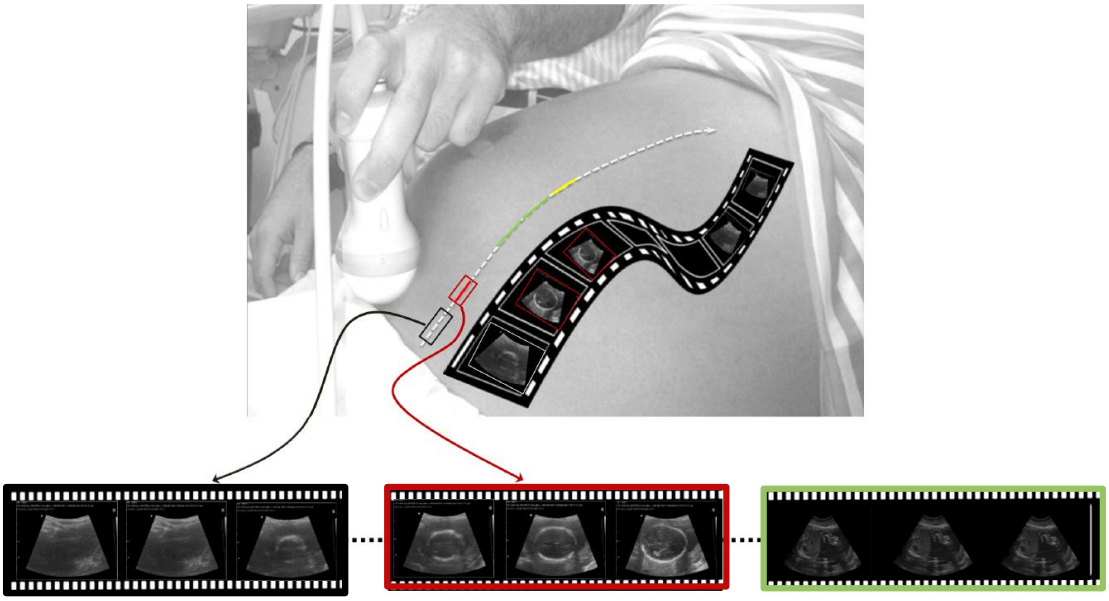


Figure III.1: A simple free-hand scan protocol. A straight ultrasound video sweep is obtained from the cervix up to the top of the maternal abdomen. Three sample frames are shown from various points in the ultrasound video sweep illustrating "Other Structures"(left), "Fetal Skull" (centre) and "Fetal Abdomen" (right).

III.2.1 Pre-processing

The acquired video sweeps were pre-processed in steps that are illustrated in Fig.III.2. A mask was created that fits to the ultrasound field of view to exclude any text. The masked frames were downsampled to half their original size, from 460 × 640 to 230 ×320 pixels, using Gaussian pyramid reduction for the trade-off that there is an improved computational efficiency at cost of the loss of image details. This is because the deep CNNs presented in this chapter were trained on a CPU with limited computational power and also because of memory limitation that it was not able to process batches of images at the original resolution i.e. 460×640.

III.2.2 Annotation

The position and angle in which the ultrasound probe is held directly affects the visual appearance of anatomical structures. As shown in Figure III.3, holding the transducer at different insonification angles results in various fetal brain anatomies

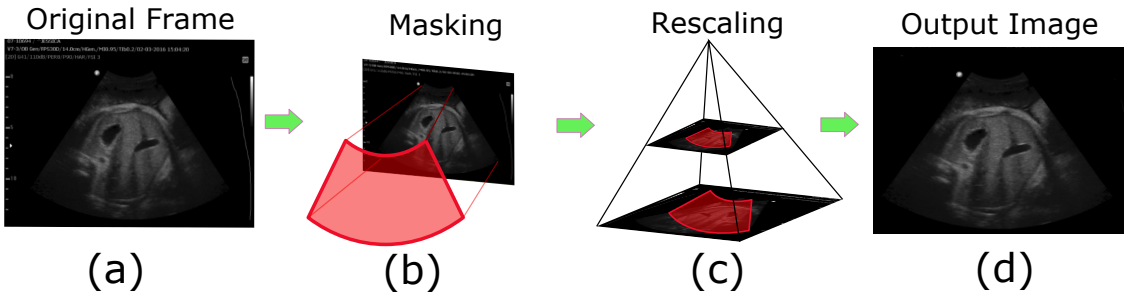


Figure III.2: Dataset Preprocessing. (a) Original frame (b) Frame is masked to keep the data at the ultrasound's "field of view" only. This is mainly to remove the text that appears on the top left and right hand sides of the image (c)Frame is downsampled to half it's original size for computational efficiency (d) Output frame

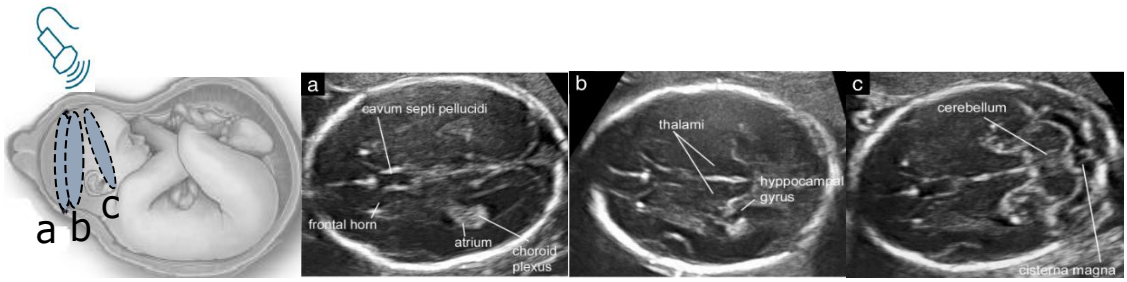


Figure III.3: Scan Fetal Brain with various probe's position and angle. The appearance of anatomical structures depends on the position and angle of the ultrasound probe. Here three different planes of the fetal brain are illustrated, where different anatomies are visible on each plane. **(a) Transventricular plane; (b) Transthalamic plane; (c) Transcerebellar plane.**

appearing or disappearing from the frames. Subject to the appearance of key anatomical landmarks satisfying a specific set of criteria , the 2D ultrasound images are acquired in a particular plane, referred to as the standard planes illustrated in Fig.III.3. The qualitative metrics already exist for various anatomical structures, they are mainly defined for structures that have been captured in their standard plane and not for structures that are out of their standard plane. According to the International Society of Ultrasound in Obstetrics and Gynecology (ISUOG) criteria for a standard plane [97], the transventricular plane in Fig.III.3 (a) should include the anterior and posterior portion of the lateral ventricles. The anterior portion of the lateral ventricles (front or anterior horns) appears as two comma-shaped fluid filled structures. They have a well defined lateral wall and medially are separated by the cavum septi pellucidi (CSP), which is a fluid filled cavity between two thin membranes. The transthalamic plane in Fig.III.3 (b), is obtained at an intermediate

level in parallel with the transventricular plane, in which the anatomic landmarks include, from anterior to posterior, the frontal horns of the lateral ventricles, the cavum septipellucidi, the thalami and the hippocampal gyruses. The transcerebellar plane in Figure III.3 (c) is obtained at a slightly different angle and lower level than that of the transventricular plane and with a slight posterior tilting and includes visualization of the frontal horns of the lateral ventricles, CSP, thalami, cerebellum appearing as a butterfly shaped structure formed by the round cerebellar hemispheres joined in the middle by the slightly more echogenic cerebellar vermis.

However, given the simple predefined free-hand protocol in Fig.III.1, the chance of capturing key anatomies in their standard plane can be low. Due to the inherent flexibility of the predefined free-hand scanning protocol, the visual appearance of anatomical structures can vary significantly in different scans. Therefore, other suitable criteria need to be defined in order to be able to formally characterize anatomical structures of interest. The new criteria should be set loosely to facilitate reproducibility of data by sonographers of different levels of expertise and also provide the means to quantitatively evaluate the learning algorithm's performance. Considering the clinical application of interest, it is of great importance that the fetal skull and abdomen are captured as part of the acquired ultrasound video in order to detect the fetal presentation. Therefore, in collaboration with experienced sonographers, new criteria have been defined for characterizing and indeed recognizing the fetal skull and abdomen in a predefined free-hand ultrasound sweep. The criteria are schematically illustrated in Fig.III.4 and Fig.III.5 which demonstrate that capturing these anatomical structures in their standardized planes is indeed highly unlikely so that is not required. Given that the structures of interest can now be formally characterized, methods to automatically identify video sequences that contain these structures of interest will be introduced and discussed in the next two chapters.

According to the new criteria, fetal structures of interest are categorized into four classes, which are the fetal skull, abdomen, heart and other structures, respectively.

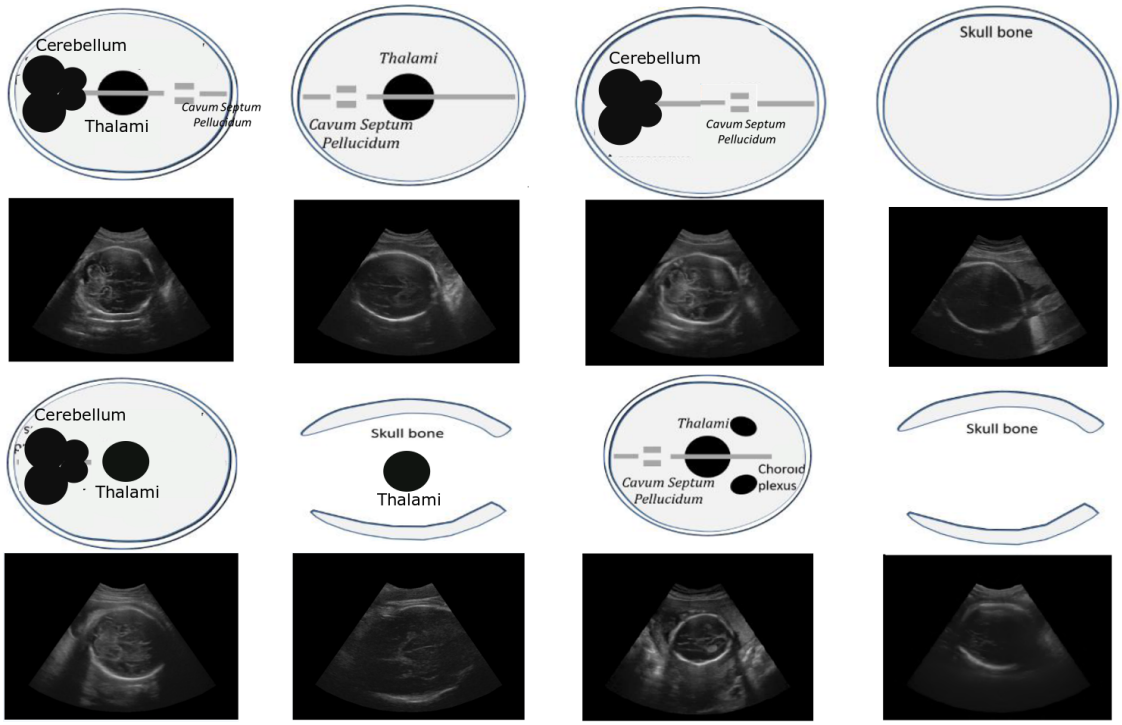


Figure III.4: Annotation criteria for fetal skull. The only necessary criterion is presence of a semi-elliptical shape with or without the other standard key landmarks. The criteria are set loosely and independent of the presence of key anatomical structures.

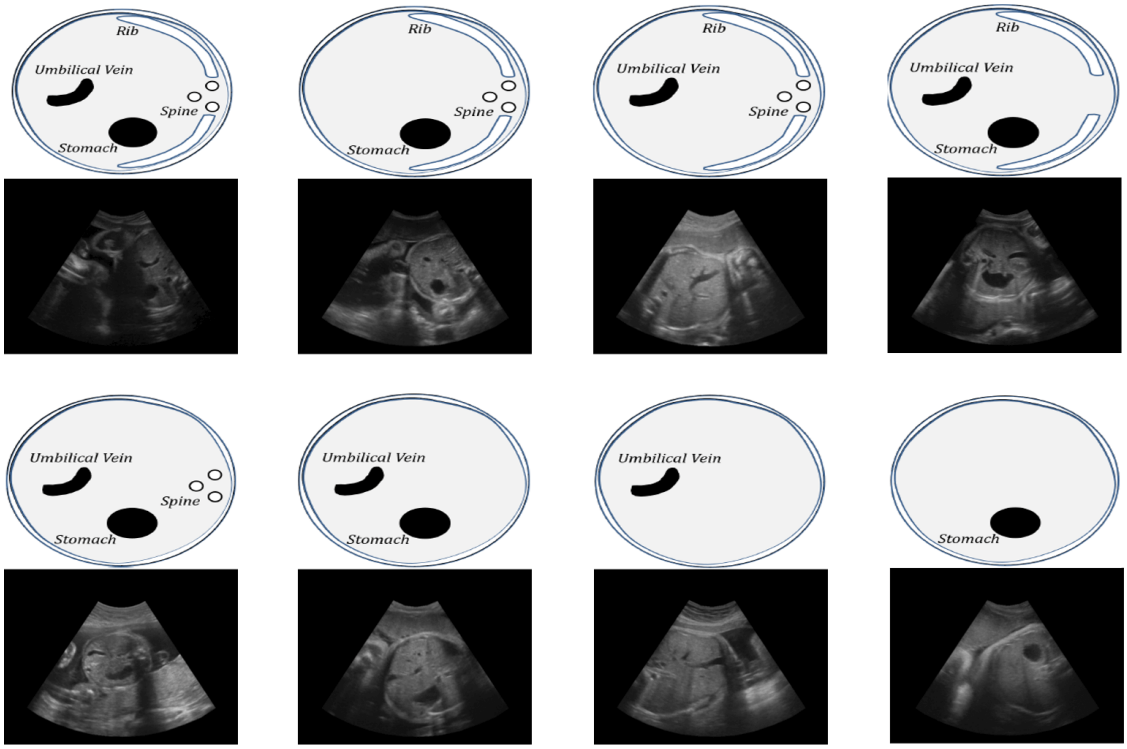


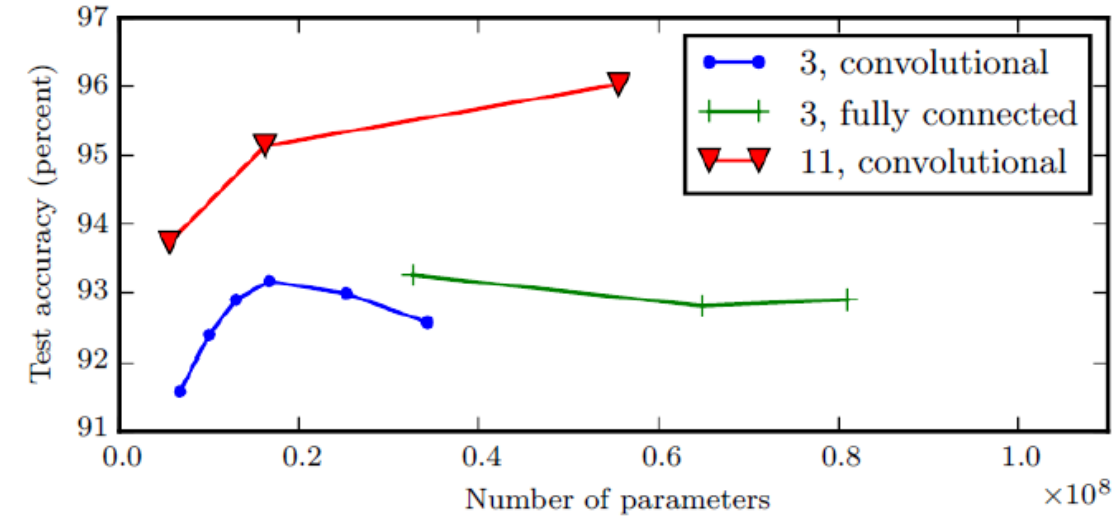
Figure III.5: Annotation criteria for fetal abdomen. The only necessary criterion is presence of at least one of the key landmarks i.e. stomach bubble and umbilical vein.

The "other structures" class included any video frame that does not fall into the other three classes. One reason for using the four classes along is, of clinical interest, it is important to have the classes i.e. fetal skull and abdomen in a scan to decide fetal presentation. Fetal heart resides in the thoracic cavity which is in between the fetal skull and abdomen so it should be considered as an independent class rather than a sub-category of abdomen. Also, this annotation criteria facilitates the reproducibility of data by sonographers of different levels of skill. More importantly, It get rids of an extremely imbalanced classification which facilitates deep learning. There will be more classes if the fetal structures (e.g. fetal heart) are sub-categorised into their standard planes (e.g. FCV, 3VV, LVOT typically) but at the same time, there will be only few training samples for many categories that leads to an extremely imbalanced classification. An image database was created consisting of 9,730 skull images, 5,009 abdomen images, 3,617 heart images and 16,177 the other images. It should be noted that the classes are still imbalanced in terms of the number of samples, because only 30% of the videos collected contain heart frames but more than 60% contain skull frames. And there is normally multiple instance of background frames i.e. the "other" frames in each video so that makes this class the largest in terms of the number of images, compared to the other three classes. In order to keep the training unbiased, only part of images were used for training. 2,705 images were randomly sampled from each category to give a training set of 10,820 images. In addition, a small set of 3,648 images is kept for testing. During the training stage, data augmentation was applied to the raw frames by extracting five 227×227 patches (the four corner patches and the center patch) and these patches are flipped horizontally. This generates 10 training samples from each original image.

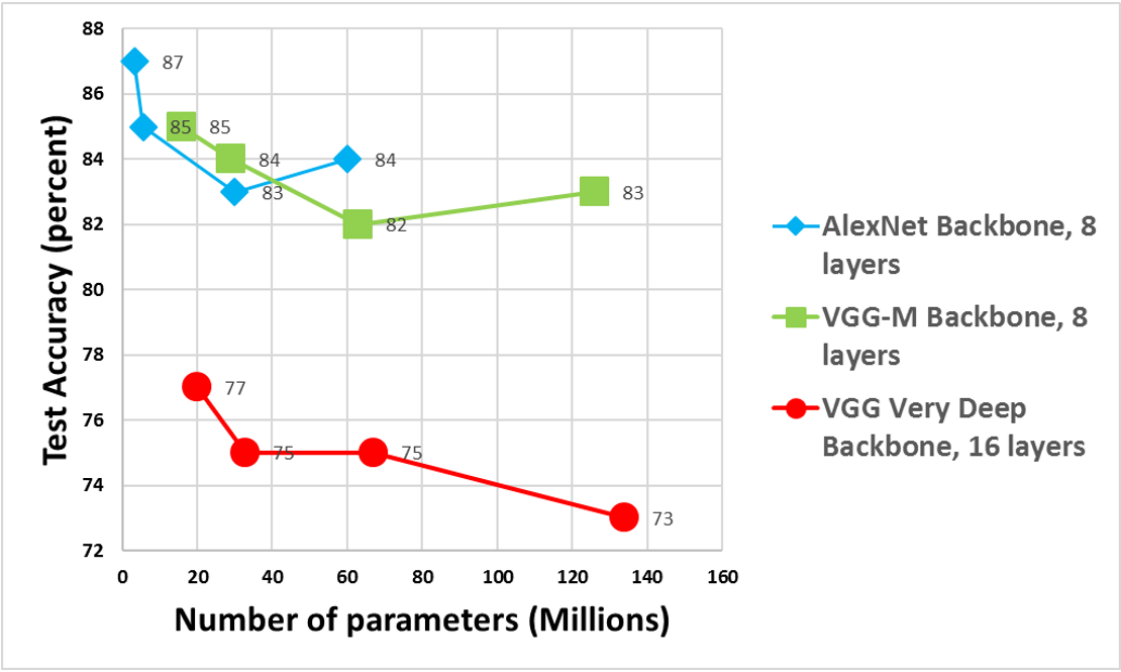
III.3 Learning Deep Ultrasound Representations from Scratch

CNN architectures arrange layers in a chain structure, with each layer being a function of the layer that preceded it. CNNs produce hierarchical feature

representations typically from high to low resolution in a feed-forward way. It is of great importance to choose the depth of the network and the width of each layer. From a representation learning perspective, CNNs learn a complicated function that is composed of several simpler functions. The learning consists of discovering a set of underlying factors of variation [34]. Goodfellow et al. [98] suggest that using deep architecture does indeed express a useful prior over the space of functions that the model learns. The experiments in Goodfellow et al. [98], as depicted in Fig.III.6, show increasing the number of parameters in CNN layers i.e. widening layers without increasing their depth is not nearly as effective at improving test set performance. It can be observed that shallow CNNs (blue curve) overfit at around 20 million parameters while deep ones can benefit from having wider layers i.e. over 60 million parameters. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler object representations e.g. edges or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other and recognize them). The capacity of deep CNNs has been studied on classification of fetal ultrasound images that instead of training on millions of images (like [98]) natural images recognition), CNNs in this chapter were trained on a much smaller training set (as mentioned earlier in this chapter) consisting of around 10 thousands fetal ultrasound images. Therefore, it is of great importance to choose a proper CNN architecture that fits to the scale of training. Three state-of-the-art CNN architectures were tailored (e.g. modify convolutional layers' width, remove some of neurons in FC layers etc) and experimented that Fig.III.6 shows that it has little effect on the test accuracy when changing the number of trainable parameters at same depth (such as AlexNet and VGG-M, both are 8 layers) but there is a substantial performance drop when going too deep (for instance, the 16 layers VGG very deep CNN).



(a)



(b)

Figure III.6: Test accuracy against the number of trainable parameters. **(a)** Deeper models tend to perform better than shallow models. Fully convolutional models tend to perform better than fully connected models. (adopted from[98]);**(b)** States-of-the-art deep CNNs trained on classification of fetal ultrasound images in this chapter. Deeper architecture tends to perform worse than shallow one.

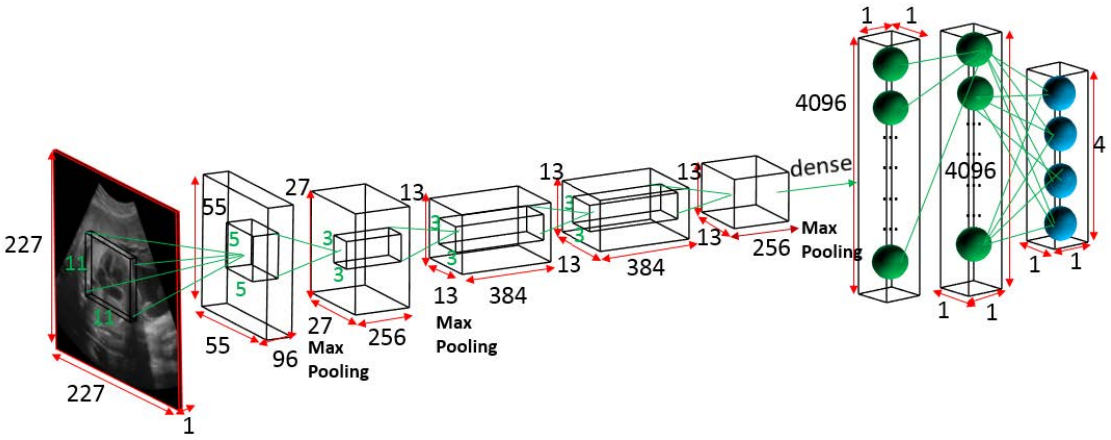


Figure III.7: A schematic of the CNN used in this paper. 96-256-384-384-256 are the number of feature maps generated in each convolutional layer. 55-27-13-13-13 are the sizes of the feature maps. There are 4096 neurons in each of the fully connected (FC) layers (2 FC layers). There are 4 neurons in the output layer i.e. the last layer.

Figure III.7 illustrates the architecture of a baseline CNN used in this chapter. The design [8] adopts relatively large kernels and stride in the first two convolutional layers which gives an aggressive reduction of spatial resolution. This effectively improves the computational efficiency and accelerates the training process. However, drawbacks of the design include that the accuracy of classification may compromise with the large stride, and the lower level features are less detailed due to the usage of large kernels [55]. The third, fourth and fifth convolutional layers have 3×3 kernels stacked together with Rectified Linear Units (ReLUs) between each other. This can be seen as imposing a regularization on the 7×7 convolutional filters, forcing them to have a decomposition through the 3×3 filters (with non-linearity injected in between). The design significantly increases the non-linearity of the network and makes the decision function more discriminative, hence makes their feature representations more expressive [56].

In order to train the CNNs, parameters including the convolution kernels $\mathbf{W}$ and the bias $\mathbf{b}$ are updated by minimizing a loss function $L(I_i, y_i; W)$ via, in practice, Stochastic Gradient Descent (SGD) with momentum. The loss function

used in the paper is defined as:

$$J(I_i, y_i; W) = -\sum_{i=1}^{N}\sum_{k=1}^{C}(1\{y_i = k\} log P(y_i = k|I_i; W)) \tag{III.1}$$

$$P(y_i = k|I_i; W) = \frac{exp(W_k^T x_i)}{\sum_{j=1}^{C} exp(W_j^T x_i)}, k = 1, ..., C \tag{III.2}$$

where, $1\{\cdot\}$ is the indicator function and y_i is the predicted label; $W_k^T x_i$ is the output of kth neuron in the last layer; $P(y_i = k|I_i; W)$ is the posterior probability of predicting the training sample I_i as the kth class among total **C** classes;

III.4 Transfer Learning and Domain Adaptation

Domain transfer (also called knowledge transfer or transfer learning) often refers to the use of knowledge learned from one source domain to efficiently develop a more accurate hypothesis for a new target task only consisting of small numbers of training examples in another domain [99]. This section explores transferring the knowledge learned from natural images to enhance the classification of fetal ultrasound images. With large differences in image appearance between the two domains, why and how the knowledge can be transferred from natural image domain to medical image domain to boost the classification performance for medical applications remains an open problem. Recently, Chen et al.[6] experimentally demonstrated the feasibility of transferring knowledge from natural image domain to medical image domian through sparse coding. They assume that ultrasound images and natural images share similar statistical pattern in low level features, e.g. oriented edges, corners and junctions. In this regards, the low level representations learned from natural images can be transferred to the medical image domain. To validate the assumption, they calculate the residual errors of reconstructing target ultrasound images with the dictionaries learned from the natural, ultrasound and corrupted images with random noise, respectively. They found that the residual errors are very small and approximately equal when reconstructing ultrasound images with the dictionaries learned from the natural and ultrasound images, while

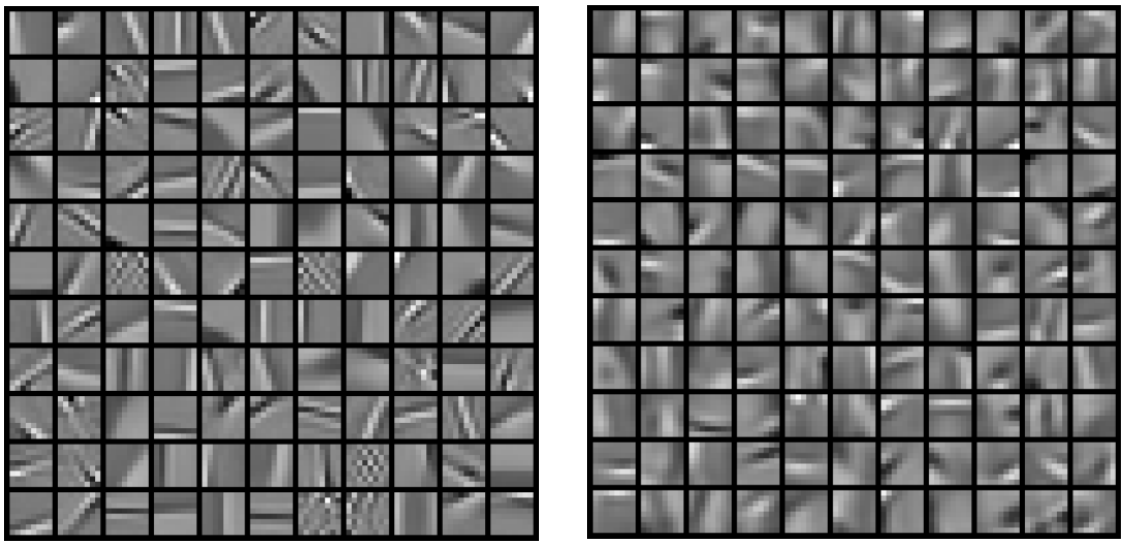


Figure III.8: Dictionary bases learned from: (Left) Natural images, (Right) US images. (Adopted from [6])

the residual error is significantly larger when constructing the ultrasound images with dictionaries learned from random Gaussian noises. This study has also shown qualitatively that ultrasound images and natural images do share similar statistical pattern in low level features, as illustrated in Figure III.8.

The first-layer features are so called general features as finding them seems to occur regardless of the training objectives and image modalities. On the other hand, it is known that the features computed by the last layer of a trained CNN must depend greatly on the chosen dataset and task. For example, in a CNN with a softmax output layer that has been successfully trained toward a supervised classification objective, each output unit will be specific to a particular class. The last-layer features are so called specific features. If first-layer features are general and last-layer features are specific, there must be a transition from general to specific somewhere in the CNN. It is of interest to explore, to what extent that features within a CNN are general, where the transition from general to specific takes places.

To answer the questions, a transfer learning approach is proposed that a base network was trained first and then copy its first n layers to the first n layers of a target network. The remaining layers of the target network are then randomly

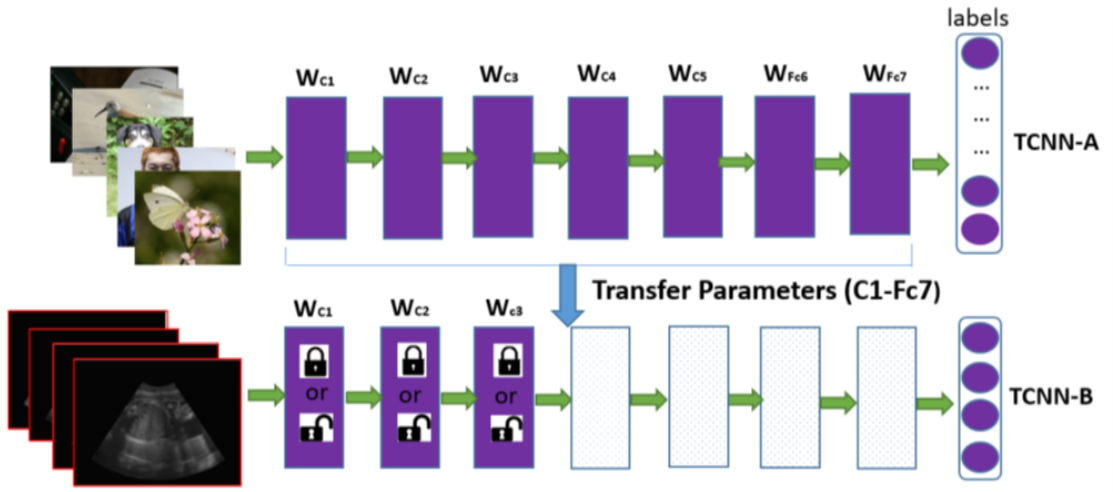


Figure III.9: Overview of transfer learning framework. **Top row**: The TCNN-A is pre-trained on the ImageNet database for classification. The labelled purple rectangles (e.g. W_{c1}) represent the weights learned by that layer. **Bottom row**: In the TCNN-B, the first l weight layers (in this example, l = 3) are copied from the base network A. The remaining layers are randomly initialized, and then the entire network is trained on ultrasound datasets with the transferred parameters locked (frozen) or unlocked (fine-tuned).

initialized and trained towards the target task. One can choose to backpropagate the errors from the new task into the base (copied) features to fine-tune them to the new task, or the transferred feature layers can be left frozen, meaning that they do not change during training on the new task. The Transfer learning approach is illustrated schematically in figure III.9. The base CNN (denoted TCNN-A), having the same architecture as described in figure III.7, was pre-trained on ImageNet. The parameters of the first l layers (l= 1,2,..,7) of the TCNN-A were transferred to another CNN, denoted as TCNN-B, and the remaining layers of TCNN-B were randomly initialized as described above in the Section 3.2. The parameters of TCNN-B were either frozen or fine-tuned by feeding ultrasound datasets. The results from each of these two techniques: fine-tuned or frozen features will be compared and discussed in Section III.6.2.

III.5 Visualization: Deep Inside CNN

Despite the success of deep CNNs in image recognition tasks, there has been little insight into the internal behavior of deep models, or how they achieve such good performance until recently. Visualizing features to gain intuition about the CNN is common practice, but has been mostly limited to the first layer where projections to pixel space are possible [58]. In higher layers alternate methods must be used. This section will discuss the visualization techniques that were employed to understand what is actually learned from fetal ultrasound images classification. The author not only visualized the low-level ultrasound representations, but also managed to look into the high-level representations. Some of the techniques, such as deconvolution [58] and saliency [100], reveal the input stimuli that excite individual feature maps and the activation of the target output. Some techniques, such as t-SNE [101], project high-dimensional feature vectors in a two-dimensional map which helps to better understand misclassified samples. Next, each of the three methods will be explained.

III.5.1 Deconvolution

Understanding what is learned by a CNN requires interpreting the feature activity in intermediate layers. A deconvolution approach [102] was used to map these activities back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps. Deconvolution can be thought of as a CNN model that uses the same operations (filtering, pooling) but in reverse, so instead of mapping pixels to features does the opposite. To do so, an unpooling operation is employed as illustrated in Figure III.10, which is an approximate inverse of the max-pooling operation by recording the locations of maxima within each pooling region in a set of switch variables. Deconvolution uses these switches to place the reconstructions from the layer above into appropriate locations, preserving the structure of the stimulus. Deconvolution provides an upsampling path back to image

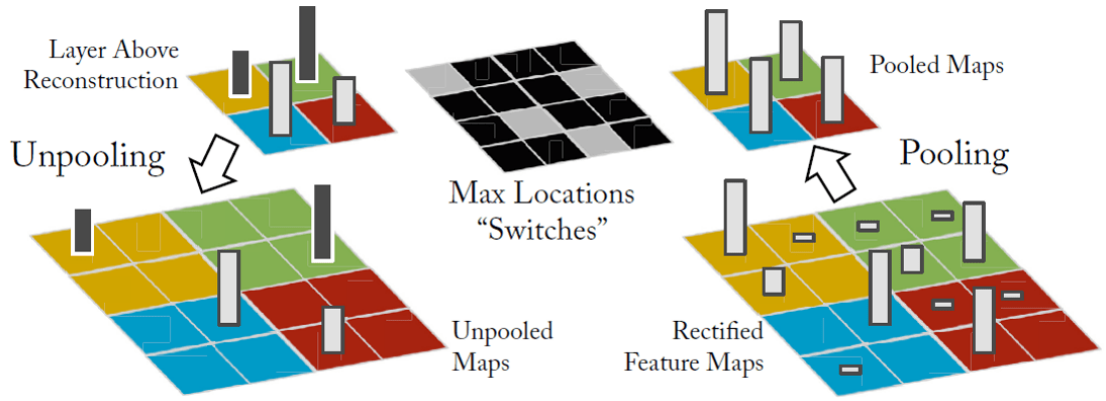


Figure III.10: An illustration of the unpooling operation, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the CNN. The black/white bars are negative/positive activations within the feature map. (Adopted from [58])

pixels as shown in Figure III.11. To start, an input image is presented to the trained CNN and features computed in a hierarchical way of downsampling throughout the layers. To visualize a given feature map at a specific layer, all other feature maps in the layer were set to zero and pass the sparse representation as input to the deconvolution process, which successively unpool, rectify and filter to reconstruct the activity in the layer beneath that gave rise to the chosen activation. This is then repeated until input pixel space is reached. ReLU non-linearity was used in the feed-forward CNN during training, which rectify the feature maps thus ensuring the feature maps are always positive. To obtain valid feature reconstructions at each layer, which should also be positive, the reconstructed signal (unpooled maps) was passed through a ReLU non-linearity. Deconvolution uses transposed versions of the filters (in practice, equivalent to flipping the filters vertically and horizontally) learned at each layer of the CNN, applied to the rectified maps. Projecting down from higher layers uses the switch settings generated by the max pooling in the feed-forward process on the way up. Because these switches settings are specific to a given input image, the reconstruction obtained from a single activation thus resembles a small piece of the original input image, with structures weighted according to their contributions toward the feature activation. Since the CNN is trained discriminatively, the reconstructions implicitly show which parts of the input

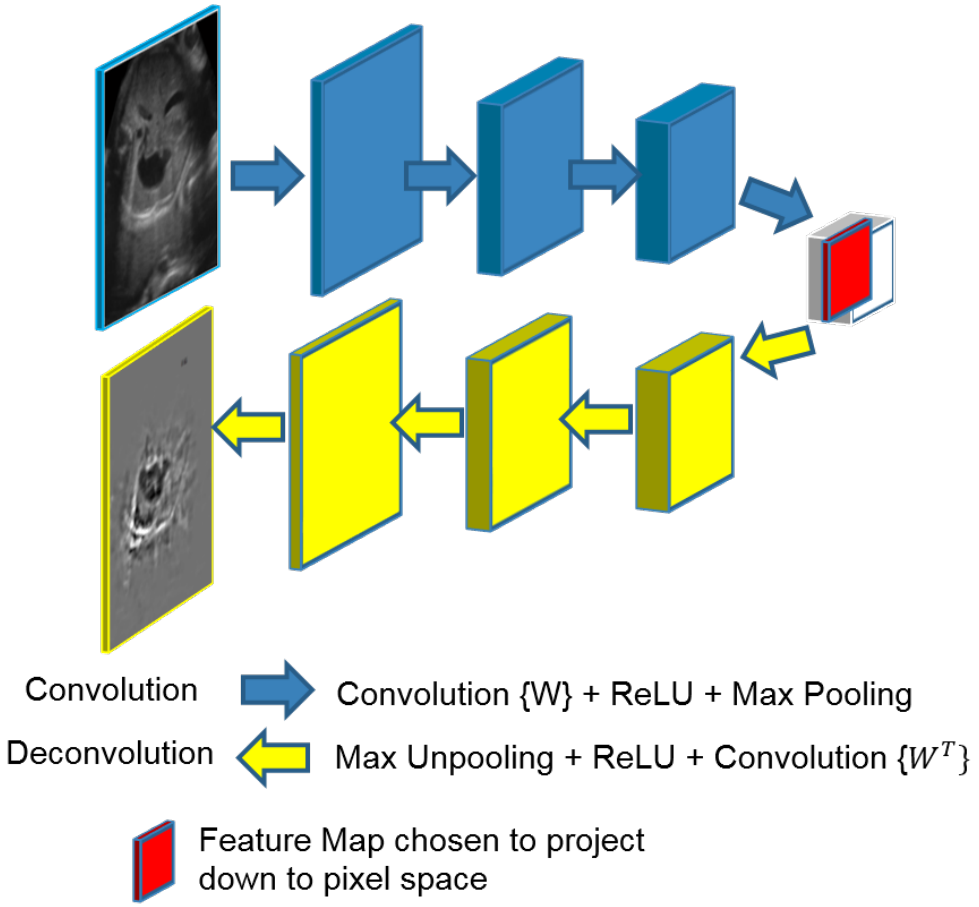


Figure III.11: Feature visualization by deconvolution that reconstructs a high-level feature map in the pixel space.

image are discriminative. The deconvolution was employed to visualize feature patterns learned at different layers of a trained CNN. The whole procedure of deconvolution is similar to backpropagating a single strong activation (rather than the usual gradients), i.e. computing $\frac{\partial y_k}{\partial x}$, which will be discussed in Section III.5.2 that y_k here is the output score of a specific class and $\mathbf{x}$ is the input image.

III.5.2 Class-Specific Saliency Visualization

Instead of backpropagating the strongest activated unit in a feature map, the class-specific score (should be the strongest activated unit in the output layer) of a well trained CNN was actually backpropagated to compute its derivatives with respect to the input image. A method proposed by [100] was used that ranks

the influence of an image, **x**, according to its influence on a specific class score. The method proceeds by linearizing the relationship between a specific output class score $y_k(x)$ and the input image **x**:

$$y_k(x) = W_k^T x + b_k \tag{III.3}$$

It is easy to see that the magnitude of elements of W_k define the importance of the corresponding pixels of **x** for the class **k**. In the case of deep CNNs, the class score $y_k(x)$ is a highly non-linear function of **x**, so the reasoning can not be immediately applied. However, given an image I_0, $y_k(x)$ can be approximated with a linear function in the neighbourhood of I_0 by computing the first-order Taylor expansion;

$$y_k(x) \approx W^T x + b \tag{III.4}$$

where **W** is the derivative of $y_k(x)$ with respect to the image **x** at the image I_0;

$$W = \frac{\partial y_k}{\partial x}|I_0 \tag{III.5}$$

The magnitude of the derivative indicates which pixels need to be changed the least to affect the class score the most. One can expect such pixels correspond to the object location in the image. Given an image I_0 (with m rows and n columns) and a class **k**, the class saliency map $SM \epsilon \Re^{m \times n}$ is computed as follows. First, the derivative **W** in equation III.5 is found by back-propagation. After that, the saliency map is obtained by rearranging the elements of the vector **W**. In the case of a grey-scale image, the number of elements in **W** is equal to the number of pixels in x_0, so the map can be computed as $SM_{ij} = |W_{ij}|$, where **ij** is the index of the element of **W**, corresponding to the image pixel in the **i-th** row and **j-th** column. It is of importance to note that the saliency maps are extracted using a classification CNN trained on the image labels, so no additional annotation is required, such as object bounding boxes or segmentation masks. The computation of the image-specific saliency map for a single class is extremely quick, since it only requires a single back-propagation pass. As mentioned above, deconvolution-based reconstruction of the **l-th** layer input x_l is actually similar to computing the gradient of the visualized

neuron activity y_k with respect to x_l, hence deconvolution effectively corresponds to the gradient back-propagation through a CNN.

For the convolutional layer $x_{l+1} = x_l * W_l$, the gradient is computed as $\frac{\partial y_k}{\partial x_l} = \frac{\partial y_k}{\partial x_{l+1}} * \hat{W}_l$, where W_l and $\hat{W}_l$ are the convolution kernel and its flipped version, respectively. The convolution with the flipped kernel exactly corresponds to computing the lth layer reconstruction R_l with deconvolution: $R_l = R_{l+1} * \hat{W}_l$. For the ReLU rectification layer $x_{l+1} = max(x_l, 0)$, the gradient takes the form: $\frac{\partial y_k}{\partial x_l} = \frac{\partial y_k}{\partial x_{l+1}} 1\{x_l > 0\}$, where 1 is the element-wise indicator function. This is slightly different from the deconvolution RELU reconstruction: $R_l = R_{l+1} 1\{R_{l+1} > 0\}$, where the sign indicator is computed on the output reconstruction R_{l+1} instead of the layer input x_l. For a max pooling layer $x_{l+1}(p) = max_{q \epsilon \Omega(p)} x_l(q)$, where the element p of the output feature map is computed by pooling over the corresponding spatial neighbourhood $\Omega(p)$ of the input x_l. The gradient is computed as $\frac{\partial y_k}{\partial x_l(s)} = \frac{\partial y_k}{\partial x_{l+1}(p)} 1\{s = argmax_{q \epsilon \Omega(p)} x_l(q)\}$. Here, argmax corresponds to the max unpooling switches in a deconvolution. So, apart from the ReLU layer, computing the approximate feature map reconstruction R_l using a deconvolution is equivalent to computing the derivative $\frac{\partial y_k}{\partial x_l}$ using backpropagation. Thus, gradient-based visualization can actually be seen as the generalization of the deconvolution, since the gradient-based techniques can be applied to the visualization of activities in any layer, not just a convolutional one. In particular, the class score neurons in the final fully-connected layer were visualized that will be seen in Section III.6.3. Next, a visualization technique that does not need backpropagation, will be introduced; instead of visualizing the input stimuli, it generates a map showing the clusters formed by each individual class.

III.5.3 t-Stochastic Neighbour Embedding (t-SNE)

Data visualization is an effective technique to directly show the discriminant capability of feature representations, hence can indicate the performance of classifiers. The t-Stochastic Neighbour Embedding (t-SNE) method [101] was employed to

reduce dimensions of raw training datasets, raw testing datasets, and features of fully-connected layers of CNNs extracted from the training and testing datasets.

SNE starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities. The similarity of datapoint x_j to datapoint x_i is the conditional probability, $P_{j|i}$, that x_i would pick x_j as its neighbour if neighbours were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby datapoints, $P_{j|i}$ is relatively high, whereas for widely separated datapoints, $P_{j|i}$ will be extremely small. Mathematically, the conditional probability $P_{j|i}$ is given by:

$$P_{j|i} = \frac{exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2)} \tag{III.6}$$

where σ_i^2 is the variance of the Gaussian that is centered on datapoint x_i. Because it is only of interest to model pairwise similarities, the value of $P_{i|i}$ was set to zero. The pairwise similarities in symmetric SNE has the property that $P_{i|j} = P_{j|i}$. The joint probabilities P_{ij} in the high-dimensional space was defined to be the symmetrized conditional probabilities as:

$$P_{ij} = \frac{exp(-\left\|x_i - x_j\right\|^2 / 2\sigma^2)}{\sum_{k \neq l} exp(-\left\|x_k - x_l\right\|^2 / 2\sigma^2)} \tag{III.7}$$

This ensures that each datapoint x_i makes a significant contribution to the cost function. In low-dimensional space, a Gaussian distribution was not used to model the pairwise similarity. This is because the distribution of pairwise distances in a low dimensional space is very different from its counterparts in a high dimensional space. If a Gaussian distribution is used to model the pairwise similarities in a low dimensional space, it can cause the very well known crowding problem [101] that a very large number of attractive forces crushes together the points in the centre of the map, which prevents gaps from forming between the natural clusters. Instead, a Student t-distribution was employed that has much heavier tails than a Gaussian to convert distances into probabilities. This allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance

in the low-dimensional map. As a result, it eliminates the unwanted attractive forces between map points that represent moderately dissimilar datapoints. The joint probabilities Q_{ij} in the low-dimensional space is defined as:

$$Q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{III.8}$$

If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j, the joint probabilities P_{ij} and Q_{ij} will be equal. t-SNE aims to find a low-dimensional data representation that minimizes the mismatch between P_{ij} and Q_{ij}, which is achieved by minimizing the sum of Kullback-Leibler (KL) divergences over all datapoints [101]. The loss function $\mathbf{L}$ is given by;

$$L = KL(P \parallel Q) = \sum_i \sum_j P_{ij} log \frac{P_{ij}}{Q_{ij}} \tag{III.9}$$

The minimization of the cost function is performed using SGD with momentum (discussed in Section II.4.3). The gradient of the KL divergence between P and Q is given by:

$$\frac{\partial L}{\partial y_i} = 4 \sum_j (P_{ij} - Q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} \tag{III.10}$$

Equation III.10 can be interpreted as the resultant force created by a set of springs between the map point y_i and all other map points y_j. All springs exert a force along the direction $y_i - y_j$. The spring between y_i and y_j repels or attracts the map points depending on whether the distance between the two in the map is too small or too large to represent the similarities between the two high-dimensional datapoints. The gradient descent is initialized by sampling map points randomly from an isotropic Gaussian with small variance that is centered around the origin. In order to speed up the optimization and to avoid poor local minima, a relatively large momentum term is added to the gradient. In other words, the current gradient is added to an exponentially decaying sum of previous gradients in order to determine the changes in the coordinates of the map points at each iteration of the gradient search. The gradient update is given by;

$$v^{t-1} = \alpha v^{t-2} - \epsilon \frac{\partial L}{\partial y^t} \tag{III.11}$$

$$y^t = y^{t-1} + v^{t-1} \tag{III.12}$$

where y^t indicates the solution at iteration t, ϵ indicates the learning rate, and α represents the momentum rate. The whole optimization process is summarized in algorithm 4. t-SNE emphasizes modeling dissimilar datapoints by means of

Algorithm 4 t-Distributed Stochastic Neighbor Embedding.

Require: data set $X = \{x_1, x_2, ..., x_n\}$
Initialize num of iterations **T**, learning rate ϵ, momentum α
Initialize velocity **v**
Ensure: low-dimensional data representation $y^T = \{y_1, y_2, ..., y_n\}$
compute pairwise affinities P_{ij} using equation III.7
sample initial solution $y^0 = \{y_1, y_2, ..., y_n\}$. from $N(0, 10^{-4}I)$
for `t=1 to T` **do**
    `compute low-dimensional affinities` Q_{ij} `using equation III.8`
    `compute gradient using equation III.10`
    `compute velocity update using equation III.11`
    `Apply update using equation III.12`
end for
return y^T

large pairwise distances, and modeling similar datapoints by means of small pairwise distances. As a result of the approximate scale invariance of the Student t-distribution, it is much faster and easier to evaluate the density of a point under a Student t-distribution than under a Gaussian because it does not involve an exponential. Particularly, t-SNE introduces long-range forces in the low-dimensional map that can pull back together two (clusters of) similar points that get separated early on in the optimization. The long-range forces in t-SNE facilitate the identification of good local optima without need of simulated annealing.

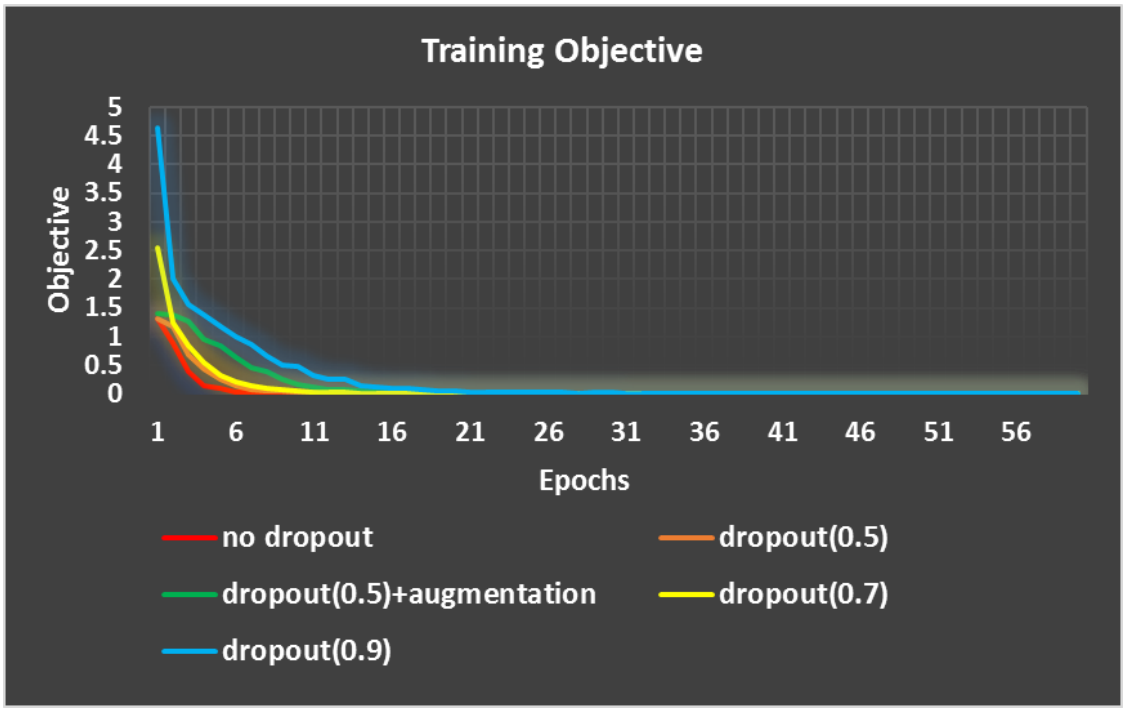
III.6 Experiments and Results

Experiments were designed to train the model (in Fig.III.7) from scratch (referred to as direct learning) for classification and compare with the results of transfer learning (in Fig.III.9). There were two settings of the direct learning that have been experimented with. The first experiment fixed the architecture, and the

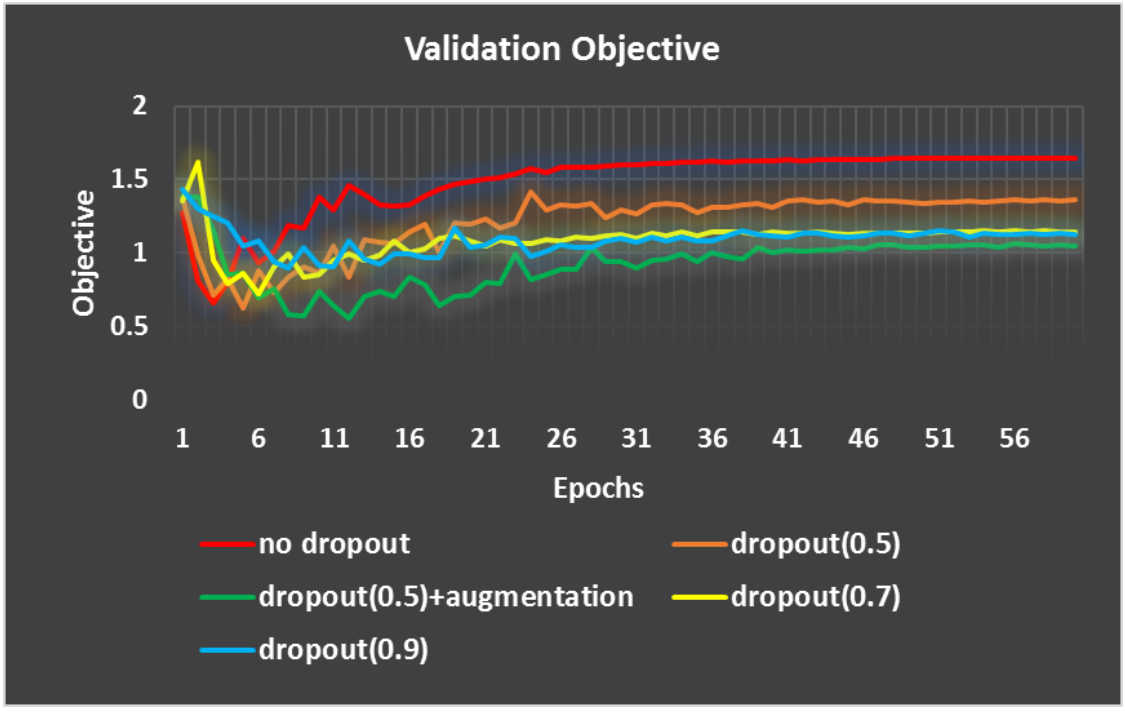
importance of data augmentation is evaluated. The second experiment studied the effect of reducing the network size on improving the generalization as well as the effect of batch normalization and dropout. The network size (approx. 60 million trainable parameters) was reduced by (1) half the feature channels in both convolutional layers and fully-connected layers resulting in approx. 30 Million trainable parameters; (2) half the feature channels in convolutional layers and use only one fully-connected layer with the number neurons reduced to 1024 (quarter of the 4096) resulting in significantly reduction of the number of trainable parameters to approx. 5.6 million; (3) half the feature channels in convolutional layers and use only one fully-connected layer with the number of neuron 512 resulting in approx. 3.3 million trainable parameters. The dropout was applied to the fully-connected layers only with the probability of 0.5, 0.7 and 0.9. The batch normalization was applied to both convolutional layers and fully-connected layers.

III.6.1 Implementation Details

Models (both direct learning and transfer learning) were trained using SGD with momentum (introduced in Section 3.2-optimization) with a batch size of 256 samples, momentum rate of 0.9, and weight decay of 0.0005. The way of initializing weights and bias in each layer were introduced in Section 3.2-Parameter Initialization. The number of epochs was set to 60 for learning without batch normalization, and 20 for learning with batch normalization. Training and validation were implemented separately for each epoch. Classification error was accumulated batch by batch for each class to compare classification performance between the classes. Learning rate ϵ was constant for all layers, and adjusted for each training epoch. The initial learning rate was set to 0.01 and continuously reduced to 10^{-4} until training termination.

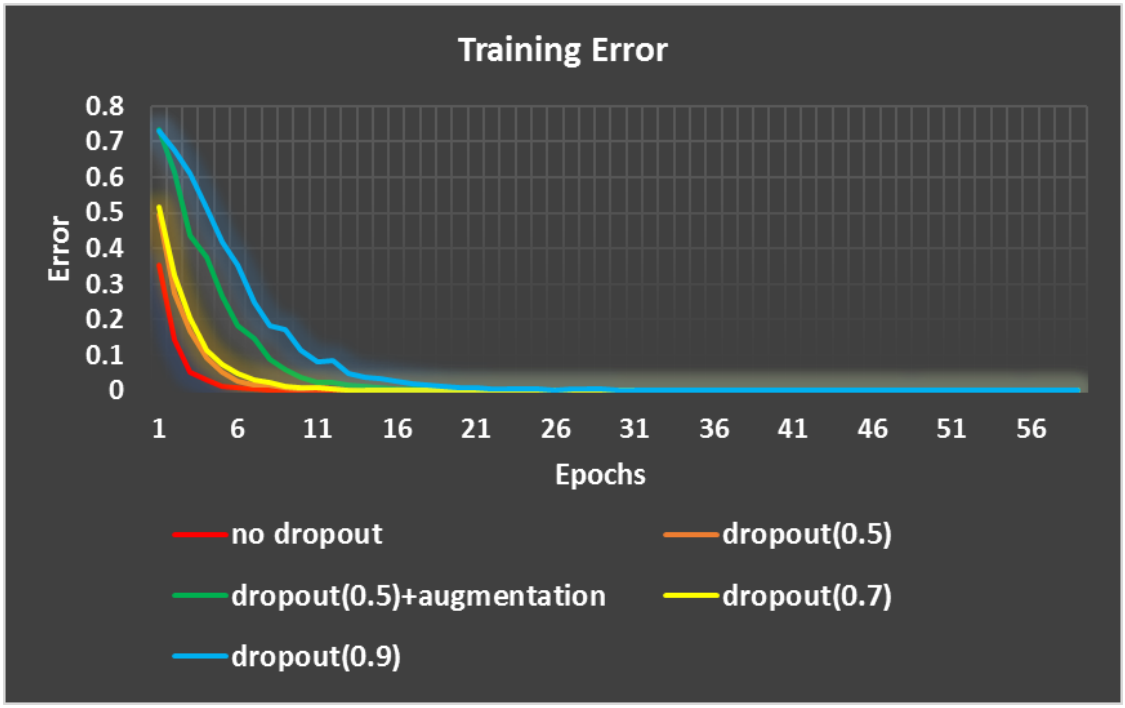


(a)

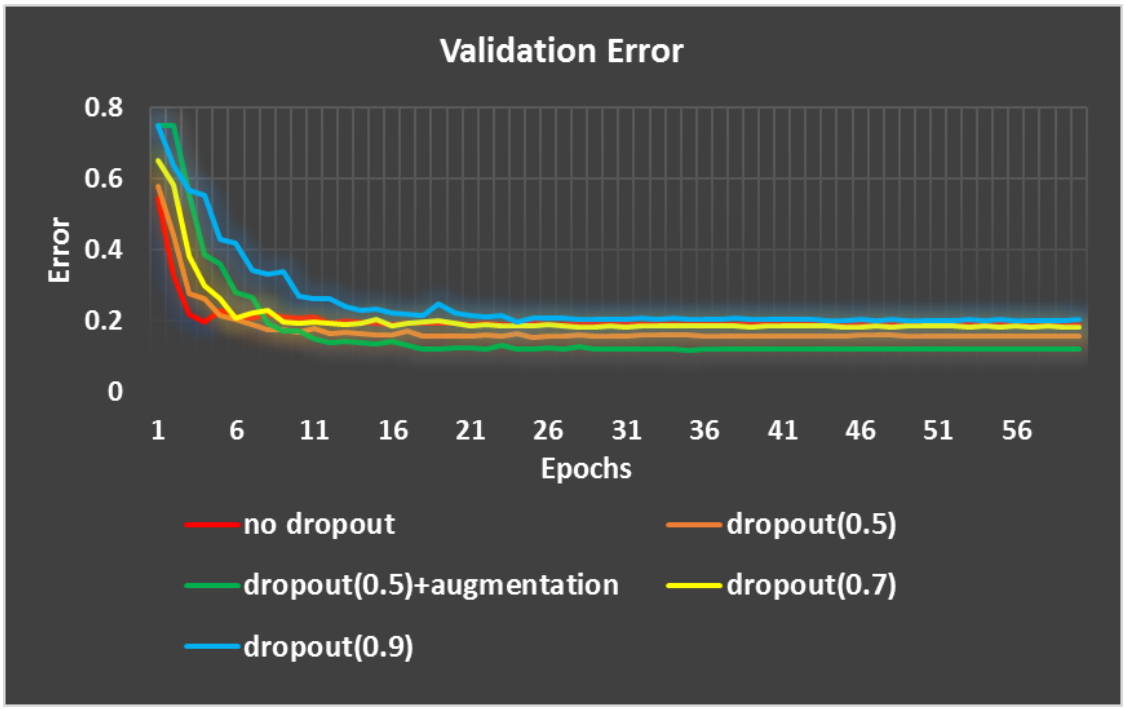


(b)

Figure III.12: Effect of data augmentation and dropout. **(a)** Training Objective (Loss), **(b)** Validation Objective.



(a)



(b)

Figure III.13: Effect of data augmentation and dropout. **(a)** Training Error, **(b)** Validation Error.

III.6.2 Quantitative Evaluation

Directly Learning Models (DCNNs)

Figure.III.12 and III.13 demonstrates that direct learning is prone to overfitting and prevent the representations learned from training data to generalize. It found that dropout and data augmentation are essential to regularize the learning and improve the generalization to the validation set. It should be noted from Fig.III.12a and Fig.III.12b that without dropout, the model is so badly overfitted that the training loss plummets to zero after about 10 epochs so as the training error in Fig.III.13a, and the validation loss starts increasing around 5 epochs (means stopping generalization). The increasing trend is less prominent when applying a higher dropout rate i.e. 0.7 and 0.9, as can be seen in Fig.III.12b that the validation loss is increasingly flattened (regularized) by applying stronger dropout and it takes longer to plummet to zero (around 20 epochs) for the training loss in Fig.III.12a. After applying augmentation to the training data, the validation loss is further regularized. The best classification accuracy is achieved by the model trained with dropout and data augmentation, as can be seen the green lines in Fig.III.12b and Fig.III.13b that learning stops to generalize to the validation and the validation error converges to approx. 0.11 after 20 epochs. Without any regularization, it should be noted (the red line in Fig.III.13b) that the learning stops to generalize to the validation after about 10 epochs and the validation converges to a much larger error about 0.2.

The classification error is broken down by class to examine the improvement closely as illustrated in Fig.III.14. It suggests by the 'standard' DCNN (without regularizations) that the class 'heart' is the most frequent misclassified class with an error rate of approx. 0.41 on the validation set (912 images per class). From the confusion matrices, it can be seen that the class heart is often misclassified as the abdomen, for example in Fig.III.14b, there are 328 (about 36% of) heart images being misclassified as 'abdomen'. By contrast, the best performance is attained by the class skull which achieves an error rate of approx. 0.08 and none of the

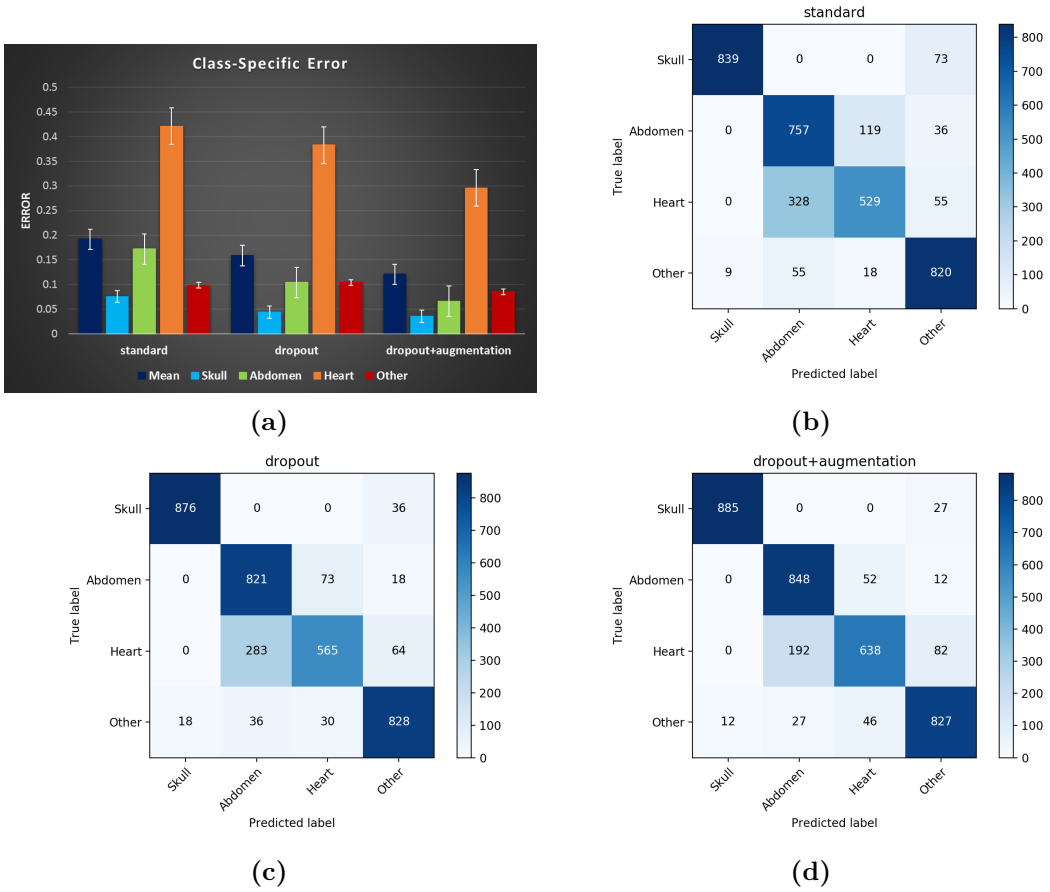
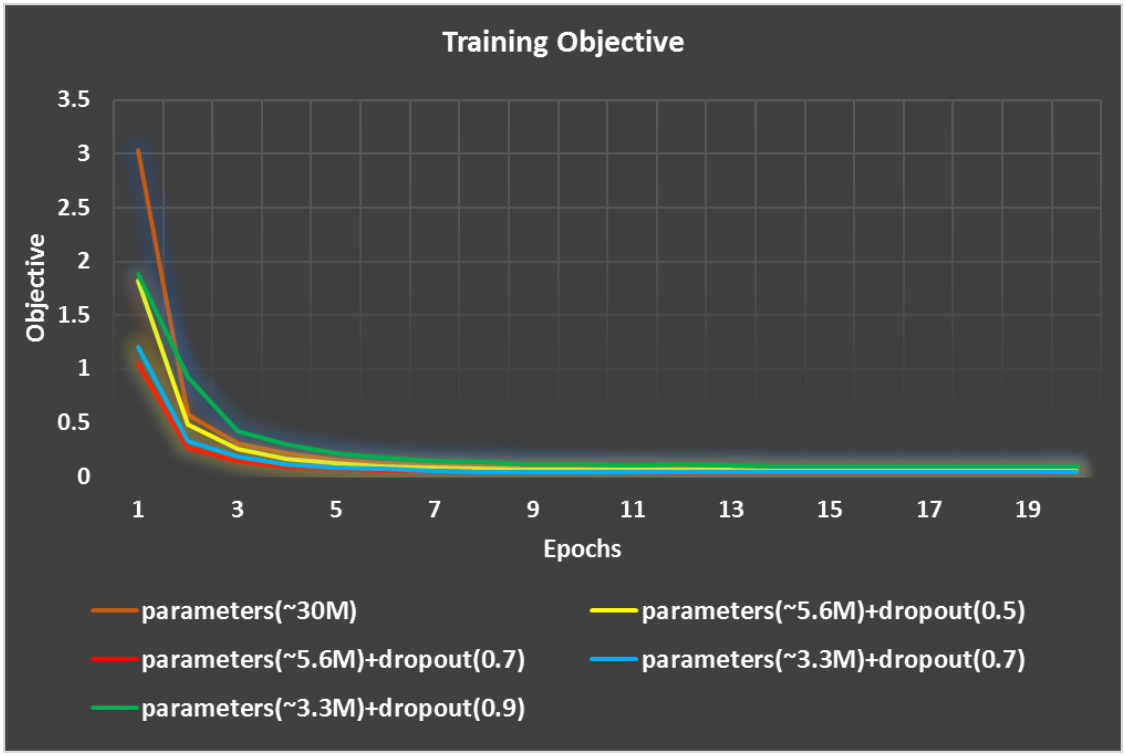
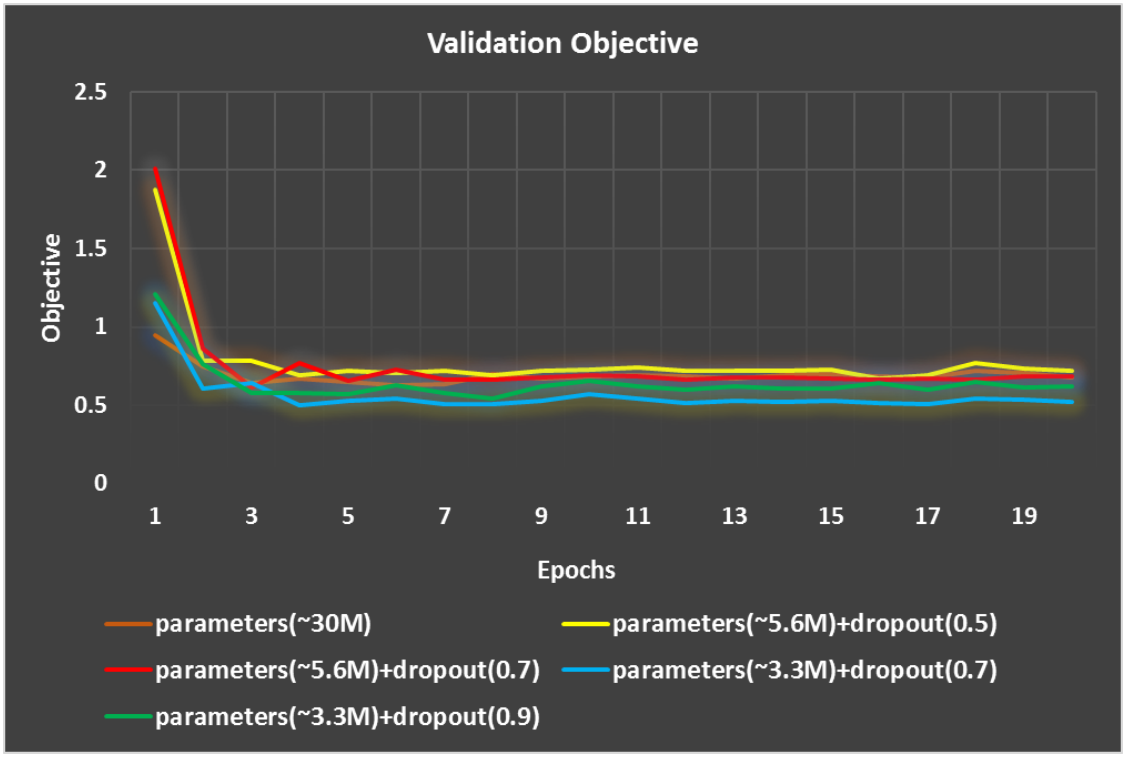


Figure III.14: class-specific Error. **standard**: without any regularization (corresponding red curves in Fig.III.13); **dropout**: 50% dropout to fully-connected layers (corresponding brown curves in Fig.III.13); **dropout**: 50% dropout to fully-connected layers plus data augmentation (corresponding green curves in Fig.III.13);

skull images are misclassified as 'heart' or 'abdomen'. The class abdomen and other achieve an error of approx. 0.18 and 0.10, respectively. Note from Fig.III.14a that the overall classification error is reduced by about 0.08 after applying 50% dropout to its fully connected layers and data augmentation to the training set. And also, there is a substantial improvement for the class heart which achieves an accuracy of about 0.70 (11% improvement over the standard DCNN). The confusion matrix in Fig.III.14d also reflects the improvement that the true predictions of 'heart' increases from 529 to 638 and the number of misclassified heart images (192, misclassified as 'abdomen') is reduced by 15% compared to the 'standard' DCNN (328, misclassified as 'abdomen').

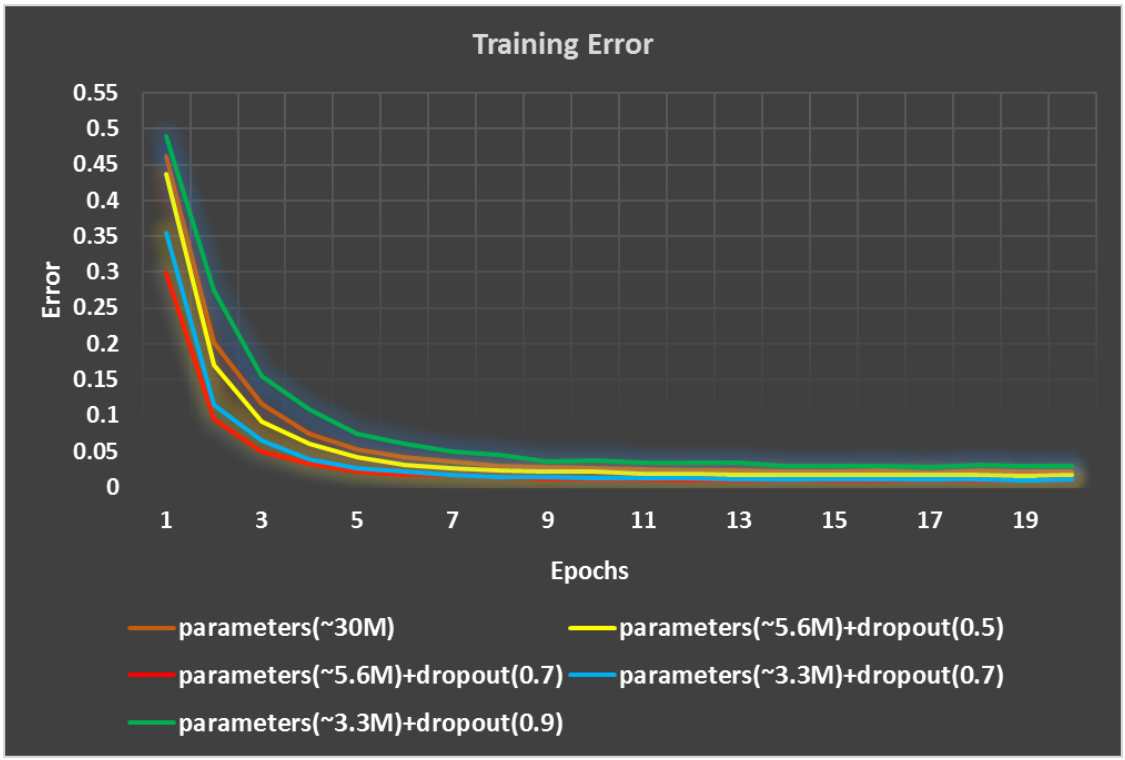


(a)

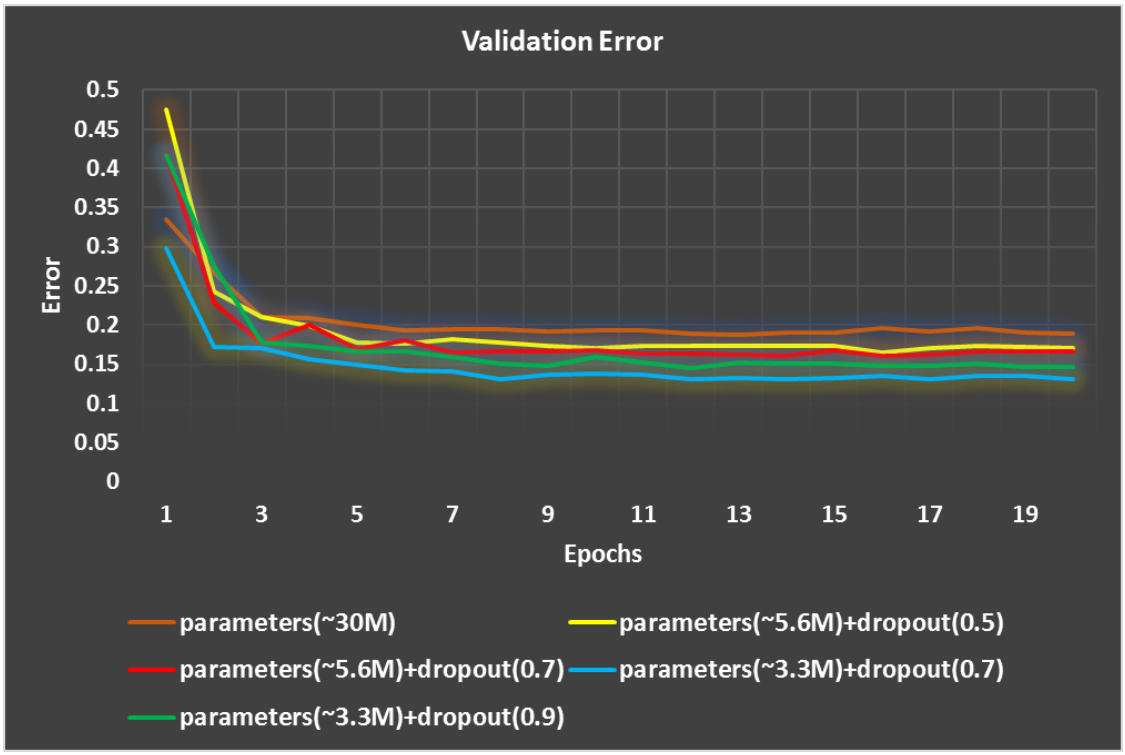


(b)

Figure III.15: Effect of reducing network size and batch normalization (BN). BN was applied to every single convolution layer and FC layer. **(a)** Training Objective (Loss), **(b)** Validation Objective.



(a)

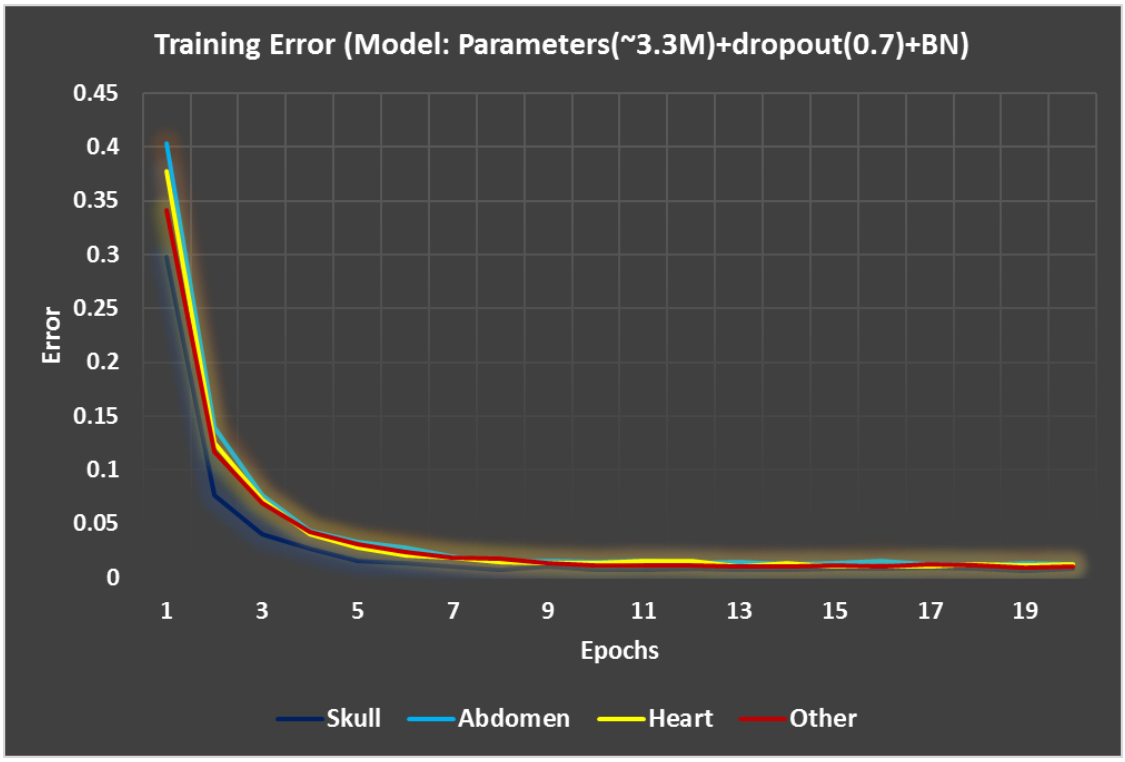


(b)

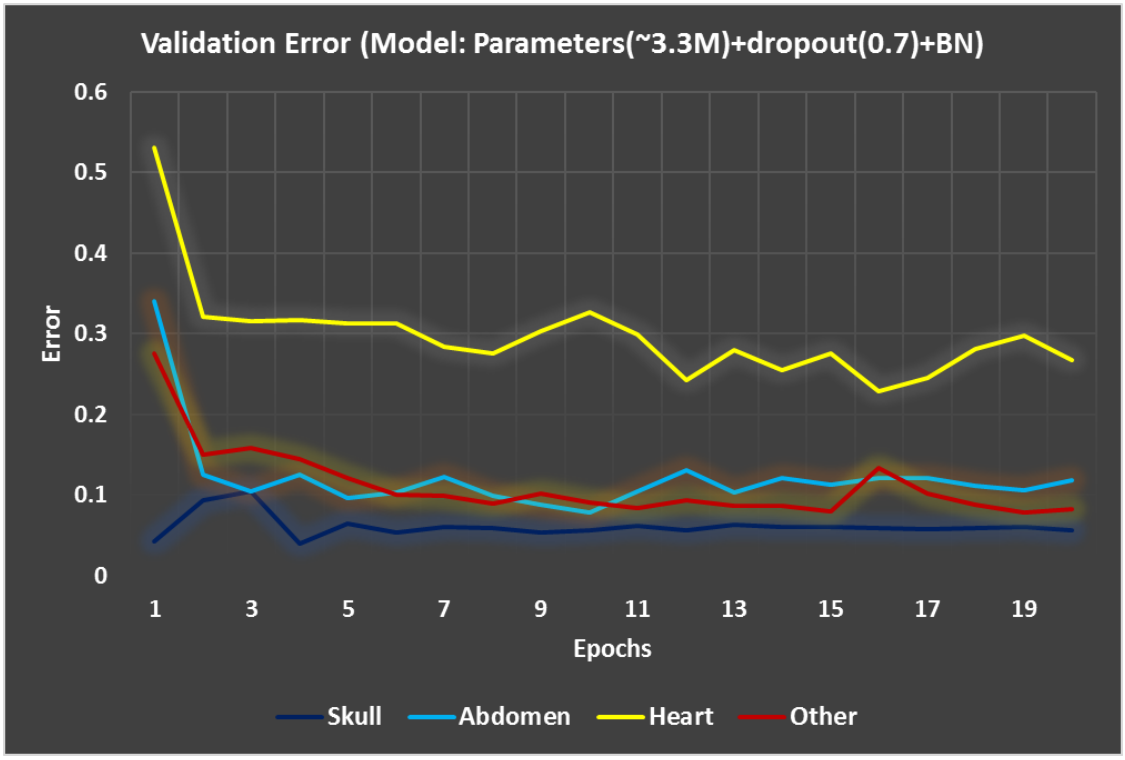
Figure III.16: Effect of reducing network size and batch normalization (BN). BN was applied to every single convolution layer and FC layer. **(a)** Training Error, **(b)** Validation Error.

Although generalization is improved by applying data augmentation and strong dropout, the model is still severely overfitted as the training error converges to 0 but validation error is far less closed to 0. The author argues this is because the limited training data (10820 images) is not sufficient to fit such a large model which has about 60 million trainable parameters. In addition, the author looked into the representations output from intermediate layers and came up with statistics of the distribution of features from which it found that the distribution varies significantly and the feature values grow exponentially when going deeper in the network. Therefore, different approaches of reducing the network size (described above) have been experimented extensively and batch normalization was applied to eliminate the internal co-variate shift, in other words, to ensure the intermediate feature distributions has approximately same variance. The models with reduced size and batch normalization are not overfitted anymore as the training loss (in Fig.III.15a) and error (in Fig.III.16a) converges to small values instead of 0 and the validation loss (in Fig.III.15b) no longer has increasing trend. And it should be noted that generally the batch normalized models with reduced size converge to a better minima with smaller validation loss (around 0.5) compared to the best model (around 1) in Fig.III.15b. Because of batch normalization, the training is significantly faster as can be seen (in Fig.III.15a and III.16a) by the fact that the training converges at after 10 epochs. However, the best model in Fig.III.12a and III.13a (as shown in green curves) takes about 20 epochs to converge. It can be found in Fig.III.15b and III.16b that smaller networks tend to achieve good classification performance i.e. smaller validation loss and error. The best validation loss and error are about 0.5 and 0.13 respectively which is achieved by the smallest model (i.e.3.3M parameters) with dropout rate of 0.7, as shown by cyan curve in Fig.III.15b and III.16b. It should also be noted that too much dropout i.e. 0.9 results in slightly poorer validation results as indicated in green curves.

The error of the best model is broken down into each individual class as illustrated in Fig.III.17b from which it can be found that the validation error is actually dominated by the class 'heart'. The model is significantly overfitted in learning



(a)



(b)

Figure III.17: Error break down into classes.

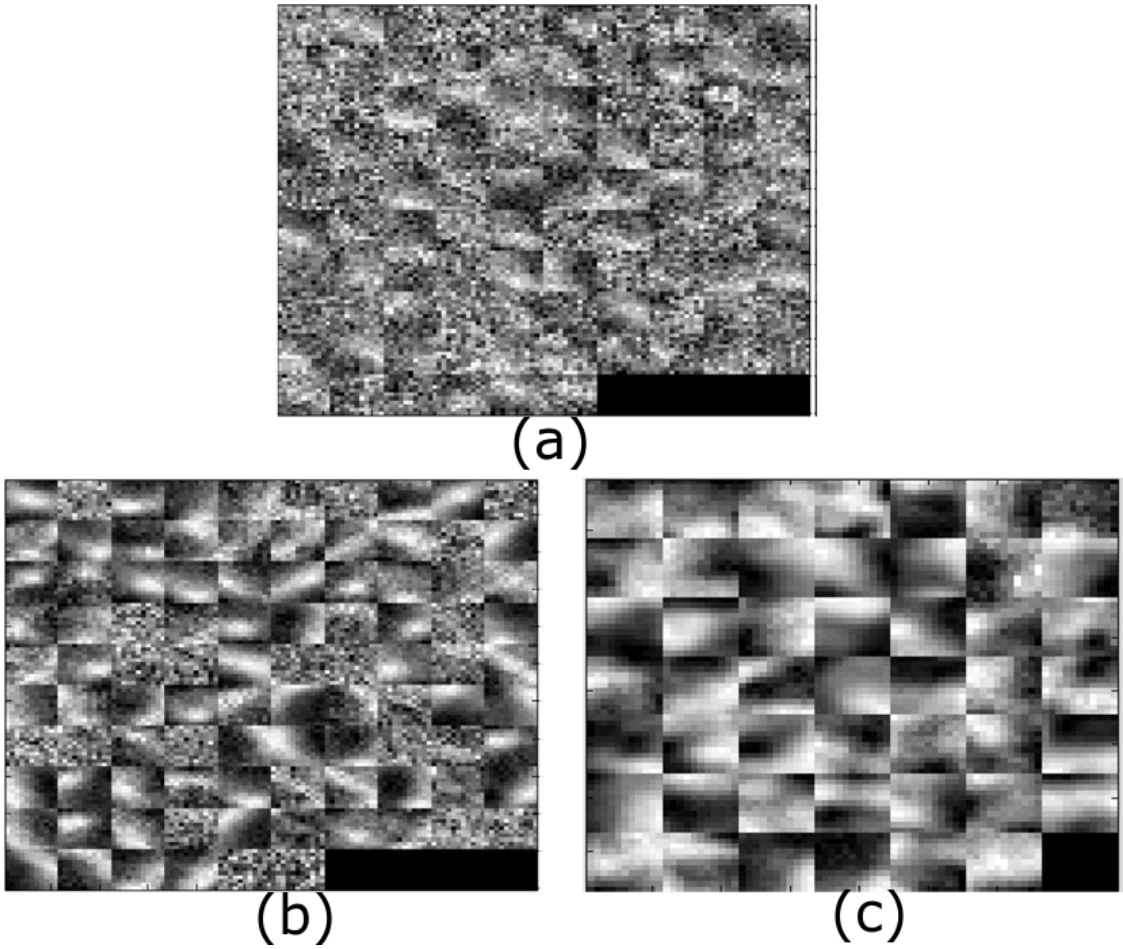


Figure III.18: Convolutional Kernels learned by the first layer: (a) without any regularization (b) dropout+ data augmentation (c) small network (half the number of trainable parameters) + strong regularization (data augmentation, dropout, batch normalization.

the classification of heart as can be seen (in green curves) the error is almost 0 but the validation error is around 0.28 at the end of training. And the validation error is fluctuated approaching the end of training which also indicates the poor generalization of the learning to the 'heart' class. In contrast, learning of the class 'skull' generalizes well, which is the best classified class with a validation error of 0.05 at end. And its validation error is fairly stable throughout the training, as shown in the dotted dark blue curve. The author argues that skull images are relatively easy to learn because skull is a rigid structure and is the most invariant class (low intra-class variations) in terms of appearance. Whereas, heart images are difficult to learn because the heart is deformable and can appear significantly different in

non-standardized planes (see next chapter) which makes it the least invariant class (high intra-class variations) of the 3 considered (and excluding "other").

Figure.III.18 illustrates the convolutional kernels learned by the first layer of DCNNs. It should be noted in Fig.III.18 (a) that the low level filters are extremely noisy without data augmentation and regularization (i.e.dropout and batch normalization). This also agrees with the results discussed above that the DCNNs are substantially overfitted without any regularization. Fewer filters are noisy after applying augmentation to the training data and dropout to the fully connected layers, as shown in Fig.III.18(b). By reducing the network size (half the number of parameters) and applying strong regularization (i.e. augmentation+dropout+batch normalization), the DCNNs are well fitted, as can be seen in Fig.III.18 (c) that the low-level patterns are clear and smooth without any noise.

Transfer Learning Models (TCNNs)

The results of transfer learning experiments are summarized in Fig.III.19. It should be noted that transfer without fine-tuning gives a total overall classification performance reduction of about 8% across layers one to seven. Inclusion of layers one and two results in moderate decline of the accuracy, indicating that, the first two layers are generally applicable. Inclusion of layers three to five showed a slightly more drop in accuracy, and inclusion of layers five to seven showed a significant drop in accuracy across all classes. This drop might be attributed to one of two effects: firstly, high level features become increasingly specific; secondly, there is a lost co-adaptation of neurons between successive layers. The original network contained fragile co-adapted features on successive layers, that is, features that interact with each other in a complex or fragile way such that this co-adaptation could not be relearned by the upper layers alone. Overall, it was observed that transfer with fine-tuning (explained in Section III.4) results in networks that generalize better than those trained directly on the ultrasound dataset (direct learning). This improvement seems to not depend on how many layers of the base network are

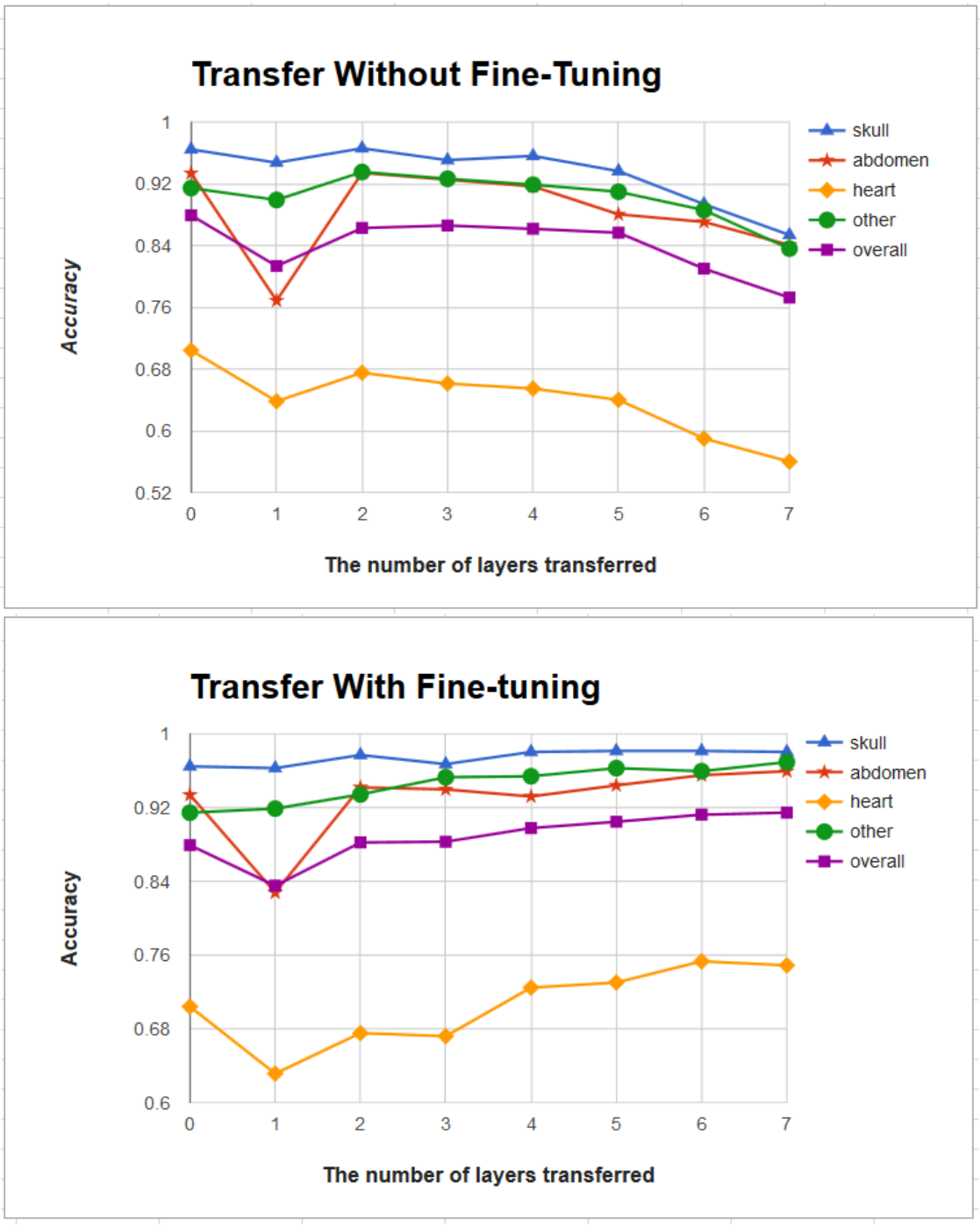


Figure III.19: Classification performance of transfer learning models.

kept to initialize the target network: keeping anywhere from one to seven layers produces improved performance, with slightly better performance as more layers are kept. As can been seen, the model learned from scratch (i.e. the number of layers transferred = 0) achieves an accuracy of about 0.88, the transfer learning with all layers transferred (i.e. the number of layers transferred = 7) achieves a better accuracy of about 0.92 that gives an overall accuracy increase of about 4%. Similar to DCNNs, the class 'heart' remains the worst classified in the transfer learning models regardless of fine-tuning or not, as can be seen its best accuracy is about 0.76 which is significantly lower than the best classified class 'skull' achieving an accuracy of about 0.97 with fine-tuning. The above experiments all indicate that the deep representations learned for the heart might not be sufficiently discriminate so that results in the poor classification performance. Section III.6.3 will look into the models learned above using different visualization technique (described in Section III.5) to gain some insight into the limitation of the models.

III.6.3 Visualization

t-SNE- Visualizing High Dimensional Features

The t-SNE technique (described in Section III.5.3) was used to reduce dimensions of raw datasets, and features extracted from fully-connected layers of the best TCNNs model (discussed in Section III.6.2), so as to plot data in a 2D space. To illustrate this, Fig.III.21 shows the visualization of (a) raw training dataset; (b) raw validation dataset; (c) high level feature vectors extracted from training dataset; (d) high level feature vectors extracted from validation dataset. As can be seen from Fig.III.21 (a) and (b), distribution of the raw data points shows a high intra-class variation that the points with same colour spread all over the space and the intra-class clusters formed are distant from each other. While, the inter-class variation is low that the points with different colours are all mixed together. The clusters on the training dataset are formed perfectly with clear boundaries separating each other, as shown in Fig.III.21 (c), which explains why the training error is almost touching down to

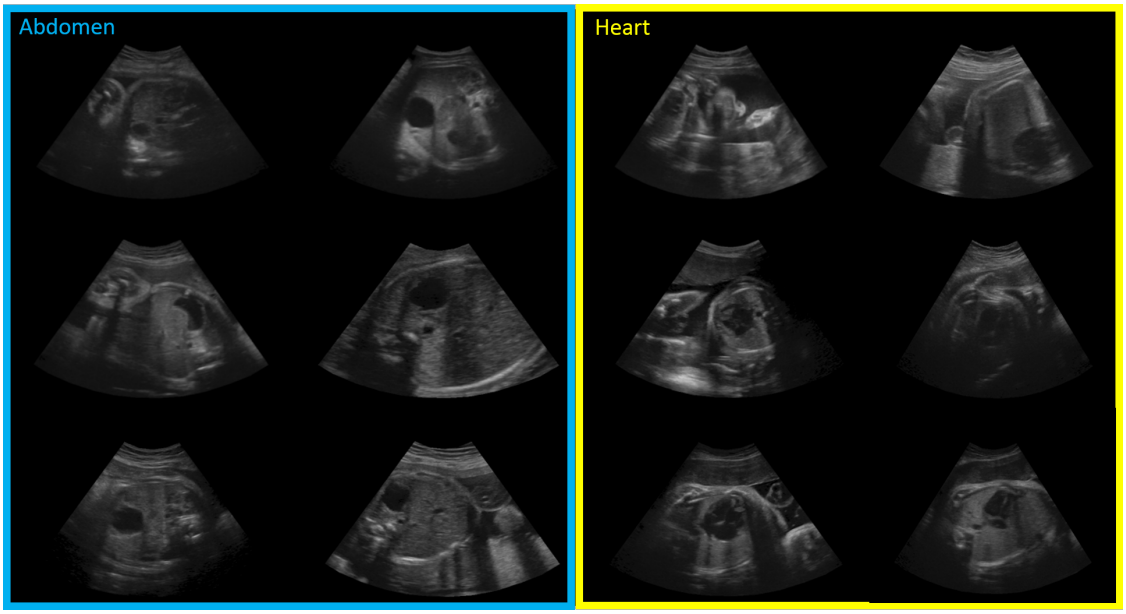


Figure III.20: Misclassified abdomen and heart images. Abdomen samples misclassified as heart (enclosed by cyan box); Heart samples misclassified as abdomen (enclosed by yellow box).

zero. While, the decision boundaries on the validation dataset are quite ambiguous between some classes, such as the 'heart' and 'abdomen', as illustrated in Fig.III.21 (d). From the clustering in Fig.III.21 (d), note that the yellow cluster is particularly close to the cyan cluster showing a low inter-class difference between the class 'heart' and 'abdomen'. One can hypothesize this is because heart frames often share similar appearance to abdomen frames, as illustrated in Fig.III.20, because (1) heart resides in the thorax that is anatomically closed to the abdomen; (2) hearts captured in non-standard planes (takes up 62.5% of all heart images) appear like hollow structures and cardiacs structures are barely visible. Anatomical structures in abdomen, such as the stomach bubble, also appears like a hollow structure. This leads to a high chance that a heart image is mis-classified as an abdomen image. In addition, fetal heart images are highly variant in terms of appearance as the fetus can be in various poses, which is observed from the high intra-class variation of the yellow datapoints (yellow clusters spreading all over the space).

To examine closely the mis-classified images, the validation images are embedded in the clustering map, as illustrated in Fig.III.22, in which it can be found the

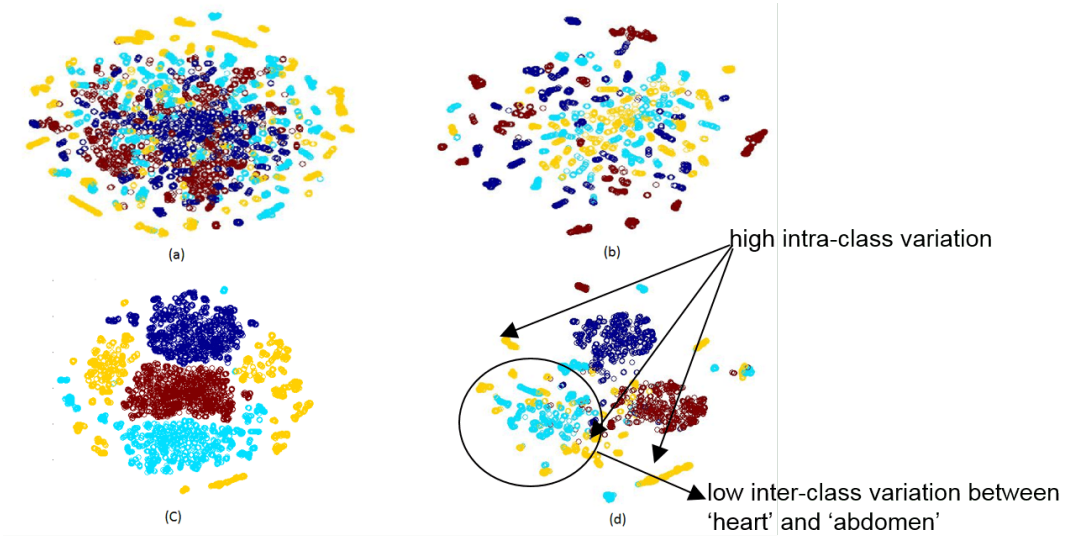


Figure III.21: t-SNE visualisation. (a) Raw training data. (b) Raw testing data. (c) Extracted features of the 7th fully connected layer from training dataset. (d) Extracted features of the 7th fully connected layer from validation dataset. Blue: Skull, Cyan: Abdomen, Yellow: Heart, Red: Other.

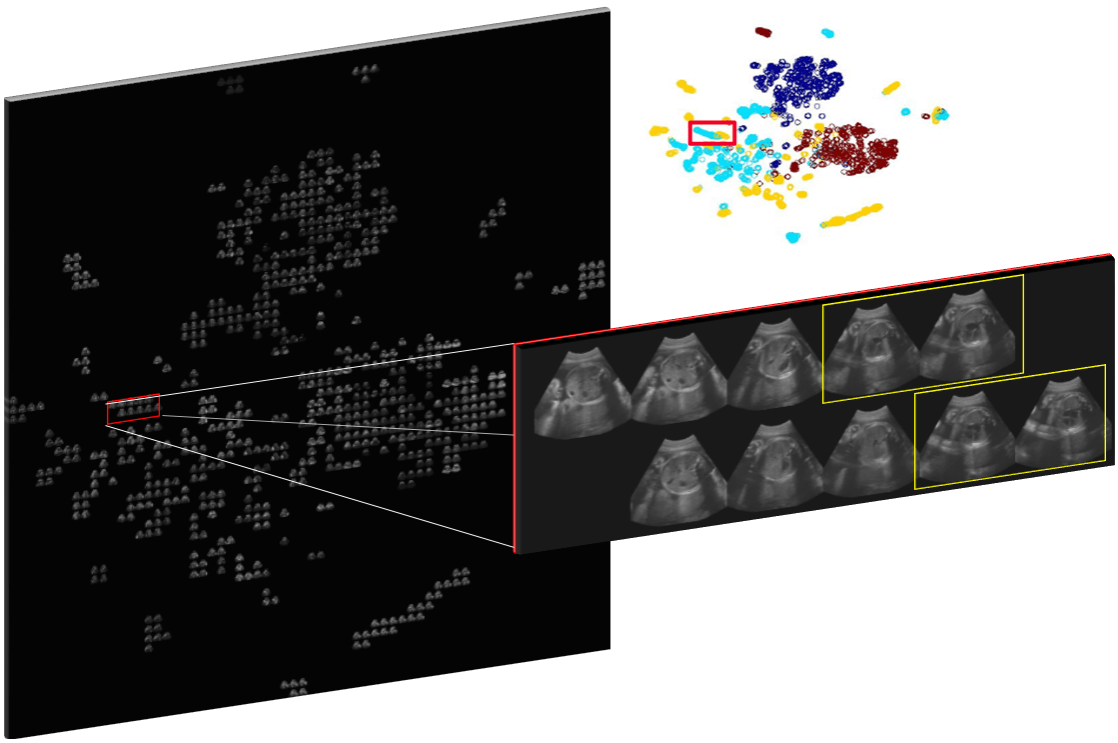


Figure III.22: Raw validation images embedding.

heart images (enclosed by yellow box) are mistakenly recognized as abdomen images by zooming into the region (enclosed by red box) in the embedded map. It can be seen that these images have similar appearance surrounding the thorax and abdomen area that a cross-section of a limb appears to the upper left. And abdomen and thorax appearing in the images are about same size. Therefore, one can explain the misclassification that global feature descriptors (FC layer features) are not good at preserving the local features (small anatomical structures in the thorax and abdomen), such as the heart and stomach bubble, and some of the encoded features are not relevant to the target structures (this is also reflected by class-specific saliency maps, see next section).

Class-Specific Saliency Visualization

A saliency map was generated per prediction, as illustrated in Fig.III.23 showing the class-specific evidence of prediction. The hot-spots highlight those pixels have high influence on predicting the target label and the brighter the hotspots are, the higher the influence is. For instance, it can be seen from the skull samples that the hot-spots are aligned with skull indicating that the skull bone is the most discriminative feature of the skull images. In contrast, for the abdomen and heart saliency, it can be seen that hot spots are scattered and distributed everywhere so that it is difficult to spot which structures are the most discriminative. Particularly, it should be noted that the hot-spots mostly aligned with the structures that are irrelevant to heart, such as cross-section of a limb (first column, third row) and ribs (first column, last row). This also agree with the finding in t-SNE visualization that the CNNs adopting fully connected layers tend to learn global feature descriptors ignoring anatomical features that relevant to the target structure i.e. heart. The saliency map of the heart image (second column, third row) is slightly more discriminative than others, on which finds the hot-spots surround and highlight the pulmonary artery and aorta area. In the abdomen saliency examples, it can also be noted that

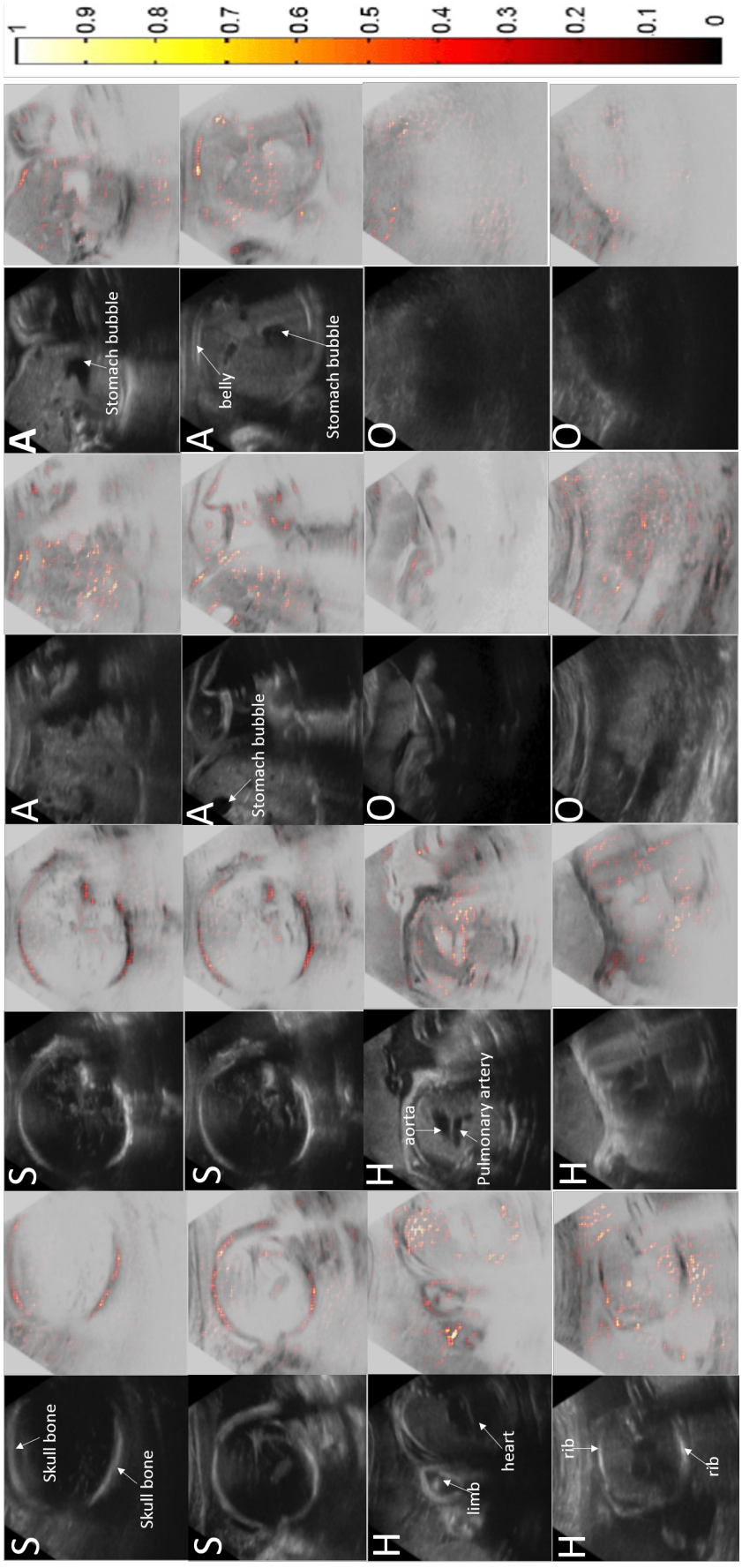


Figure III.23: Class-specific Saliency Maps. Four columns from left to right. In each column: original images (Left); corresponding saliency maps on which hot spots highlight the class-specific pixels (Right). **S: Skull**, **A: Abdomen**, **H: Heart**, **O: Other**.

the hot-spots highlight the area surrounding the belly and the inside anatomical structures such as stomach bubble.

Deconvolution

Figure.III.24 illustrates feature visualizations of layer 5 from the best DCNN model, which shows the single strongest activation for a given feature map. Projecting each feature channel separately down to the pixel space reveals the different structures that excite the feature maps, hence showing its invariance to input deformations. Alongside some of the patterns learned in layer 5 are shown together with their corresponding images. It should be noted from the visualizations that each feature channel focuses on the discriminant structure. For instance, the patterns show significantly variation, but is very class-specific, such as the stomach bubbles with different shape and the part of fetal skull.

III.7 Discussion and Conclusion

To summarize, the author has experimented thoroughly with classification of fetal ultrasound images using different CNNs architectures: DCNNs (learning from scratch) and TCNNs (transfer learning). The results suggest learning the deep models can be regularized by augmenting the training data and dropping out the dense layers. In addition, the standard architecture, such as AlexNet, designed for large scale image classification (1000 categories) do not fit our small classification tasks (4 categories) well with much less training images available and much fewer classes. It is of great importance to reduce the network size i.e. trainable parameters to avoid overfitting the CNNs when learning from scratch. Batch normalization was adopted in every trainable layer in our CNN design to accelerate convergence of training. The experiments suggest that batch normalization indeed results in a faster convergence of training but there is no indication that batch normalization can improve the validation accuracy. In terms of transfer learning, the results suggest

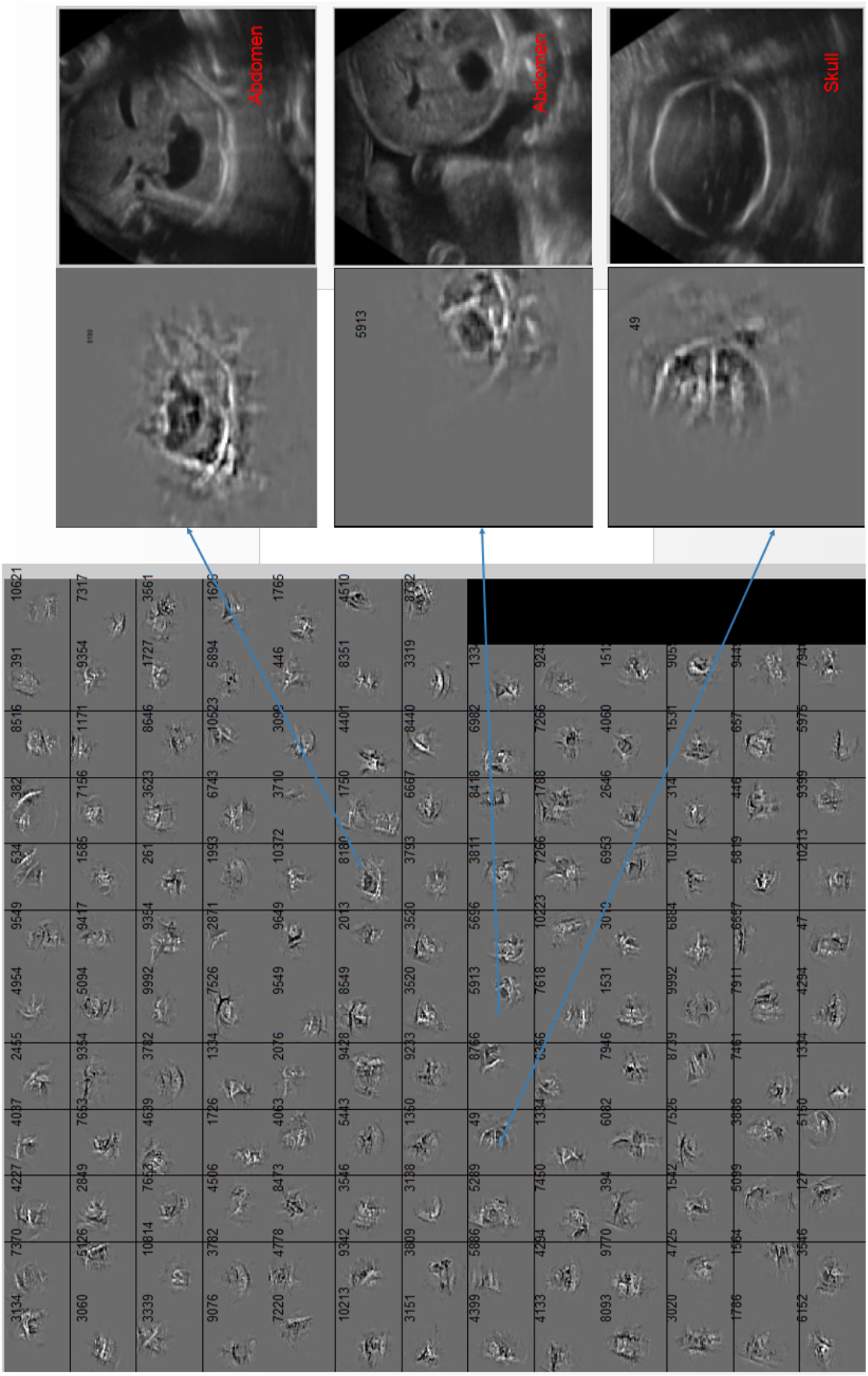


Figure III.24: Visualization of mid-high level features: showing the toppest activation of feature maps in the 5th layer (layer before fully-connection) across the validation data, projected down to pixel space using deconvolution approach.

that accuracy can be enhanced via initializing networks with features learned from natural images which is also helpful to avoid overfitting.

In this chapter, visualization has also been studied to qualitatively evaluate what is actually learned in a deep CNN. t-SNE was employed to visualize the clusters formed by the original dataset and the features extracted from dense layers. This helps to understand the distribution of the original data that suggests the inter-class variation is low (especially between the 'abdomen' and 'heart'), and the intra-class variation is high (particularly the 'heart'). This explains the poor classification performance of the heart images which are very often be classified as abdomen images. The author argues that this is because our CNN design adopts the high level dense layer which tends to learn global feature descriptors and discard the fine-grained local features. This is actually proved by our class-specific saliency visualization in which it can be found that the pixels around the abdomen area actually have higher influence than the pixels around the heart when predicting a heart image. To address the limitations, a fully-convolutional architecture is introduced in the next chapter which preserves the spatial correspondence of local features and substantially reduces the number of trainable parameters. To learn a more discriminative feature representation of heart, the next chapter explores the importance of incorporating temporal context from consecutive frames.

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [9]

IV

Weakly-Supervised Detection, Localization and Characterization of Fetal Heartbeat

Contents

IV.1 Introduction **90**
IV.2 Dataset **91**
IV.2.1 Sequence Annotation 92
IV.2.2 Preprocessing 95
IV.3 Learning to Extract Motion from Videos in CNN . . . **95**
IV.3.1 CNN Architecture 95
IV.3.2 Multi-scale Learning 99
IV.4 Motion-Appearance Fusion: Two-Stream CNN **100**
IV.5 Experiments and Results **102**
IV.5.1 Implementation Details 103
IV.5.2 Classification Results 104
IV.5.3 Localization Results 106
IV.5.4 Characterization of Heartbeats 111
IV.6 Discussion and Conclusion **112**

In this chapter, the author addresses the challenge of fetal heart recognition by incorporating a motion cue into the convolutional feature representation. Automated detection and localization of a beating fetal heart, in free-hand ultrasound sweeps

is a very challenging task due to high variation in heart appearance, scale and position (because of heart deformation, scanning orientations and artefacts). This chapter presents a two-stream Convolutional Network (CNN) -a temporal sequence learning model- that recognizes heart frames and localizes the fetal heart using only weak supervision. An early version of the work presented in this chapter has been reported in:

Y Gao, JA Noble. "Detection and Characterization of the Fetal Heartbeat in Free-hand Ultrasound Sweeps with Weakly-supervised Two streams Convolutional Networks". International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI), 2017.

IV.1 Introduction

Automated detection and characterization of the fetal heart is of great help in diagnosis of Congenital Heart Disease (CHD). However, this task is very challenging in clinical free-hand ultrasound (US) videos for a number of reasons [79]. The appearance of a fetal heart is highly varied throughout the cardiac cycle, and depends on the pose of the fetus relative to the transducer. Fetal hearts also appear indistinct sometimes, due to variations in imaging contrast, as well as the presence of artefacts (e.g. acoustic shadow). To support checking that a fetal heart is beating and the assessment of fetal cardiac abnormalities, there is a clinical need to develop automated recognition tools, which can not only identify and index the different viewing planes of the fetal heart, but can also localize and characterize the fetal heart from the planes in quick succession.

An end-to-end trainable two-stream Convolutional Network, inspired by [63] and [103], was proposed for fetal heart recognition and characterization in clinical free-hand fetal US videos. The network is fully convolutional, consisting of a temporal stream that extracts motion features, and a spatial stream that extracts appearance features. The two streams are fused to learn a spatio-temporal representation. It

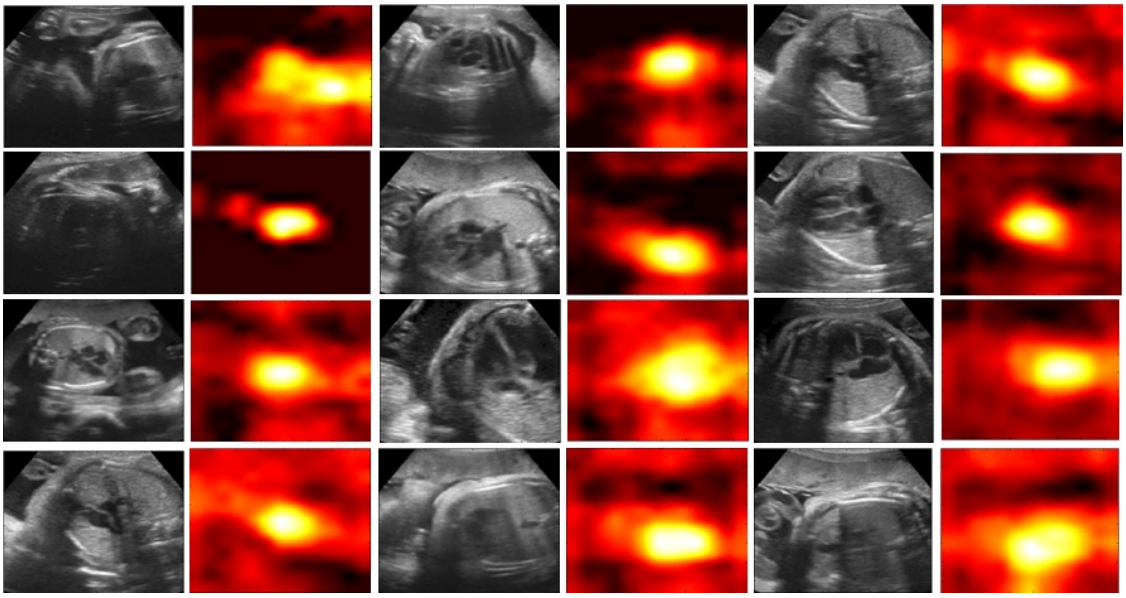


Figure IV.1: Localization confidence maps for fetal heart detection of our weakly-supervised two-stream ConvNet. Note our model is robust to variations in fetal heart appearance, scale, position and image contrast.

demonstrates that the framework leads to (1) substantial improvement in terms of correct identification of fetal heart frames compared to spatial CNNs (as introduced in Chapter III); and (2) accurate localization (predicts approximate location, as illustrated in Fig.IV.1) of fetal hearts with only image-level supervision; (3) generate optical flow maps describing motion locally and globally.

IV.2 Dataset

As introduced in Section III.2 Chapter III, 323 fetal ultrasound sweeps were acquired from participants who were taking part in the INTERGROWTH-21st project. In this chapter, another 133 such sweeps were managed to be acquired following the same protocol. Therefore, there are 456 videos in total used in this chapter.

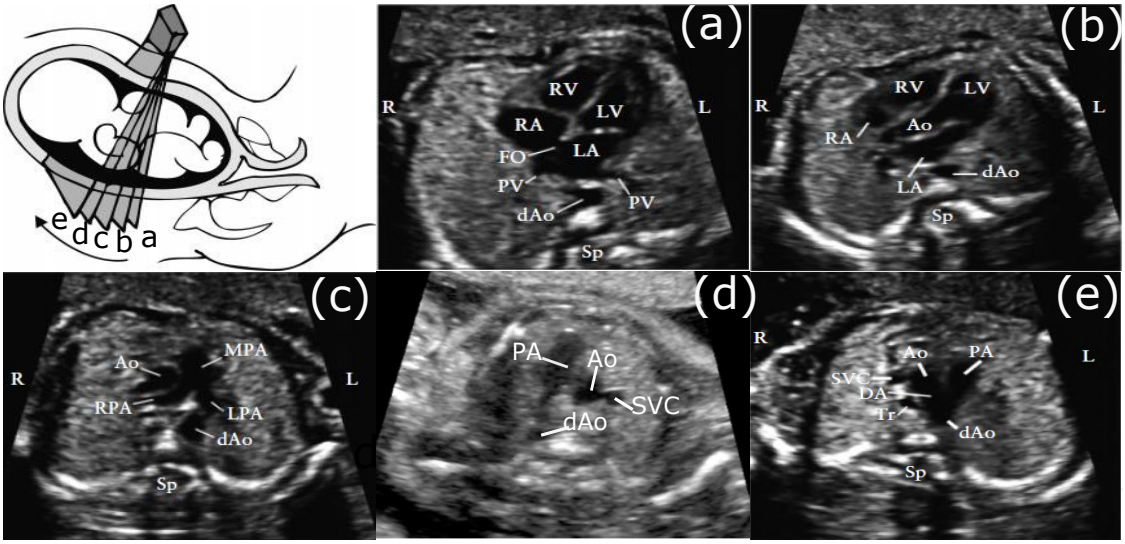


Figure IV.2: Acquisition of fetal cardiac planes (adapted from [104]). The five axial views for optimal fetal heart screening are illustrated: **(a) Four-chamber** view of the fetal heart, showing the right and left ventricles (RV, LV) and atria (RA, LA), foramen ovale (FO) and pulmonary veins (PV) to the right and left of the descending aorta (dAo). **(b) Left ventricular outflow tract view**, showing the aortic root (Ao), LV, RV, LA, and RA and a cross-section of the dAo. **(c) Right ventricular outflow tract** view showing the main pulmonary artery (MPA) and the bifurcation into the right (RPA) and left (LPA) pulmonary arteries and cross-sections of the ascending aorta (Ao) and dAo aorta. **(d) Three-vessel view.** This view best demonstrates the relationship between the pulmonary artery (PA), aorta (Ao) and superior vena cava (SVC) in the upper mediastinum. The pulmonary artery, to the left, is the largest of the three and the most anterior, whereas the SVC is the smallest and most posterior. **(e)Three vessels and trachea view** showing the superior vena cava (SVC), pulmonary artery (PA), ductus arteriosus (DA), transverse aortic arch (from proximal Ao to dAo) and trachea (Tr). IVC, inferior vena cava; **L, Abdominal left; R, Abdominal right.**

IV.2.1 Sequence Annotation

Instead of annotating images frame-by-frame, video clips containing one of the three fetal structures i.e. fetal abdomen, heart and skull were annotated. Each of the annotated clips contains only one of the structures of interest. Video clips do not fall into the three classes are labeled as background frames. In terms of abdomen and skull, the annotation follows the criteria defined in Chapter III Section III.2.2.

To annotate the fetal heart, it is necessary to understand how the standard cardiac planes are actually acquired, as illustrated in Fig.IV.2. The figure shows the ideal heart screening i.e. fetal echocardiography. When sweeping the probe

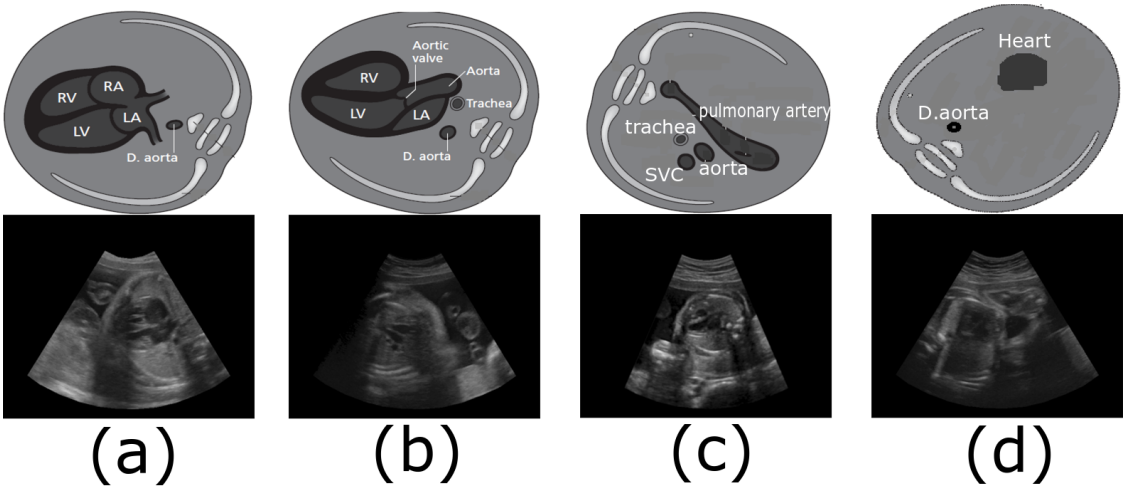


Figure IV.3: Annotation criteria for fetal heart. (a) Four chamber view. (b) Left ventricular Outlow Tract (LVOT). (c) Three Vessel View. (d) Non-standard View. The only necessary criterion is presence of a heart with or without the other standard key landmarks. The criteria are set loosely and independent of the presence of key anatomical structures.

towards the fetal heart, it starts from picking a four-chamber view (Fig.IV.2(a)) then sweeping further upwards, LVOT (Fig.IV.2(b)) and RVOT (Fig.IV.2(c)) views are acquired successively that the aorta and pulmonary artery arise from the left and right ventricle respectively. Sweeping further upwards, the three vessels start appearing (Fig.IV.2(d)) from left to right, the vessels are the pulmonary artery, the aorta and the superior vena cava, respectively. Their relative diameters decrease from left to right, with the pulmonary artery being larger than the aorta, and the aorta larger than the superior vena cava. The top-most view is the three vessel and trachea (Fig.IV.2(e)) that the three vessels are merged and trachea appearing relatively more posterior than the superior vena cava.

In contrast to fetal echocardiography, some of the key landmarks are not necessarily visible in our heart video frames due to the simple acquisition protocol. The standard heart frames in our data are dominated by the four chamber view (Fig.IV.3(a)), the LVOT (Fig.IV.3(b)) and the three vessel view (Fig.IV.3(c)). It can be seen the image quality of our data is not as good as the echocardiology examples so that it is unlikely to capture all the key landmarks defined in standard planes (in Fig.IV.2). Therefore, a loose criteria is used to annotate fetal heart planes as

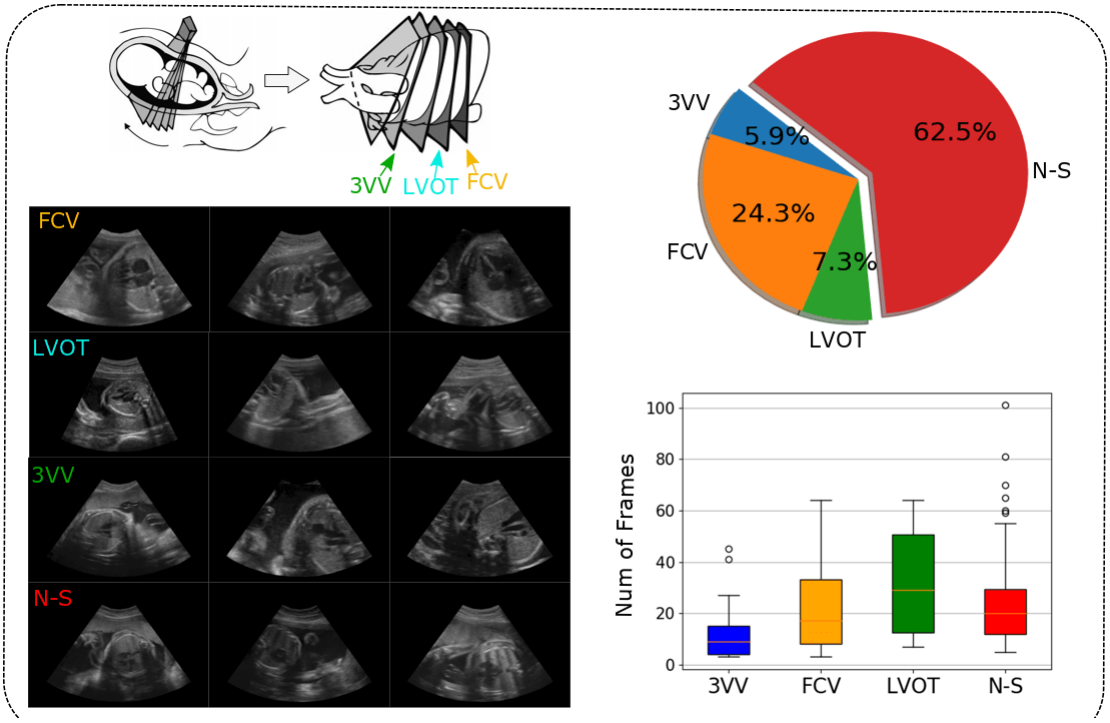


Figure IV.4: Intraclass Statistics e.g. Heart. FCV: Four Chamber View, 3VV: Three Vessel View, LVOT: Left Ventricular Outflow Tract, N-S: Non-Standard Views.

illustrated in Fig.IV.3 that the four chamber is labelled as long as the two ventricles and two atria are clearly visible on the plane; the LVOT is labeled as long as the aorta is clearly visible arising from the left ventricle on the plane; the three vessel is labelled as long as the three vessels i.e. pulmonary artery, aorta, and SVC are clearly visible on the plane; otherwise the heart plane is labeled as non-standard plane.

The statistics of heart planes in our data is summarized in Fig.IV.4 showing that there is significantly more non-standard planes than standard ones. It should be noted that fetal hearts appear very differently on the non-standard planes, which also explains why the intra-class variation is high for the class 'heart' (as discussed in the Chapter III Section III.6.3). The standard planes, including the Four Chamber View (FCV), Left Ventricular Outflow Tract (LVOT), 3 Vessels View (3VV), take account for only about 38% of all heart video clips. In terms of video length, the LVOT view is longer than the other views and the 3VV is the shortest. The length of non-standard view is highly varied ranging from a couple of frames to more than a hundred frames as shown in Figure IV.4.

IV.2.2 Preprocessing

The videos clips were sub-sampled to create sequences of three consecutive frames with a step size determined by the length of each video. The sampling size varies adaptively with the length of input videos, the longer the video the larger the sampling step. This is to ensure none of the samples are too similar in terms of appearance. 80% of the videos were randomly chosen as training data. The rest were split in validation (10%) and test (10%). All video frames had an original size of 240×320 pixels and were cropped at center to produce 152×152 square patches. To increase robustness to changes in brightness and contrast, each video frame was preprocessed by first subtracting a Gaussian smoothed version of the image and then applying standard normalization with a sliding disk mask (i.e. subtract mean intensity of each frame and divide it by standard deviations obtained locally).

IV.3 Learning to Extract Motion from Videos in CNN

IV.3.1 CNN Architecture

To preserve local correspondence of features, a fully-convolutional CNN (originally introduced in [103]) is adopted to learn a motion detector, as shown in Fig.IV.5. The network consists of four convolutional layers, the Conv1 and Conv2 layers are responsible for learning the motion feature representation and the Conv3 and Conv4 learn to map the representations to an optical flow map i.e. 2D flow vector for every pixel. Convolution in the Conv1 and Conv2 layer is padded with zeros to keep the spatial resolution of feature maps unchanged. The downsampling is achieved with the max-pooling layers after Conv1 and Conv2 respectively. The output is thus quarter the resolution of the input. Each location in the output flow map corresponds to a spatially-constrained receptive field in the input, and thus each

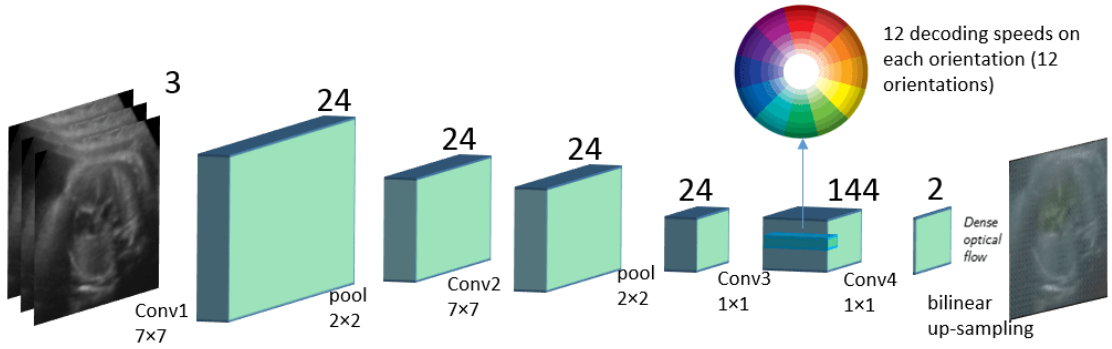


Figure IV.5: CNN motion detector. The CNN takes raw pixels of video frames as input, and outputs a dense optical flow at quarter its resolution. The network consists of 2 convolutional layers, 2 max-pooling layer, and 2 pixelwise weights (1×1 convolutions).

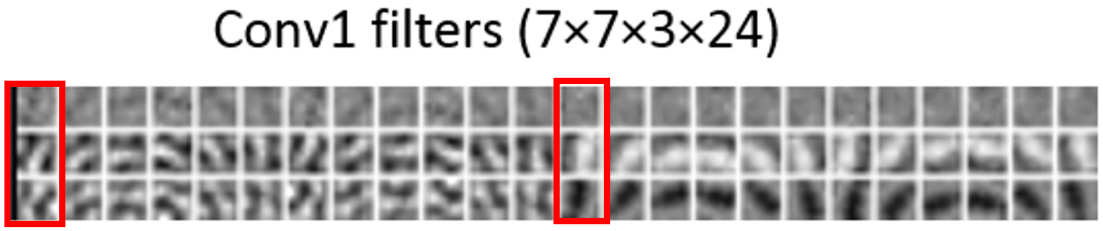


Figure IV.6: Conv1 filter patterns.

pixel of a training sequence can thus be seen as a unique training point. Finally, bilinear upsampling is employed to obtain flow fields at the input resolution.

In the context, x^l_{ijc} denotes the feature value at coordinates (i,j,c) in the 3D tensor obtained by evaluating the l^{th} layer, where indices i and j refer to spatial dimensions, c to feature channels. A colon (:) refer to all elements along a dimension. $*$ refers to convolutions in 3D.

Motion detection

Conv1 learns to detect the direction and magnitude of motion, and is given by:

$$x^{l+1}_{::c} = x^l_{:::} * W^1_c + b^1_c \tag{IV.1}$$

The learned kernels are illustrated in Fig.IV.6 showing the oriented spatial-temporal Gabors tuned with different frequency. They respond to moving patterns at different orientation and speed. Actually, only two groups of spatial Gabor filters (as enclosed

by the red box in Fig.IV.6) were learnt. A rotation-invariance is enforced by rotating them to different orientations. And in our implementation, the number of orientations is explicitly defined to be 12. Thus, this in total results in 24 i.e. 2×12 feature channels in the Conv1 layer.

Invariance to local image phase

The estimated motion should be invariant to local image phase changes. Conventionally, the models in [105] and [106] employ quadrature pairs of odd-phase and even-phase filters that the sum of the squared output of a sine-phase filter i.e. odd filter, plus the squared output of a cosine-phase filter gives a measure of Gabor energy that is invariant to the phase of the signal. In our CNN model, this can be approximated by the point-wise squaring rectification [103] of the Conv1 output, as:

$$x^l_{ijc} = (x^l_{ijc})^2 \tag{IV.2}$$

Invariance to local image structure

The estimated motion should be independent from the dynamic texture in the images. In the case of fetal heart videos, there is sometimes the acoustic shadow appearing caused by the ultrasound beam passing through the fetal ribs. This acoustic shadows can affect the visibility of key cardiac structures and prediction of true flow specific to heart. To account for this, the squaring rectified responses are normalized by the sum over all orientations, as:

$$x^l_{ijc} = \frac{x^l_{ijc}}{\sum_{c'} x^l_{ijc'} + \epsilon} \tag{IV.3}$$

Another convolutional layer i.e. Conv2 with max-pooling is introduced to approximate the smooth regularization of variational models, such as [107] and [108], which penalizes high variations of flow (large gradients):

$$x^{l+1}_{::c} = x^l_{:::} * W^2_c + b^2_c \tag{IV.4}$$

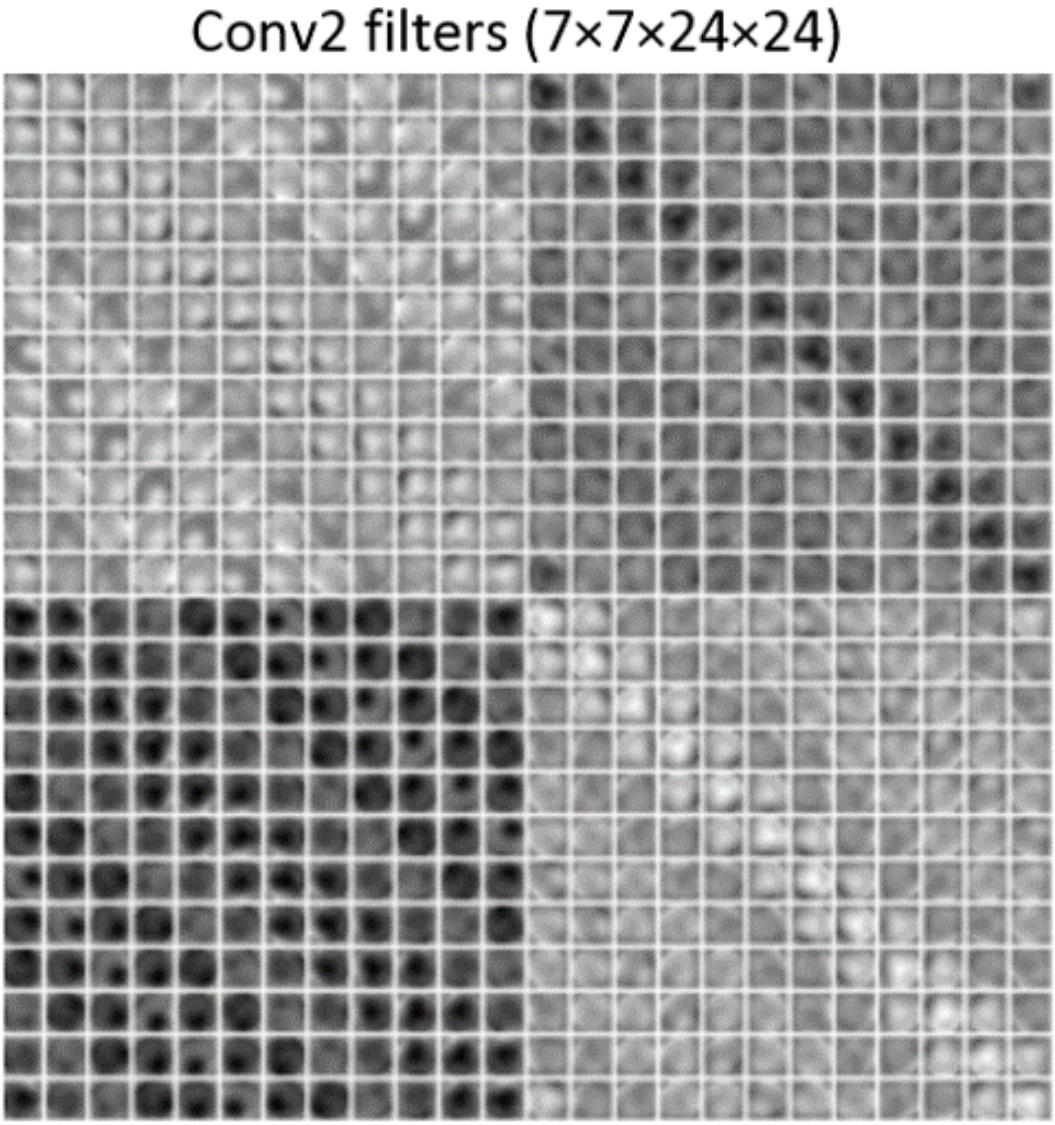


Figure IV.7: Conv2 filter patterns.

Figure IV.7 shows that the Conv2 layer learns Gaussian-like kernels, which are slightly non-isotropic, and modeling some cross-channel and center-surround interactions.

Decoding into flow vectors

Conv3 performs a pixel-wise convolution with 1×1 kernels to decode the motion representation into 144 dimensional scores (12 orientations with 12 speeds on

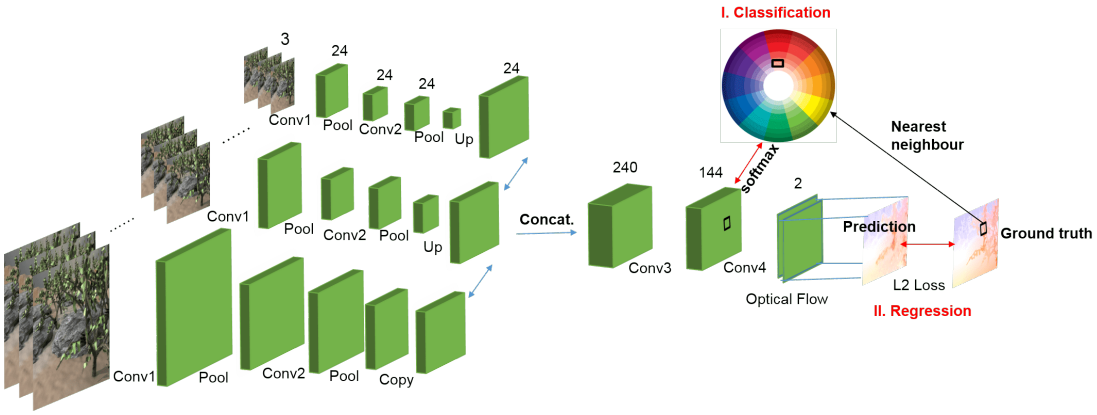


Figure IV.8: Multi-scale learning for motion detection. **Concat.** denotes stacks the feature maps at the same spatial locations.

each orientation):

$$x^{l+1}_{::c} = x^{l}_{:::} * W^3_c + b^3_c \tag{IV.5}$$

Conv4 linearly projects the motion scores to an optical flow map with 2 channels i.e. horizontal and vertical displacement:

$$x^{l+1}_{::c} = x^{l}_{:::} * W^4_c + b^4_c \tag{IV.6}$$

IV.3.2 Multi-scale Learning

As described above, equations IV.1-IV.6 form the entire network mapping pixels to an optical flow map. However, the large displacement may not be detected due to the small effective receptive field of the output units (shallow design with limited number of convolutional layers). This is managed with a multi-scale training scheme, as illustrated in Fig.IV.8 , without increasing the number of parameters to train. The network (Eq. IV.1-IV.4) is applied on multiple downsized versions of the input frames. The feature maps are brought back to the original resolution by bilinear upsampling and concatenated before feeding into the decoding stage (Eq.IV.5-IV.6).

The network was trained end-to-end with backpropagation. After Conv3, the motion is represented by the feature maps that each feature channel correspondence to a unique flow vector. The network was firstly trained for a pixel-wise classification

of flow, as depicted in Fig.IV.8. The classification ground-truth is generated from the ground-truth flow map (from the Middleburry [109] and MPI Sinitel [110] dataset) by finding its nearest neighbour (from the 144 flow vectors) at each pixel. Softmax is applied pixel-wise to the Conv3 output to generate a uni-modal distribution (i.e. assuming single motion) at each pixel, which is used to train classification with a cross-entropy loss. Then, the network is fine-tuned by learning a pixel-wise regression to the ground-truth flow vectors with an L2 loss.

IV.4 Motion-Appearance Fusion: Two-Stream CNN

The architecture of our two-stream ConvNet is summarized in Fig.IV.9. As introduced in the previous sessions, the motion stream implements a shallow Fully Convolutional Networks (FCN) that takes a preprocessed stack of three consecutive frames of a video as input, and extracts high dimensional motion representations, which are then projected as 2D optical flow maps (spatially 4× downsampled). **Conv1** in the motion stream performs as a motion detector that captures patterns moving at different speeds and orientations. A local phase-invariant response is approximated by element-wise squaring rectification and 2×2 max-pooling with stride of 2. Channel-wise L1-Norm is then applied to the output to account for intensity variance of patterns moving at different orientations. **Conv2** in the motion stream is to smooth the encoded motion locally. **Conv3** layer performs as a motion decoder using 1×1 kernels, which decodes features into 144 channels representing scores at different speeds and orientations. **Conv4** linearly projects the scores to an optical flow map. The appearance stream adopts the configuration of VGG very deep ConvNets (11 weights layer) [56]. All convolutional layers in the appearance ConvNet consists of 3×3 kernels, batch normalization (bnorm) [46] and Rectified Linear Units (ReLUs) non-linearity [8]. The size of feature maps are spatially matched throughout the feed-forward processing between the motion and appearance stream. Instead of add fusion, the two streams are fused by stacking their feature maps (normalized to [0,1]) together. One reason for the concatenation

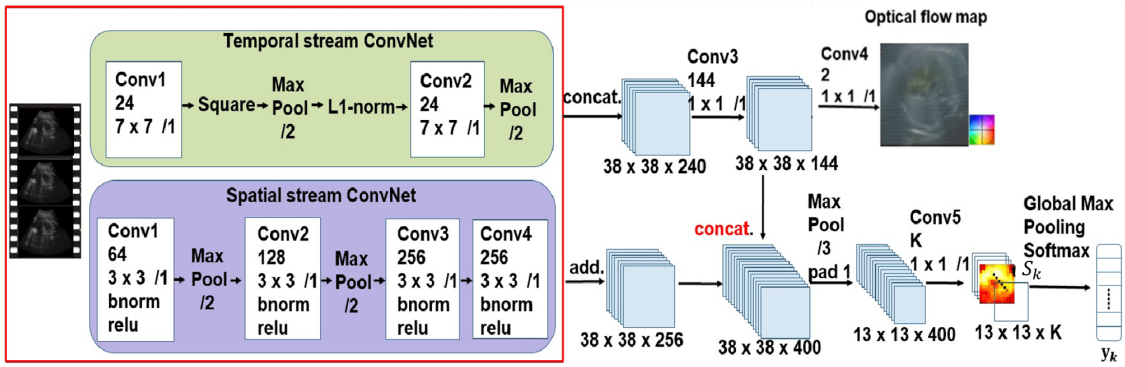


Figure IV.9: Network Architecture. The numbers in each layer (e.g. 64 and 3×3 in Conv1 refer to the number of kernels and their size, /1 denotes a stride of 1,and the size of max-pooling window is equivalent to its stride (e.g./2 denotes window size 2×2, pooling with a stride of 2). **concat.** denotes stacks the feature maps at the same spatial locations. **add.** denotes the sum of the feature maps at the same spatial locations.

is that the distribution of spatial and temporal features are very different that training could become unstable and difficult to converge if simply adding them together; the other reason is the number of feature channels is not identical between the spatial (256 channels) and temporal (144 channels) feature maps so that they can not be added directly. The **Conv5** layer performs a convolutional fusion which projects the concatenated motion-appearance representation to K classes score maps (denoted by S). Instead of using fully connected (FC) layers, a Global Max Pooling (GMP) layer is employed to obtain the score vector for classification because GMP enforces correspondence between feature maps and categories and the feature maps can be easily interpreted as categories confidence maps. Furthermore, the fully connected layers are prone to overfitting and heavily depend on dropout regularization (see Chapter III), while there is no parameters to optimize in the GMP that is itself a structural regularizer to prevent overfitting.

The fully convolutional architecture makes it flexible with regard to the size of input frames. The two-stream ConvNet was trained with a multi-scale scheme (introduced in the previous session) as illustrated in Fig.IV.10 . All input frames are first rescaled by a factor $s\epsilon\{0.03, 0.05, 0.07, 0.09, 0.14, 0.2, 0.25, 0.33, 0.5, 1\}$. The network enclosed by the red box (in Fig.IV.9) was applied on the multiple downsized versions of the input frames. Feature maps of the down-sampled versions are

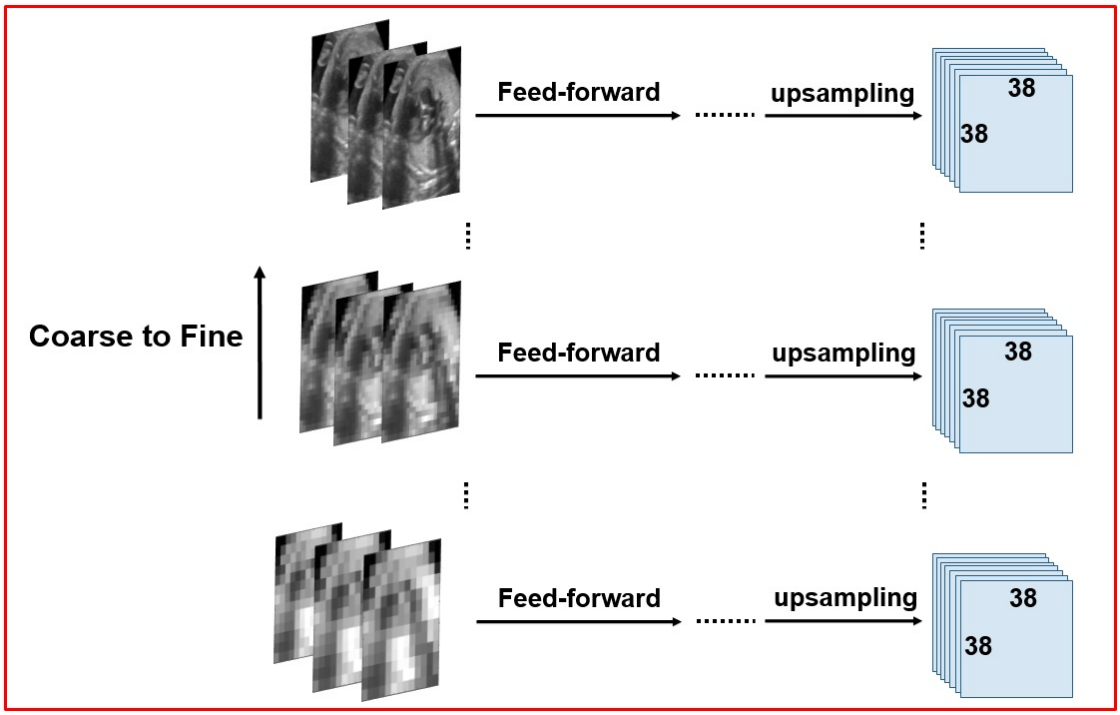


Figure IV.10: Coarse to Fine Augmentation. The network enclosed by red box applied on multiple down-sampled versions of input frames.

brought back to the common resolution i.e. 38×38 by bilinear up-sampling, and then concatenated (motion stream) or added together (appearance stream). As both the motion and appearance CNN are shallow, the detection of features is limited by its small effective receptive field. Therefore, the multi-scale processing is used to compensate for this. In addition, the scale of fetal hearts can vary significantly in free-hand scans. The multi-scale scheme includes some scale-invariance in the network.

IV.5 Experiments and Results

The proposed two-stream CNN was trained for detecting not only fetal hearts, but also the fetal skull and the fetal abdomen (plus background resulting K=4, in our experiments) with a short temporal window of three consecutive frames. Choosing such a short temporal window is mainly because of the limitation of hardware (CPU training and inference) that from experiments, it found a drastic slowness at both training and inference time when processing sequences longer than three frames.

The motion stream is pre-learned to detect motion, and the weights learned from motion detection was frozen when learning the classification of the short video clips. Gradient back-propagated from the classification loss was allowed to flow through the fused network and the appearance network only to update the weights accordingly. In the following experiments, the author both quantitatively and qualitatively evaluates the classification performance and detection performance of fetal heart. Particularly, a guided-backpropogation[111] was introduced (improved method of [100]) to visualize the fetal-heart detection saliency maps. Through experiments, it finds the proposed model learns to adapt to the importance of (discriminative) features for predicting different structures. For instance, the activation of fused representation is dominated by the motion feature when predicting the fetal heart, while the fused representation learns to discard the motion features and the activation is dominated by the appearance features when learning the structures relatively static, such as fetal skull and abdomen.

IV.5.1 Implementation Details

The motion network was pre-trained on the Middleburry [109] and MPI Sinitel [110] dataset to extract optical flow. As introduced in Section 3.2, the motion network was first trained for a flow classification with a cross-entropy loss at each pixel. Then, the network is fine-tuned for flow regression by minimizing Euclidean end-points error loss (i.e. L2 loss) over the network output. The appearance network is initialized with a VGG very deep (11 weights layer) architecture, pre-trained for classification of ultrasound images. The proposed two-stream CNN model was trained for 20 epochs with initial learning rate of 0.01 and continuously reduced to 10^{-4} until training termination. Models were trained using min-batch (batch size = 52) Stochastic Gradient Descent (SGD) with momentum (momentum rate 0.9 and weight decay 0.0005). To cope with imbalanced training classes, the cross-entropy loss was re-weighted with the reciprocal of class frequency in the training set such that learning

gives more attention to minority classes (the fetal heart in our experiment). 50% dropout was also added after the Conv5 layer to prevent overfitting.

IV.5.2 Classification Results

The performance of the two-stream network was evaluated in terms of per-class classification accuracy, precision and recall scores respectively (as summarized in Table. IV.1). And also the appearance network was studied alone. 10-fold cross validation was implemented that the whole dataset was shuffled through and 10% was kept as test dataset at each fold. Here the author reports the mean and standard deviation of the metrics i.e. accuracy, precision and recall on the test datasets. In Chapter III, AlexNets and its tailored versions were trained for fetal ultrasound image classification from scratch (referred to as DCNNs). The author has further experimented on other state-of-the-art CNN architectures for classifying single frames. These single stream architectures were tailored, in which only single fully-connected layer was used with a reduced number of neurons 4096→1024, and the training was carefully regularized with dropout and data augmentation to avoid overfitting. Furthermore, the tailored AlexNet has been experimented for classifying on three consecutive frames. The early fusion approach [61] was used to combine information across an entire temporal window (i.e. three consecutive frames) immediately on the pixel level. This was implemented by modifying the filters on the first convolutional layer in the single-frame model by extending them to be of size $11 \times 11 \times 3 \times 96$). The performance of our proposed two-stream model is compared with these modified state-of-the-art models in Table.IV.1. TCNNs refer to transfer learning CNNs that has been introduced in Chapter III. It found that the TCNN achieved the best results among the CNNs trained on single-frame. And classifying on three frames (e.g.AlexNet) did not make much difference overall, in terms of the metrics, compared to the CNNs classifying on single-frame. However, it should be noted that the proposed two-stream ConvNet model achieves much better classification performance for most performance metrics compared to the

Table IV.1: Classification performance (%) for our proposed two-stream ConvNet and its comparison with other single-stream spatial ConvNets (10-fold cross validation).

Models	Fetal Skull			Fetal Abdomen			Fetal Heart (all views)		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
AlexNet [8] tailored	90.4±1.7	83.5±2.5	73.8±2.2	87.4±3.6	83.1±3.0	73.2±3.6	65.4±3.9	54.2±4.3	50.3±3.1
CNN-M [112] tailored	91.8±2.8	84.2±1.4	75.4±2.1	89.6±4.2	85.3±3.6	74.7±3.7	61.7±4.3	57.4±3.8	51.0±3.7
CNN-S [112] tailored	85.5±2.6	82.0±2.1	73.5±3.0	80.1±3.2	78.2±3.8	72.3±4.1	59.1±3.7	50.5±3.2	54.0±2.8
VGG-VD-A [56] tailored	91.5±2.7	80.2±2.3	75.1±2.0	85.4±5.2	80.2±3.3	70.4±5.2	55.1±4.3	52.1±4.1	51.0±3.8
TCNN [81]	**96.3±1.4**	**85.7±1.3**	**77.3±2.3**	**91.2±4.5**	**85.5±3.9**	**75.2±6.1**	**72.3±3.8**	**65.5±3.2**	**60.1±4.8**
AlexNet [8] (three frames)	90.1±2.3	82.4±2.1	74.4±2.5	88.5±3.3	84.2±3.7	74.6±3.2	64.5±3.6	55.6±3.4	51.4±2.9
Appearance Stream (as proposed)	93.3±2.3	85.7±2.1	76.6±2.2	89.3±3.7	85.2±3.5	75.1±3.6	**74.7±2.9**	**69.4±3.2**	**64.3±3.4**
Two-stream CNN (as proposed)	**94.5±2.8**	**86.3±2.8**	**78.2±2.6**	**90.2±3.5**	**86.4±4.2**	**79.3±3.9**	**85.5±4.8**	**82.6±3.7**	**76.0±3.2**

single-frame CNNs. Overall, the class 'skull' has relatively smaller variance in the metrics compared to other classes because it is the simplest class to be modelled. In comparison, the performance variance is more significant when it comes to learning the class 'heart' and 'abdomen'. This might be explained by the high intra-class variation and also the stochastic learning process (no guarantee converge to a global mininma). Regarding the fetal heart recognition, it should be noted that the two-stream network achieved much better results compared to the other models. The two-stream network achieves the best accuracy, precision and recall of 85.5%, 82.6% and 76.0%, respectively, which outperforms the second best model (i.e. appearance network alone) by a large margin (10.8%, 13.2% and 11.7%, respectively). This sufficiently demonstrate the superiority of incorporating the motion in learning the fetal heart recognition. In comparison between the classes, the class skull achieves the best accuracy of 94.5%, which is slightly higher than the abdomen and heart achieving an accuracy of 90.2% and 85.5%, respectively. All the classes achieve the comparable precision and recall.

IV.5.3 Localization Results

Intermediate Activation Maps

To understand how the proposed two-stream CNN works, Fig.IV.11 looks into a two-stream CNN learned for heart recognition, which demonstrates the intermediate activation of the representations at different convolutional layers. A test example of three convective heart frames is fed forward through the two-stream CNN that the hierarchical feature representations are produced in both streams. The representations at each Conv layer are then max-pooled channel-wise to pick the maximum activation at each spatial location: $max_c(F_l(i,j))$, where the c denotes the feature channel index, F_l is the feature maps at the l^{th} convolutional layer, (i,j) is the spatial index. Finally, the activation maps are brought back to the original resolution by bilinear upsampling, as illustrated in Fig.IV.11.

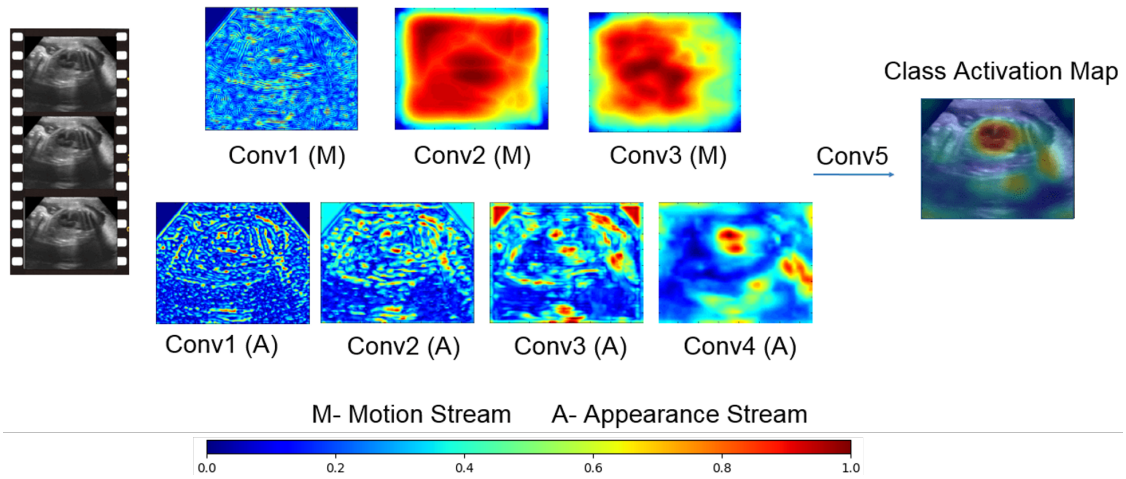


Figure IV.11: Intermediate layer's activation. It shows the progressive evolution of activation of features when feeding-forward through the motion and appearance network respectively, and finally after convolutional fusion i.e. Conv5 a class activation map is obtained.

It should be noted that no matter which stream, the activation is progressively specific to the target i.e. fetal heart. In the low level representation i.e. Conv1, both streams preserve the small structures i.e. edges, the activation can be seen less significant than higher levels which are gathered around the small structures. As approaching higher level (see the Appearance stream), the activation is progressively stronger and less scattered. Conv4(A) is already able to capture the semantic patterns i.e. fetal heart although it is not sufficiently discriminative (high activation elsewhere). It should also be noted that the extent of activation in the motion representations, such as Conv2(M) and Conv3(M) is significantly larger than the mid-high level activation in the appearance stream. This is because the motion stream works as a motion detector designed to capture the motion at different scale. Particularly, Conv2(M) aggregates the smoothed multi-scale motion representations which can be seen that the activation is not discriminative at all. The Conv3(M) learns to decode the aggregated motion into a representation which discriminates the motion specific to the heart beating. As can be seen, the activation of Conv3(M) is specific to the heart area that the activation elsewhere is significantly decreased. Eventually, Conv5 i.e. fusion Conv learns to fuse the motion and appearance representation and decode it into a class-specific activation map, which

is semantically discriminative. It can been seen from the example in Fig.IV.11 that the classification activation map precisely captures the fetal heart and its extent.

Saliency by guided backpropogation

As demonstrated in Fig.IV.11, the localization of fetal heart is approximated by bilinear up-sampling the class-specific score map S_k (more examples see Introduction Fig.IV.1). To have a more precise localization, the visualization method introduced in Chapter IV Section 1.2 was used that it calculates how much each original input pixel I_{ij} contributes to the activation of the class-specific score in the final output, and obtained localized saliency maps M_{ij} with the resolution of the original input frames. The maps are obtained by computing the partial derivatives $M_{ij}^k = \frac{\partial y_k}{\partial I_{ij}}$ through backpropagation, where k denotes the index of the target label.

In visualizing the proposed two stream CNN, a slightly modified and improved method was used which is also based on backpropagation, so called **guided backpropagation**, originally introduced by Springenberg et al. in [111]. The method rectifies the gradient of ReLUs during backpropagation. As the author is only interested in finding the derivative of the class score with respect to the input, the gradient refers to the derivative with respect to the input (or activations in the case of gradients of intermediate layers), $\mathbf{x}$, and not the weights of the CNN. During standard backpropagation, a ReLU unit backpropagates the gradient of the layer next to the ReLU but for only the positive portion of the input:

$$\frac{\partial y_k}{\partial x_l} = \frac{\partial y_k}{\partial x_{l+1}} \cdot \mathbf{1}\left(x_l > 0\right) \tag{IV.7}$$

where $\mathbf{1}(\cdot)$ is the element-wise indicator function, x_l and x_{l+1} are the input and output of ReLU activation respectively. $\frac{\partial y_k}{\partial x_{l+1}}$ is the gradient of the layer succeeding the ReLU and $\frac{\partial y_k}{\partial x_l}$ is the gradient after backpropagating through ReLU. Springenberg et al. [111] further restricts the gradient of the ReLU unit by only propagating positive gradients with positive inputs:

$$\frac{\partial y_k}{\partial x_l} = \frac{\partial y_k}{\partial x_{l+1}} \cdot \mathbf{1}\left(x_l > 0\right) \cdot \mathbf{1}\left(\frac{\partial y_k}{\partial x_{l+1}} > 0\right) \tag{IV.8}$$

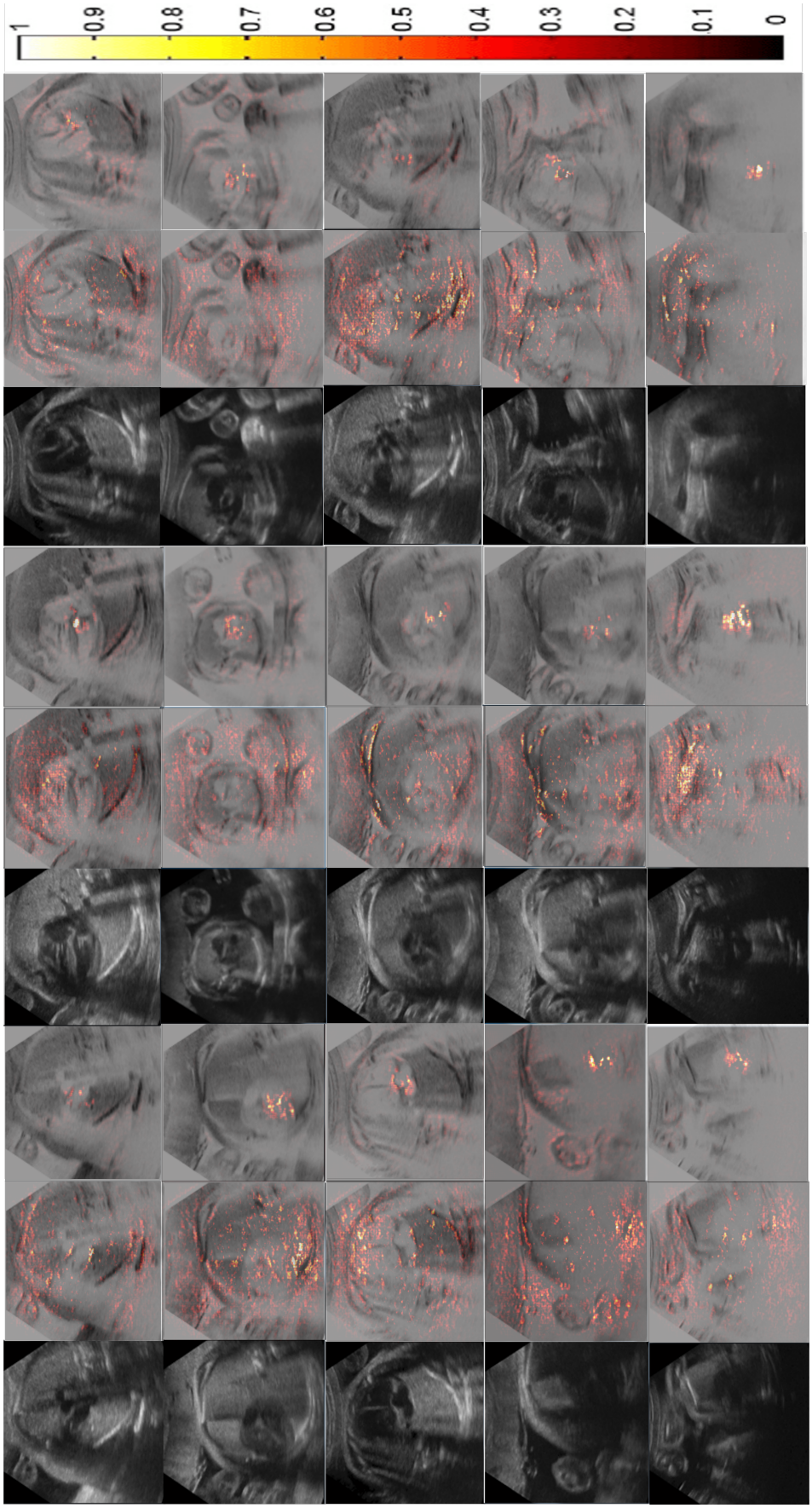


Figure IV.12: Localization saliency (Spatial CNN vs. Two stream CNN) of test heart examples. 1st Col: **Raw input**. 2nd Col: **Single stream CNN**. 3rd Col: **Two-stream CNN (as proposed)**. First row shows the standard planes (i.e. three vessel, four chamber, and LVOT), the rest of examples are all non-standard planes.

This is in a way a combination of normal backpropagation (introduced in Chapte III Section III.5.2) and the deconvolution method (introduced in Chapter III Section III.5.1) which has a ReLU gradient that depended only on the positivity of the gradient of the next layer i.e. $\frac{\partial y_k}{\partial x_l} = \frac{\partial y_k}{\partial x_{l+1}} \cdot \mathbf{1}\left(\frac{\partial y_k}{\partial x_{l+1}} > 0\right)$.

To evaluate the localization performance, Fig.IV.12 illustrates the localization saliency maps of fetal hearts produced by the best single stream CNN (i.e. Parameters(3.3M)+dropout(0.7)+BN in Chapter III) and our proposed two-stream CNN. Hotspots on the maps indicate pixels that contribute to the activation of class-specific score. The brighter the hotspots, the greater the contribution. Of note, the activation hotspots are highly random and scattered in single-stream CNN that they either focus on the structures outside the abdomen area or ribs (appearing high intensity). This may explain why the single-stream CNNs generalize poorly in detection of fetal heart that the saliency maps suggests, instead of leaning discriminative features of hearts, the single-stream CNN learned loads of noise that irrelevant to the fetal heart. In contrast, by incorporating temporal contextual information i.e. motion, our two-stream CNN localizes fetal hearts precisely, as can bee seen the saliency maps are way more discriminative and better localized to the fetal hearts compared to those of the single-stream CNN. And the localization is robust to the variation of view points, scale and location. Even in some very hard situations, the last row in Fig.IV.12, for example, shows very low contrast of fetal hearts, in which cardiac structures are almost invisible. However, the two-stream CNN still captures the location of the fetal heart very precisely.

The improved saliency may also be explained by the guided backpropogation that helps improving the discriminativity of the two-stream CNN in the way that it discards the negative gradient (deactivating the target score). As our objective is to minimize the loss, an increase in loss indicates an increased error of predicting the ground truth which gives the negative gradient. The two-stream CNN eliminates the negative gradient and facilitate the positive gradient to flow back through the network to correct the prediction during training time.

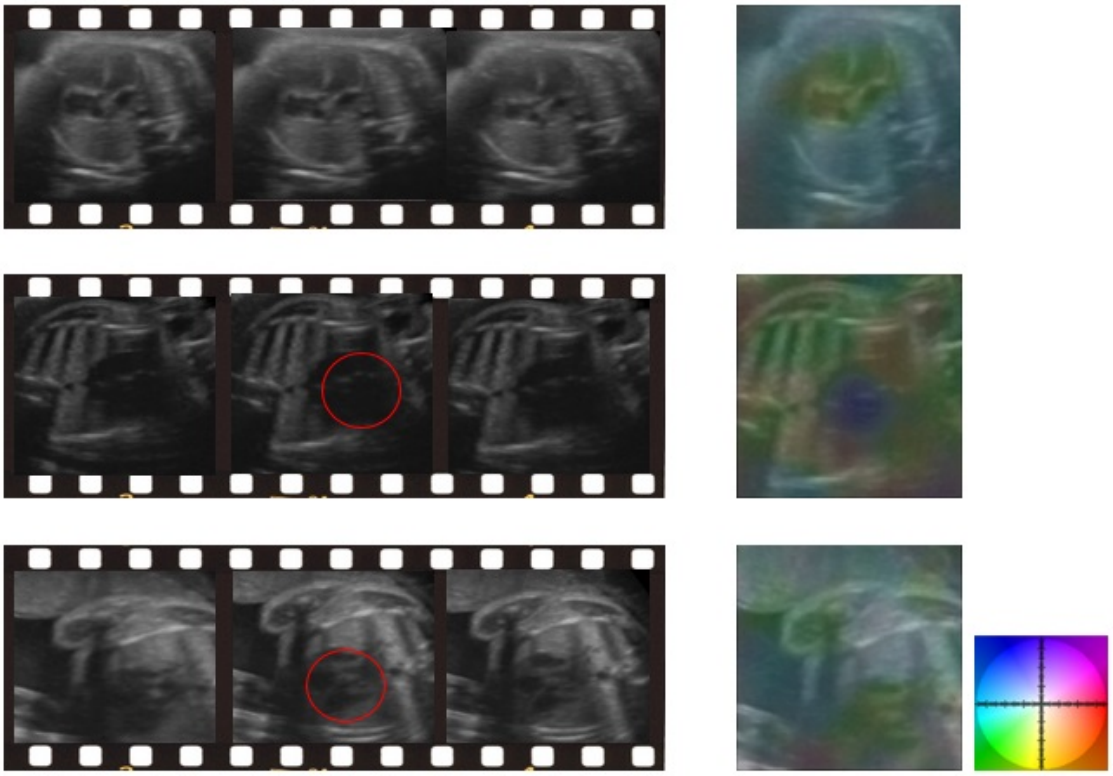


Figure IV.13: Characterization of fetal heart motion (optical flow is color coded).

IV.5.4 Characterization of Heartbeats

Fig.IV.13 illustrates the optical flow maps extracted from the network, which characterize motions locally and globally (such as probe moving). The flow maps are color-coded to show the direction of motion, in which darker colors indicate a larger motion speed. The flow maps provide visualization of fetal heart function locally. Even in low contrast cases (like the last two examples), the model can still sense the motion of the heartbeat.

To examine the heart beating closely, the obtained flow maps were thresholded to identify the most active region. For example, Fig.IV.14 shows the optical flow map of a four chamber view that small flows are removed after thresholding. The thresholded flow map can be seen that it highlights the tricuspid valve (valve between right ventricle and right atrium) which, in the left frame, is moving downwards i.e. closing, while in the right frame, is moving upwards i.e. opening, It should be noted

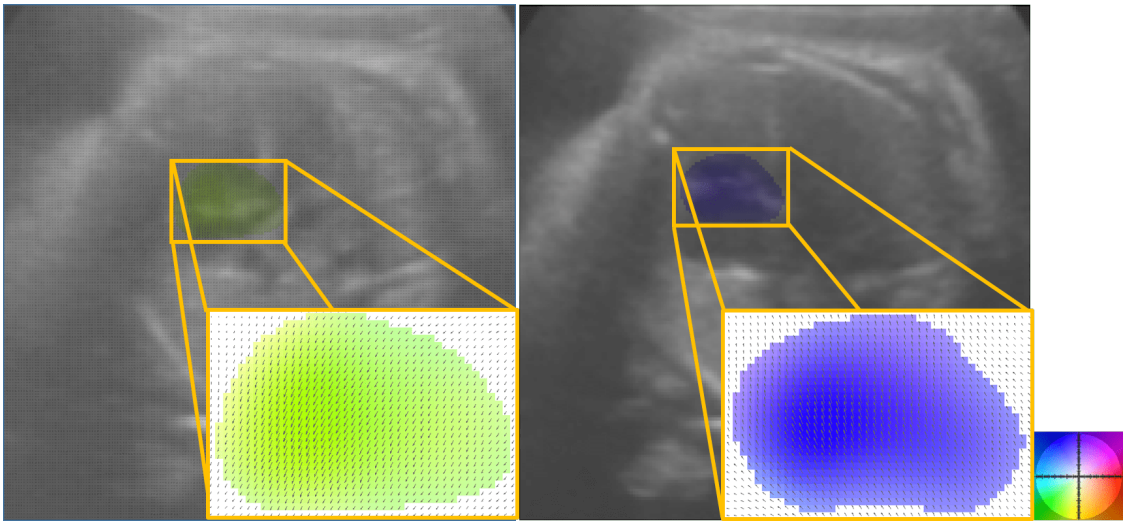


Figure IV.14: Threshold and zoom into the flow map.

the upward and downward flow are slightly off the central vertical axis which is because of the global flow added to the local flow. In this particular example, the global flow points to the left horizontally (the first row in Fig.IV.13) which causes both the upward and downward flows are deviated to the left, as can be seen by zooming into the particular region in Fig.IV.14. If the flow is played in context, the tricuspid valve can be seen moving up and down periodically. Visualizing the motion pattern helps to identify the abnormalities of the cardiac functionality and also to verify the global moving of the probe.

IV.6 Discussion and Conclusion

To summarize, the author has demonstrated that temporal information of consecutive sequences i.e. motion can provide contextual clues for better discrimination in recognition of the fetal heart in general freehand ultrasound sweeps. In this chapter, a two-stream model was proposed and designed to be fully convolutional that completely get rid of the fully-connected layers. In Chapter IV, the author argues that the fully-connected design tends to learning a global feature descriptor discarding the local details. The merit of the fully-convolutional design is two fold: (1) it substantially reduces the network size i.e. the number of trainable

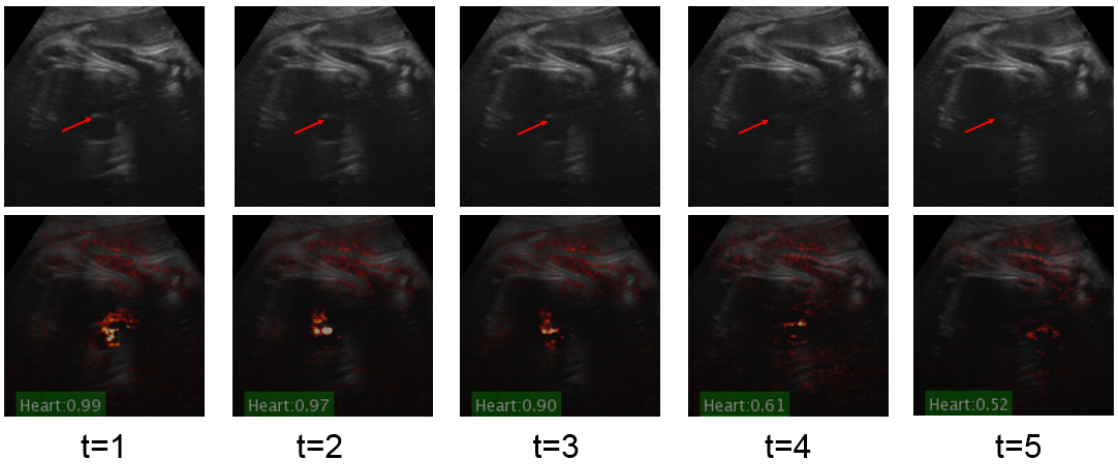


Figure IV.15: Localization saliency of fetal hearts at test time. Red arrow points the cardiac valve which gradually disappears.

parameters which helps to prevent the learning from overfitting; (2) it preserves the spatial correspondence of features which produces representations that are well described locally (important for localization of structures). The Network consists of two streams: a motion detector and an appearance discriminator that the motion detector is pre-learned on optical flow datasets and is then joint with the appearance discriminator for classifying the fetal structures. Localization is learned in a weakly supervised way with image-level labels only. A Guided backpropogation is trained for visualizing the class-specific activation that the results in Section 5.3 have demonstrated the two-stream CNN generates much more discriminative localization saliency than single stream CNN (trained in Chapter III). For instance, Fig.IV.15 illustrates the localization saliencies generated from a very tricky example sequence (extremely low contrast) at test time. It can be seen that the hotspots around the cardiac valve (pointed by red arrow) are significantly brighter than the hotspots elsewhere. The valve gradually disappears while the hotspots are less bright and the prediction confidence is also decreased dramatically. The improved saliency (compared to single stream CNNs) may be explained by: (1) the representation of fetal heart becomes more discriminative after incorporating the motion features; (2) from the training perspective, the guided backpropogation eliminates the negative gradients (loss with respect to image pixels) that deactivate the ground truth prediction. The ground truth here is the image level labels (one-hot labels) for

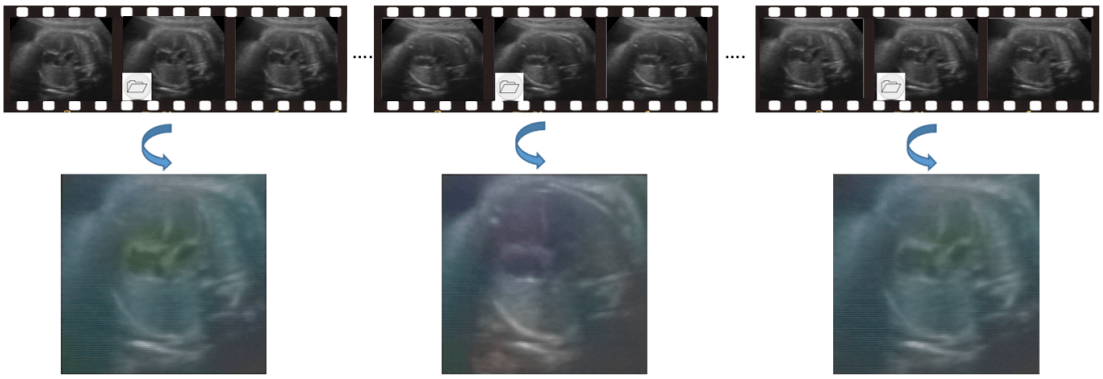


Figure IV.16: Indexed flow maps for characterizing the fetal cardiac motion.

training sequences. As the objective is to minimize cross-entropy, loss decreases (error minimized) and backpropagates positive gradients when the prediction (target probability) approaches one. On the contrary, loss increases (error increased) and backpropagates negative gradients when the prediction (target probability) gets away from one. In terms of quantitative evaluation, the classification results have demonstrated that the two-stream CNN outperformed the best single-stream CNN by a significant margin.

In addition to output classification scores, the proposed two-stream CNN also outputs an optical flow map that describes the motion locally and globally in the given short video sequence. The indexed flow maps, as illustrated in Fig.IV.16, can characterize the heart beating that helps clinicians to closely examine the moving patterns of inside the fetal heart. However, the flow maps can be noisy sometimes if the non-standard heart view planes are captured. The might be because there is no way to fine-tune the motion representations on ultrasound data (because no ground truth available). A potential improvement would make the motion stream unsupervised so that the motion representations can be directly learned from the ultrasound images themselves. And it should also be emphasized that the motion stream is designed to learn a motion detector rather than a motion discriminator in the proposed model. Motion detector here means it only learns to detect where moving patterns are and the moving direction and magnitude. A

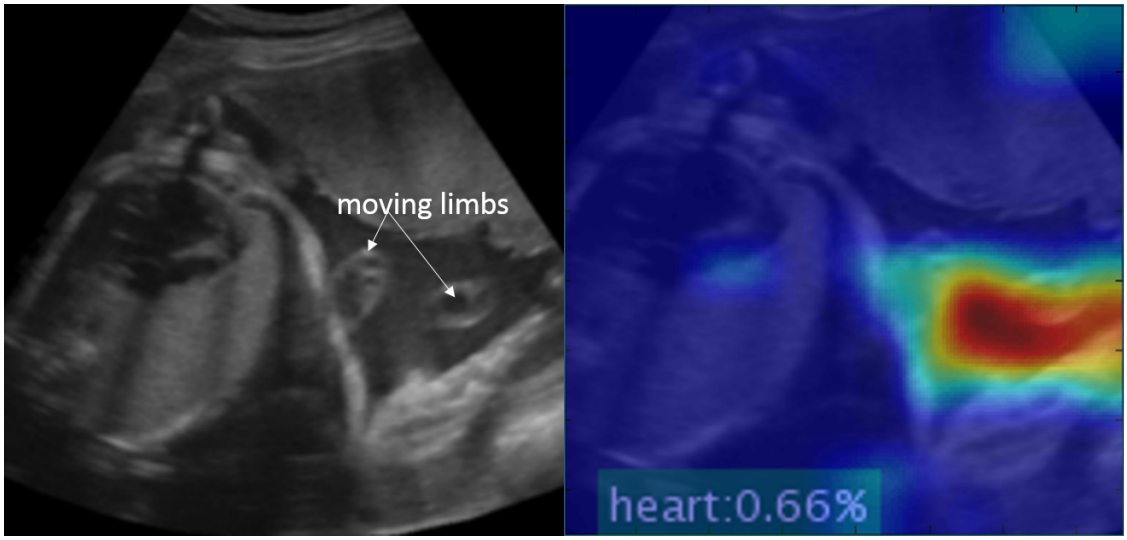


Figure IV.17: Failure example of localizing the fetal heart.

motion discriminator not only learns to detect the motion direction and magnitude, but also learns semantic meaning of moving patters, for examples, it should be able to discriminate a heart beating from a limb moving. Because of the limited capability, the motion stream can be sensitive to the magnitude of motion and sometimes misleading the localization, in the case, there are multiple instance of moving structures in the video, as illustrated in Fig.IV.17. It shows a low confidence of prediction and the class activation map fails to localize the heart. A potential improvement would carry out a continuous learning on recognizing different cardiac moving patterns and the task can be formulated as a classification of cardiac phase (for examples, predicting systole and diastole) given a video sequence. This would guide the network to focus on learning the moving patterns inside the heart and ignore moving patterns outside the heart. However, this can not be achieved by the free-hand fetal ultrasound used in this thesis because it requires to acquire high quality fetal heart videos and annotate the cardiac phase of heartbeat. Alternatively, in next chapter, a more sophisticated spatial-temporal model is proposed which incorporates LSTMs for modelling the video temporal dynamics. This novel hybrid model (CNN+RNN) not only shows promising results on detecting fetal heartbeat but also robust to the distraction of moving patterns that not relevant to the fetal heart.

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [9]

V

Learning Deep Spatial-Temporal Features for Detection of Fetal Structures in Free-Hand Ultrasound Videos

Contents

V.1 Introduction 118
V.2 Dataset 120
V.2.1 Preprocessing 121
V.2.2 Annotation 122
V.3 Delving into CNN for Learning Ultrasound Video Representations 123
V.3.1 Convolutional LSTM 123
V.3.2 Attention-gated ConvLSTM 126
V.3.3 Recurrent Convolutional Networks 127
V.4 Learning Weakly Supervised Classifiers for Localization 128
V.4.1 Cross-Entropy Loss with Doubly Stochastic Regularization 131
V.4.2 Adam Optimizer 132
V.5 Experiments and Results 134
V.5.1 Implementation Details 134
V.5.2 Evaluation of Classification Performance 135
V.5.3 Visualization of Attention 144
V.5.4 Evaluation of Localization Performance 146
V.5.5 Evaluation of running time 150
V.6 Discussion and Conclusion 150

As introduced in Chapter III and IV, our fetal ultrasound data is dominated by non-standard planes. Identifying and interpreting patterns in the nonstandard fetal ultrasound planes is of great challenge, even for human experts, due to high variability of the anatomies in terms of their appearance, scale and positions. In this chapter, the contribution is three-fold: (1) local dynamics of video clips was modelled by applying convolutional LSTMs on the intermediate CNN layers, which learns to detect fetal structures at various scale. (2) an attention-gated LSTM was proposed which generates temporal attention maps showing the intermediate process of structure localization. (3) our approach is end-to-end trainable, and the localization is achieved in a weakly supervised fashion i.e. with only image-level labels available during training. The works presented in this chapter has been reported in:

Y Gao and JA Noble. "Learning and Understanding Deep Spatio-Temporal Representations from Free-Hand Fetal Ultrasound Sweeps". International Conference on Medical Image Computing and Computer Assisted Interventions (MICCAI), 2019.

Y Gao and JA Noble. "Learning Deep Spatial-Temporal Features for Detection of Fetal Structures in Free-Hand Ultrasound Videos". IEEE Transactions on Medical Imaging (TMI), 2019. (Under Review)

V.1 Introduction

A deep spatial-temporal model is presented for automated detection of fetal structures in free-hand ultrasound videos, which is of great challenge because the inconsistency of imaging quality, and high variability of fetal anatomies in terms of their scale, appearance and location. A novel attention-gated spatial-temporal neural network is introduced and trained in a weakly supervised fashion, for detection of fetal anatomical structures, such as the skull, abdomen and heart, in consecutive frames. Inspired by [113], recurrent neural networks (RNNs) are applied on the

intermediate features of a convolutional neural network (CNN), not solely focusing on the top layer but also on the intermediate layers, to characterize temporal dynamics of features extracted at different scales (as illustrated in Fig.V.1). Applying RNNs on intermediate convolutional maps, however, inevitably results in an increased number of trainable parameters mapping the input-to-hidden transformation due to finer spatial resolution of convolutional maps. To cope with this issue, sparsity and locality are introduced in the Long Short-term Memory (LSTM) cell [114], by replacing its fully-connected linear projection with local convolution, to reduce the memory requirement and model complexity.

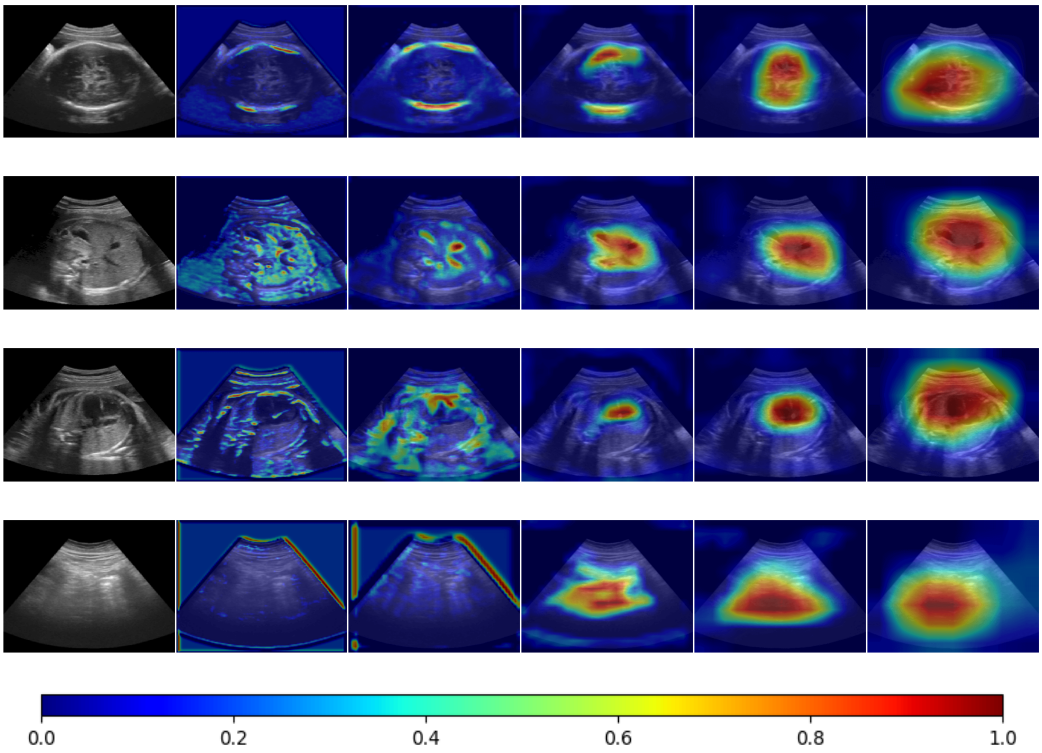


Figure V.1: Class-specific features learned at different scales. Row-wise (top-down): skull, abdomen, heart and other (none of structures of interest). Column-wise (left-right): last frame in a sequence, Conv1, Conv2, Conv3, Conv4, Conv5 i.e. indices of convolution layers

In addition, the LSTM cell is extended to be soft attention-gated, regularized by a doubly stochastic mechanism [115] that encourages LSTM cell to fully exploit local information, and more importantly, provides visual evidence of progressive

refinement of structures localization during forward pass, as shown in Fig.V.2.

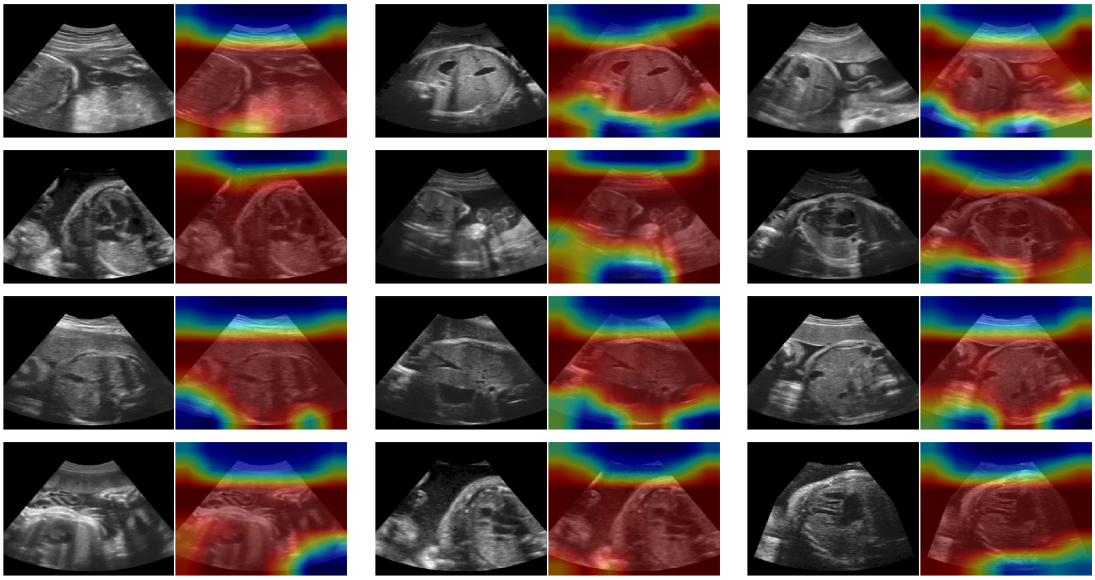


Figure V.2: Doubly Stochastic Attention. Examples of 10 consecutive frames on the left-hand side each column and their respective attention dynamics on ride-hand side. The attention maps show a progressive change of focus spatially over the course of generation, and localisation are progressively corrected. Best viewed in Adobe Reader. All videos should play automatically.

V.2 Dataset

As introduced in Chapter III and IV, our dataset consisted of 456 fetal ultrasound videos of healthy subjects with gestational ages 28 weeks or higher. The videos contained 47234 frames in total, and the average length is 345 frames as shown in Fig.V.3. The videos were annotated by extracting the sequence that contains the anatomy of interest i.e. Skull, Abdomen, Heart and Other. Other included the frames that did not fall into the other three classes. There is normally multiple instances of the Other frames in each video, only one was taken from them to avoid a highly imbalanced annotation. The overview statistics of annotation is summarized in Fig.V.3, depicting the distribution of the modelled classes. The Other takes the largest share and has the longest average video length, whilst the Heart and Abdomen are smaller classes and relatively shorter.

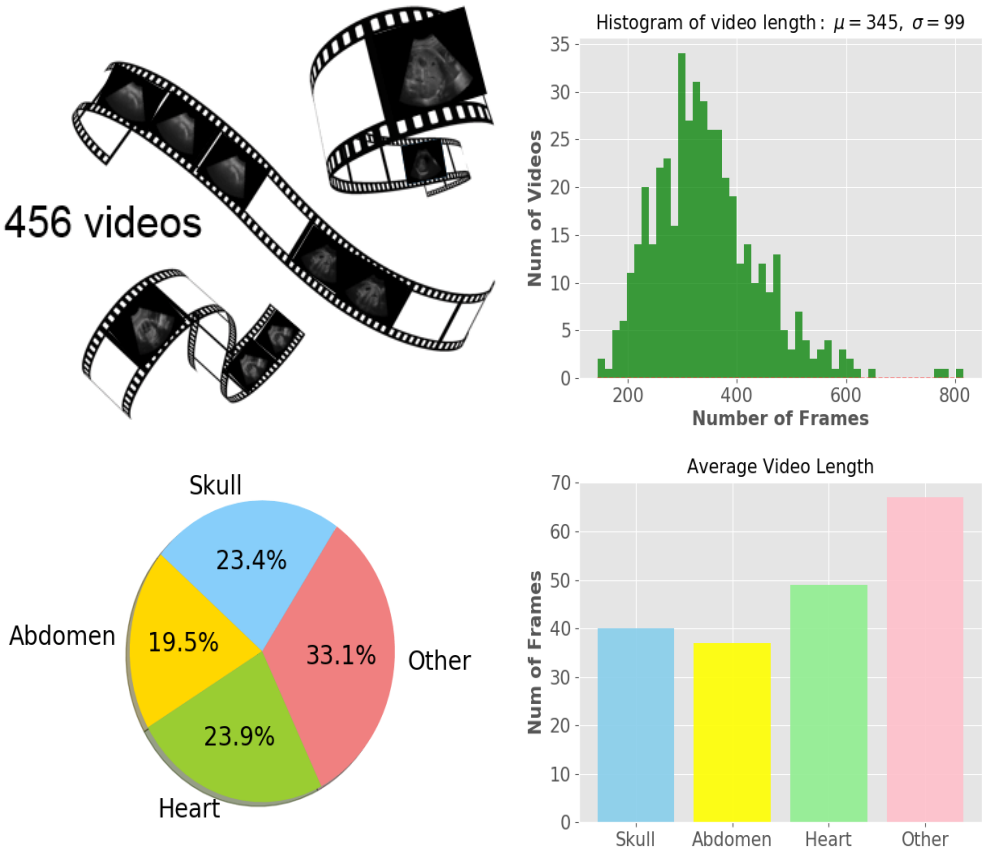


Figure V.3: Data and Annotation.

V.2.1 Preprocessing

Similar to the data preprocessing in Chapter IV, the annotated videos were sub-sampled with steps conditioned on their length (the longer video the larger the sampling step), to create sequences consisting of consecutive frames. This study focuses on short term dependency, so four subsets with temporal size of 3, 5 and 10 frames respectively were created as illustrated in Fig.V.4. In the last step, the datasets were randomly shuffled and prepared for a 5-fold cross validation. For each fold, 20% of the sub-samples are kept for 10% validation and 10% test (no overlapping between each fold), and the remaining 80% as training data. Most importantly, the split is made according to the video identity to guarantee that no samples originating from validation and test videos were used for training.

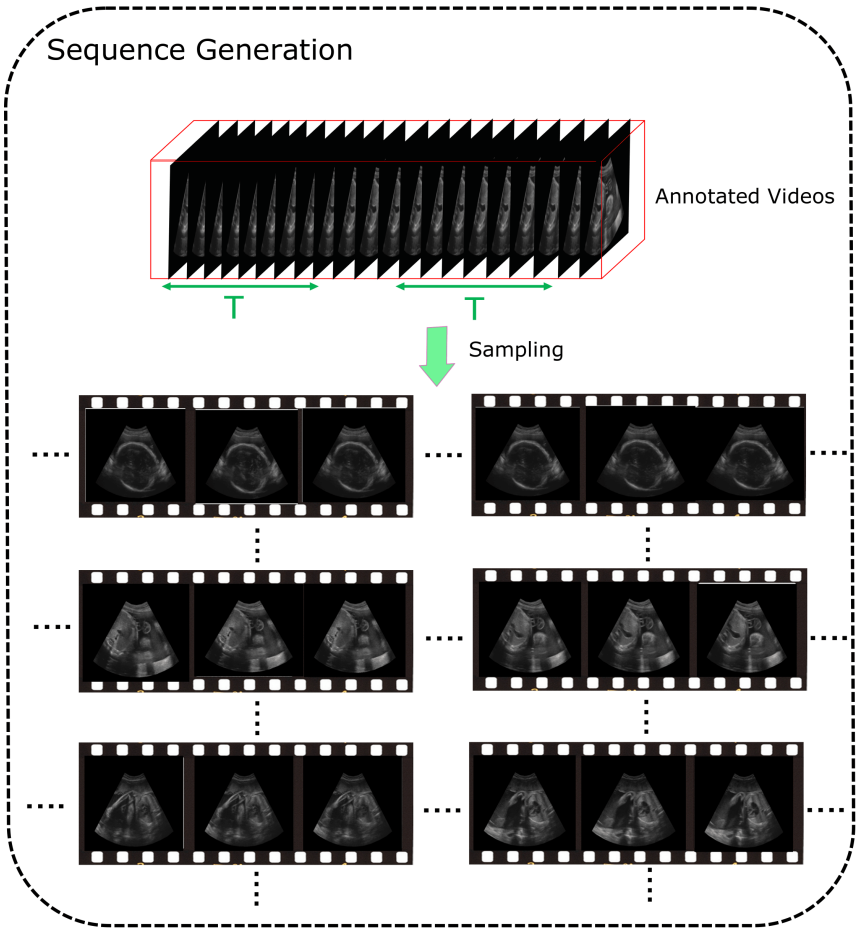


Figure V.4: Steps for Preprocessing. Frame Generation: Frames are masked to keep the *field of view* only by removing the texts on the top and scale marks near the right boundary of frame. Sequence Generation: Sampling consecutive frames with Tϵ {3, 5, 10} frames in our experiments.

V.2.2 Annotation

To evaluate the localization quantitatively, the last frames were taken from the test video clips and bounding boxes were annotated around the structures of interest i.e. skull, abdomen, heart. Figure V.5 shows the annotation tool used to annotate the bounding boxes. One bounding box is assigned to each of the test clips since there is only single instance of the structures of interest in each clip. For the class 'other', default box is the whole frame area. To measure the localization performance, the Jaccard similarity coefficient, or the Intersection over Union (IoU), is calculated

between the ground truth bounding boxes and the predicted bounding boxes.

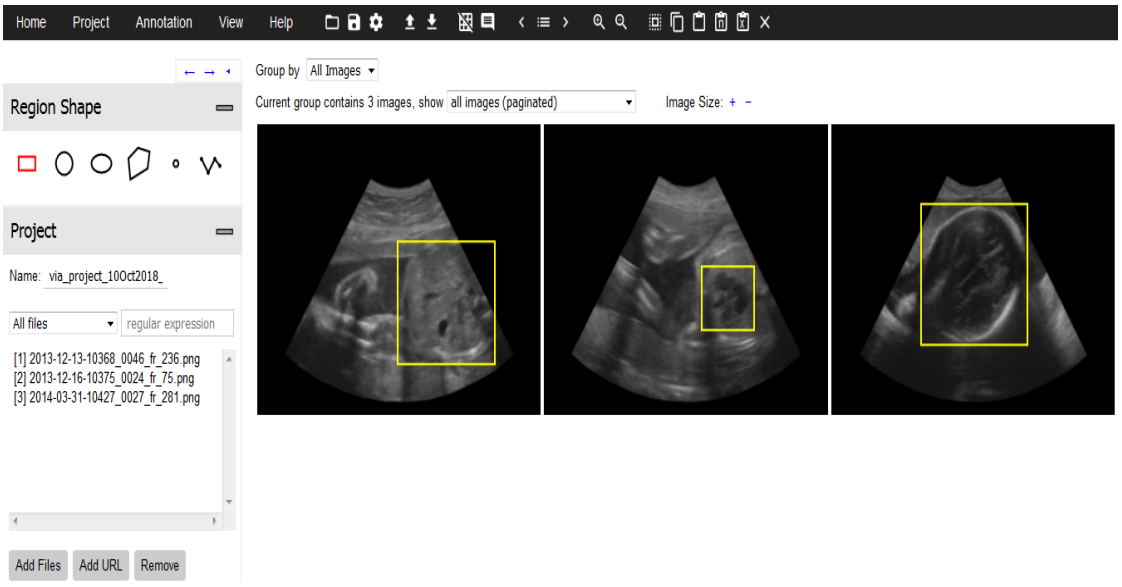
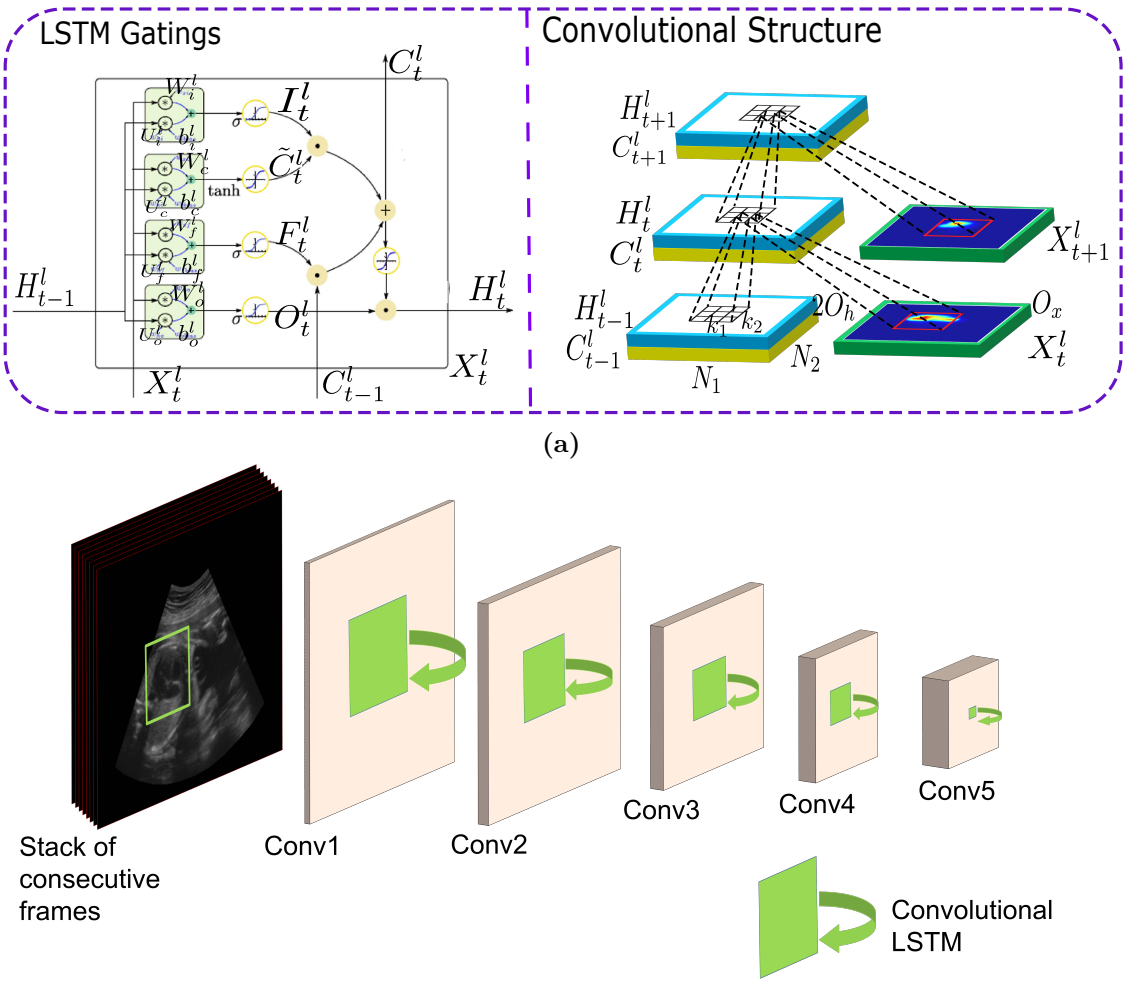


Figure V.5: Annotation Tool. Annotation tool used to annotate the bounding boxes of the structures of interest. Annotation is done on the last frames of test video clips.

V.3 Delving into CNN for Learning Ultrasound Video Representations

V.3.1 Convolutional LSTM

Instead simply applying an element-wise non-linearity to the affine transformation of inputs and hidden states, the The Long short-term memory (LSTM) [114] introduces an internal recurrence (a self-loop), in addition to the outer recurrence of the ordinary recurrent network (RNN). Each cell has the same inputs and outputs as an ordinary RNN, but has more weights and a gating mechanism controlling the flow of information (gradient can flow long duration), which can change the time scale of integration dynamically. A LSTM cell is illustrated in Fig.V.6a and a convolutional LSTM was made operating on the intermediate feature maps of CNNs. The convolutional maps $(X_t^l, ..., X_t^{L-1}, X_t^L)_{(t=1...T)}$ extracted form L layers (L=5 in our CNN) at different time steps in a video, are composed of patterns with



(a)

(b)

Figure V.6: (a) Inside the convolutional LSTM. The input-to-hidden and hidden-to-hidden transitions are done locally with convolution operations. (b) Convolutional LSTMs are applied to a local spatial neighbourhood repeatedly over successive frames.

strong local correlation over time, and the temporal dynamics, such as motion, appearing and disappearing of anatomies, tend to be smooth and restricted in a local spatial neighbourhood i.e. associated with a given patch in successive frames. In other words, the location of key anatomies stay relatively static over successive frames, as illustrated in Fig.V.6b, moving structures related to the fetal heart, such as cardiac valves, are restricted inside the heart area and heart itself is restricted in the thorax area and can not move distantly over the successive frames. Therefore, such a prior was embedded in the proposed model, in contrast to [87] using fully connected LSTM, by replacing the fully connected units with convolution operations.

Therefore, the convolutional recurrent gating configurations have sparse connectivity and share their parameters across different spatial locations, as shown in Fig.6:

$$IG_t^l = \sigma(W_i^l * \tilde{X}_t^l + U_i^l * H_{t-1}^l + b_i^l) \tag{V.1}$$

$$FG_t^l = \sigma(W_f^l * \tilde{X}_t^l + U_f^l * H_{t-1}^l + b_f^l) \tag{V.2}$$

$$OG_t^l = \sigma(W_o^l * \tilde{X}_t^l + U_o^l * H_{t-1}^l + b_o^l) \tag{V.3}$$

$$\tilde{CG}_t^l = \sigma(W_c^l * \tilde{X}_t^l + U_c^l * H_{t-1}^l + b_c^l) \tag{V.4}$$

$$CG_t^l = FG_t^l \odot CG_{t-1}^l + IG_t^l \odot \tilde{CG}_t^l \tag{V.5}$$

$$H_t^l = OG_t^l \odot tanh(CG_t^l) \tag{V.6}$$

where, the gates IG_t^l, FG_t^l, OG_t^l, the candidate memory $\tilde{CG}_t^l$, memory cell CG_t^l, CG_{t-1}^l, and hidden state H_t^l, H_{t-1}^l, at l_{th} Conv layer, are 3D tensors (spatial dimension and input channels) and defined based on a local neighbourhood of size $(k_1 \times k_2)$ at the location (i, j). So, the trainable parameters W_i^l, W_o^l, W_f^l, W_c^l and U_i^l, U_o^l, U_f^l, U_c^l are 2D convolutional kernels, having a size of $k_1 \times k_2$ and all b are learned biases.

The kernel size $k_1 \times k_2$ (in our case 3×3) is chosen to be significantly smaller than convolutional map size $N_1 \times N_2$ for computational efficiency and reducing parameters. A standard LSTM would require input-to-hidden parameters (all W) and hidden-to-hidden parameters (all U) to be of size $N_1 \times N_2 \times O_x \times O_h$, where O_x and O_h are the dimensionality of the convolutional maps and the LSTM hidden states respectively, whilst our design require much less parameters i.e. $k_1 \times k_2 \times O_x \times O_h$. In addition, using convolution also saves computational cost. Each recurrent module applies 8 2D-convolutions at each time step (2 per gate and 2 for computing the candidate memory), which requires $O(4TN_1N_2k_1k_2(O_xO_h + O_hO_h))$ multiplications, that are much lower compared to a fully-connected LSTM requiring $O(4TN_1N_2N_1N_2(O_xO_h + O_hO_h))$ computations.

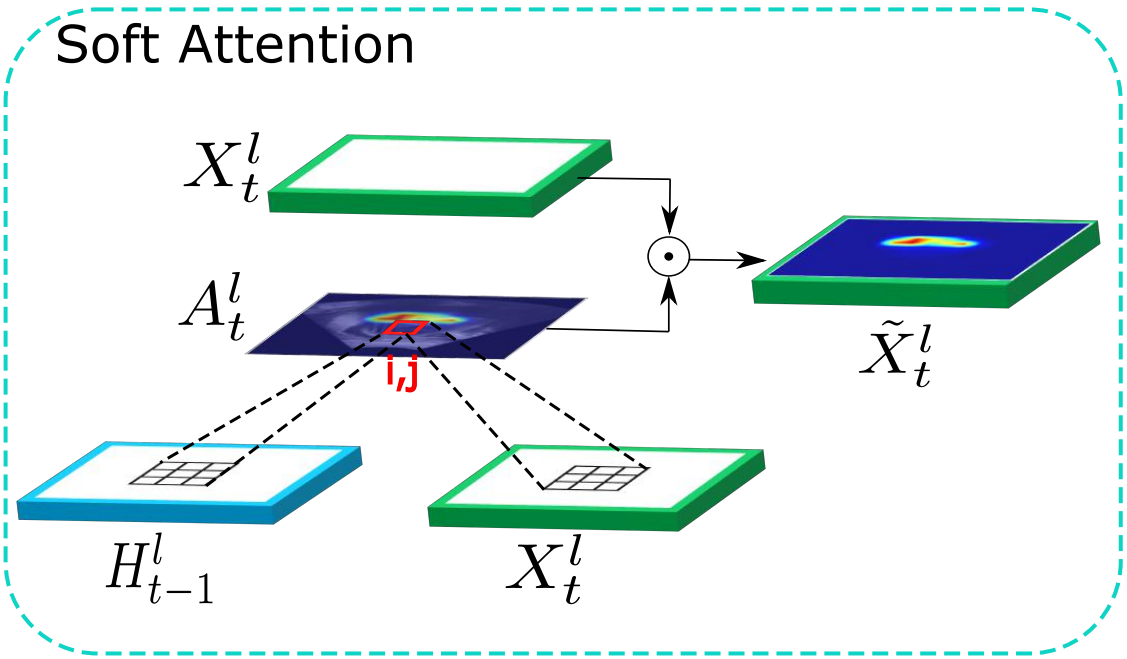


Figure V.7: Inside the attention cell.

V.3.2 Attention-gated ConvLSTM

As illustrated in Fig.V.7, instead of directly applying the convolutional maps to the convolutional LSTMs, a soft attention mechanism is introduced conditioned on the convolutional maps X_t^l and their latest hidden state H_{t-1}^l, which learns to focus on different salient regions over time:

$$Z_t^l = \psi * tanh(W_a^l * X_t^l + U_a^l * H_{t-1}^l + b_a^l) \tag{V.7}$$

where again the gating units are characterised by a set of small 2D-convolution kernels: $W_a^l \epsilon \mathbb{R}^{k_1 \times k_2 \times O_x \times O_h}$, $U_a^l \epsilon \mathbb{R}^{k_1 \times k_2 \times O_h \times O_h}$, $\psi \epsilon \mathbb{R}^{k_1 \times k_2 \times O_h \times 1}$ and bias term $b_a \epsilon \mathbb{R}^{O_h}$. *tanh* is non-linearity activation function. The output of this operations is a 2D map from which a normalised spatial attention map is computed through the equation:

$$A_t^l(i,j) = p(att_{ij}|X_t^l, H_{t-1}^l) = \frac{\sigma(Z_t^l(i,j))}{\sum_i \sum_j \sigma(Z_t^l(i,j))} \tag{V.8}$$

where $A_t^l(i,j)$ is the element of the attention map in position (i, j). Instead of using the *softmax* operator, a sigmoid unit is used, noted as σ, because softmax operation can sparsify the attention map, and can be very sensitive to local changes

even though the network is wanted to attend to the region with high variability. Luckily, our recurrent infrastructure is capable of capturing the spatial-temporal patterns with high variation in time, as the size of receptive field associated with $H^l(i,j)_t$ increases in the previous hidden state H^l_{t-1}, H^l_{t-2}...as going back further in time. The attention mechanism is just to adaptively guide the LSTMs focusing on the salient (high variant) regions and produce the sparse hidden representation feeding into the LSTMs, by applying the attention map to the input convolutional map with an element-wise product between each channel of the feature maps and the attention map:

$$\tilde{X}^l_t = A^l_t \odot X^l_t \tag{V.9}$$

V.3.3 Recurrent Convolutional Networks

Recurrent Convolution Networks (RCN) have been recently introduced for learning video representation in [116], [117] and [118] that leverage both recurrence and convolution. Such approaches focus on characterizing the temporal variation of the high-level representations extracted from the top layers of a CNN. However, the top layers produce very coarse representation that tends to discard local information. For this reason, such approaches typically focus on global appearance changes. To address the issue, a novel RCN is introduced that not only focuses on the top layer of a CNN but also on the intermediate convolutional layers. These intermediate convolutional layers preserve a finer spatial resolution of the input video from which local spatio-temporal patterns are extracted.

The architecture of our proposed model is given in Fig.V.8. Overall, the model consists of a *CNN* for extracting appearance features from consecutive frames and an *Attention Gated LSTM* for processing the Convolutional features recurrently at different locations, exploring temporal variation by selectively attending to different regions of the spatial features maps at different levels. The VGGNet

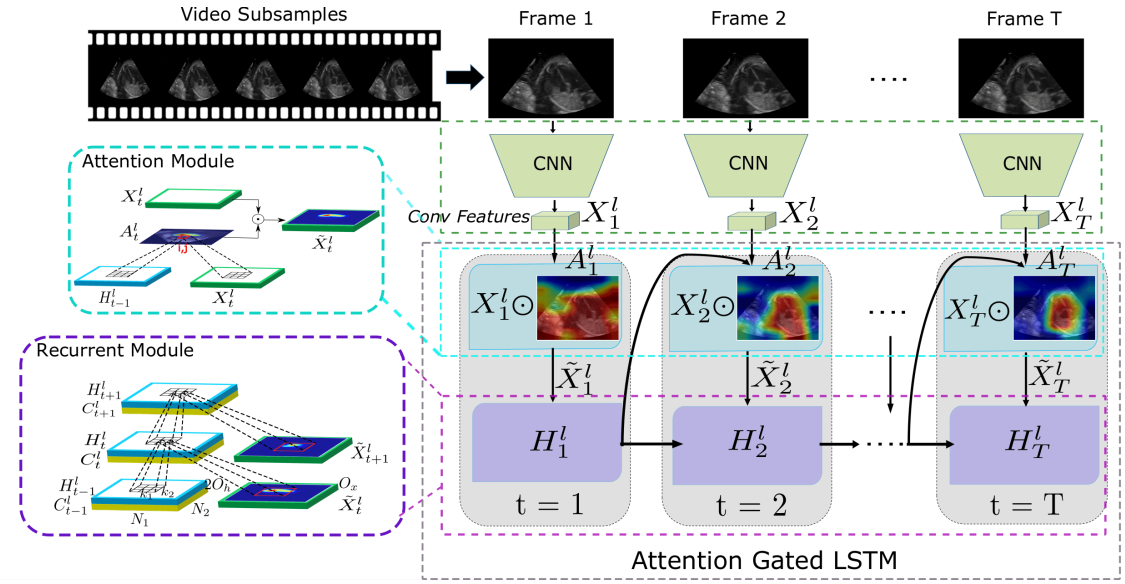


Figure V.8: Network Overview. CNN feature maps are fed to the Attention Module to generate an attention map at each time step, then the attention-warped (element-wise product) feature maps are fed to the Recurrent Module to update the hidden states. Inside the Attention Gated LSTM: the input-to-hidden and hidden-to-hidden transitions are done locally with convolution operations.

[56] configuration is adopted that increases CNN depth using very small convolution filters stacked with non-linearity injected in between. All convolution layers consist of 3×3 kernels, batch normalization (bnorm) [46] and Rectified Linear Unit (ReLUs) non-linearity [8]. The full architecture, using shorthand notation, is $2\times C(32,3,1)$-MP-$2\times C(64,3,1)$-MP-$3\times C(128\ 3,1)$-MP-$3\times C(256,3,1)$-MP-$3\times C(256,3,1)$-MP, where $C(d,f,s)$ indicates a convolution layer with d filters of spatial size $f\times f$, applied to the input with stride s. MP represents non-overlapping max-pooling operation with a kernel size of 2×2.

V.4 Learning Weakly Supervised Classifiers for Localization

The hidden states of the last time step are fed into the weakly supervised classification layers, as illustrated in Fig.V.9, which is built with two state-of-the-art approaches that are adaptation [90], [119] and Class Activation Mapping (CAM) [91].

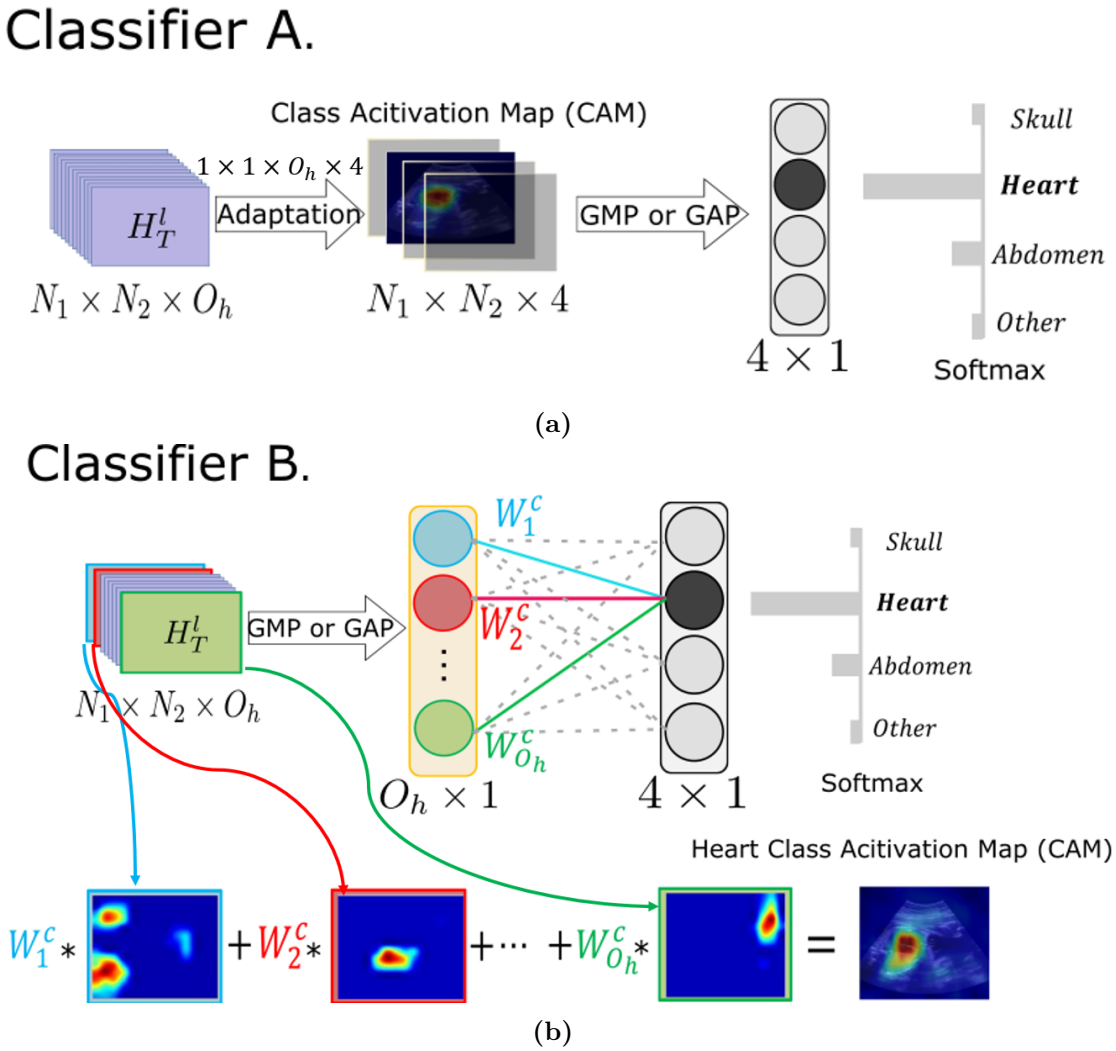


Figure V.9: Weakly Supervised Classification Layers. (a) **Classifier A.**: An adaptation layer (1×1 convolution) is applied to map the hidden representation (at last time step) to Class Activation Maps, which are then fed into a Global Average Pooling (GAP) or Global Max Pooling (GMP) layer to obtain a classification score vector. (b) **Classifier B.**: The hidden representation (at the last time step) is first fed into a GAP or GMP layer to obtain a global feature embedding (vector) then a dense layer is used to map the global feature vector to classification scores.

As illustrated in Fig.V.9a, the adaptation approach applies a 1×1 convolution with a kernel $(1 \times 1 \times O_h \times 4)$ to mapping the feature maps down to the class activation maps without changing its spatial resolution, and the class activation maps are then spatially aggregated using either a Global Max Pooling (GMP) or Global Average Pooling (GAP) operation to obtain a prediction vector, which is fed into the final softmax. In contrast, the CAM approach is different in that it aggregates the feature maps to a feature vector first with global pooling operation, then a dense layer i.e. output layer is applied to mapping the feature vector to categorical scores. Different to the adaptation approach, the CAM approach obtains class activation maps by the weighted sum of feature maps, as illustrated in Fig.V.9b, weights associated with the class heart in the output layer are multiplied with feature maps correspondingly and summation of the weighted feature maps results in the heart class activation map.

Let $F_c(i, j)$ represent the activation of c_{th} channel in the last time step hidden state H_T^l at spatial location *(i, j)*. For a given class k, the class score, y_k, is:

$$y_k = \sum_{i,j} \sum_{c} w_c^k F_c(i, j) \qquad \textbf{Classifier A.} \tag{V.10}$$

$$y_k = \sum_{c} w_c^k \sum_{i,j} F_c(i, j) \qquad \textbf{Classifier B.} \tag{V.11}$$

where, $\sum_{i,j}$ is a spatial aggregation operator, either a global Mean or Max function. $\sum_c$ is a Sum function applied over all feature channels. W_c^k is the weight corresponding to class k for c_{th} feature channel, which essentially indicates the importance of F_c to activate class k. Intuitively, the Class Activation Map is simply a weighted sum of all features maps given by:

$$CAM_k(i, j) = \sum_{c} w_c^k F_c(i, j) \tag{V.12}$$

Where, each spatial unit is expected to be activated by certain visual pattern preserved in F_c within its receptive field. By upsampling the class activation map to the size of the input (bilinear upsampling is used), it can identify the patterns that are most discriminative to the particular class, as shown in Fig.V.1.

V.4.1 Cross-Entropy Loss with Doubly Stochastic Regularization

Our goal is to tell whether an instance of the anatomy of interest (fetal skull, fetal abdomen and fetal heart) is present in a video subsample. This task is formulated as a multi-way classification problem, following the conventional practice that train the proposed model by minimizing the cross-entropy loss:

$$CE = -\frac{1}{N}\sum_{n=1}^{N}\left(y_k(x_n) - log\sum_{k=1}^{C} e^{y_k(x_n)}\right) \tag{V.13}$$

where there are N training video clips x_n and C training classes (C=4), y_k is the k_{th} component in the score vector $\epsilon\mathbb{R}^C$, and k is the true class of x_n.

As an imbalanced classificationin encountered, a class-balanced weighting factor, i.e. **CB** is included in the cross-entropy loss, which is set by inverse class frequency:

$$CB_n = \frac{1}{freq(k)} \tag{V.14}$$

While this only emphasizes the importance of a rare class during training, it does not differentiate between easy/hard examples. For instance, it is relatively harder to differentiate between heart and abdomen classes than between heart and skull, because heart frames often share similar appearance to abdomen frames (as a heart is part of abdomen). So another modulating factor, i.e. *MF*, is introduced to down-weight easy examples and thus focus training on hard examples [120]:

$$MF_n = (1 - P_k)^{\gamma} \tag{V.15}$$

where P_k is the true class posterior probability of x_n, given by softmax as: $\frac{e^{y_k(X_n)}}{\sum_{i=1}^{C} e^{y_i(X_n)}}$. Intuitively, the modulating factor reduces the loss contribution from easy examples. This in turn increases the importance of correcting misclassified examples. When an example is misclassified and P_k is small, the factor *MF* is near 1 and the loss is unaffected. As $P_k \rightarrow 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted. γ is a positive integer which smoothly adjust the

rate at which easy examples are down-weighted. As γ is increased the modulating effect of the factor MF is likely increased.

In addition, a doubly stochastic regularization[115] is introduced to regularize the learning of spatial-temporal attention. By construction in Eqn.(V.8), it obtains an attention map at every frame (time step) which is actually a spatial probability distribution i.e. $\sum_{i,j} A_t(i,j) = 1$ and this is so called single stochastic constraint at spatial level. However, the spatial stochastic constraint is not enough because in this work a spatial-temporal attention is modelled. Thus, it also needs to impose a temporal stochastic constraint i.e. $\sum_t A_t(i,j) \approx 1$ that it encourages (forces) the model to look at each region of the frame at each time step. Together, this is so called doubly stochastic regularization. In our experiments, it is observed that this penalty is important to improve the dynamics and smoothness of generated attention maps. Without the regularization, attention stays static (at same location) over time.

Concretely, the model is trained end-to-end by minimizing the following penalized negative log-likelihood:

$$J = -\frac{1}{N}\sum_{n=1}^{N} CB_n MF_n(y_k(x_n) - log\sum_{k=1}^{C} e^{y_k(X_n)}) + \lambda\sum_{i,j}(1 - \sum_{t=1}^{T} A_t(i,j))$$

V.4.2 Adam Optimizer

In previous chapters, Stochastic Gradient Descent (SGD) with momentum is introduced to optimize the training of deep CNNs. SGD with momentum works well on stabilizing the training with minibatches, however, it does not guarantee good convergence due to the limitations: (1) it can be difficult to choose an appropriate initial learning rate. A learning rate that is too small leading to a slow convergence, while a learning rate that is too large can hinder convergence; (2) the learning rate is adjusted by a pre-defined annealing schedule [121] and thus unable to adapt to a dataset's characteristics [122]; (3) the same learning rate is applied to all parameters updates at each epoch. In our case, data is sparse and features have different

frequencies (class prior). It might not be appropriate to update all of the parameters to the same extent, but ideally apply larger update to rarely occurring features.

To address the limitations, an adaptive optimization algorithm, called Adaptive Moment Estimation (Adam) [123], is used that adjusts the learning rate adaptively according to the importance of features. It performs smaller updates i.e. low learning rates for parameters associated with frequently occurring features, and larger updates i.e. high learning rates for parameters associated with infrequent features. The update rule of Adam is summarized in algorithm 5. As can be seen, in addition to storing an exponentially decaying average of past squared gradients (denotes **s**) like RMSprop [124], Adam also keeps an exponentially decaying average of past gradients (denotes **r**), similar to momentum. Therefore, Adam can be seen as a variant on the combination of RMSProp [124] and momentum with two distinctions: (1) momentum is Incorporated as an estimate of the first order moment (exponential decaying of the gradient); (2) bias correction is included to estimate both the first-order moments and the uncentered second-order moments to account for their initialization at the origin.

Algorithm 5 Adaptive Moment Estimation (Adam)

Require: step size ϵ (default: 0.001)
Require: momentum rates: ρ_1 and ρ_2 (default: 0.9 and 0.999 respectively)
Require: small constant δ (default: 10^{-8} for numerical stabilization.
Require: initial parameters W
initialize 1st and 2nd moment variables r_1=**0**, r_2=**0**
initialize time step t=0
while stopping criterion not met **do**
    Sampling a minibatch of M examples from the training set $\{x^{(1)}, ..., x^{(M)}\}$ with corresponding targets $y^{(i)}$.
    Computing gradient estimate: $\hat{G} \leftarrow +\frac{1}{M} \bigtriangledown_W \sum_i J(f(x^{(i)}; W), y^{(i)})$
    update biased first moment estimate:$r_1 \leftarrow \rho_1 s + (1 - \rho_1)\hat{G}$
    update biased second moment estimate:$r_2 \leftarrow \rho_2 r + (1 - \rho_2)\hat{G} \bigodot \hat{G}$
    correct bias in first moment: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$
    correct bias in second moment:$\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
    compute update:$\Delta W = -\epsilon \frac{\hat{r_1}}{\sqrt{\hat{r_2}}+\delta}$
    apply update:$W \leftarrow W + \Delta W$
end while
return

V.5 Experiments and Results

V.5.1 Implementation Details

Experiments Setup

The proposed model (in Fig.V.8) was trained with the weakly supervised classifiers (in Fig.V.9) for classifying the short video sequences. Four baseline models (non attention gated version) were evaluated in our experiments, which are noted as AD-GAP (Adaptation with Global Average Pooling i.e. Classifier A.), AD-GMP, CAM-GAP and CAM-GMP (Class Activation Mapping with Global Max Pooling i.e. Classifier B.), respectively, so that their attention gated counterparts are noted as AG-AD-GAP (AG: attention gated), AG-AD-GMP, AG-CAM-GAP and AG-CAM-GMP, respectively.

Five recurrent modules are applied independently on each of the convolution maps extracted from the first to fifth i.e. Conv1 to Conv5 MP (max-pooling) layers. For the attention gated models, the attention module (introduced in Section V.3.2) is only incorporated into Conv3 to Conv5 covnolutional LSTMs. All the states of the LSTMs are initialized to zero which corresponds to total ignorance of the future i.e. assuming no prior knowledge about the future. Each input-to-hidden and hidden-to-hidden convolution has zero padded 3×3 convolution preserving the spatial dimension of the inputs. The number of channels of each respective hidden representations are 32, 64, 128, 256 and 256. Five hidden-representations are obtained at each time step. The simplest and most stable aggregation strategy (explained in [89]) is implemented that the hidden representations of the last timestep are fed to 5 separate classifiers. Each classifier therefore learns prediction by focusing on only one hidden representation at a specific scale. The classifier outputs are then averaged to get the final decision.

Training

All the models are trained end-to-end from scratch with an Adam optimizer (learning rate: 10^{-4}, ρ_1= 0.5, $\rho_2 = 0.9$ and $\epsilon = 10^{-8}$). The hyper-parameters: γ in Eqn.(V.15) is set to be 0, 1, 2 and 5 in our experiments, λ in Eqn.(V.16) is either 0 (without regularization) or 1 (with regularization). All the trainable weights are initialized with the method [35]. The networks are implemented with Tensorflow and trained on a Nvidia GeForce GTX 1080 GPU.

Data Augmentation

Several random on-the-fly data augmentation strategies were applied during training, including (1) cropping square patches at the center of the input frames with a scaling factor randomly chosen between 0.7 to 1, and resize the crops to the size of 224×224 (input resolution); (2) rotation with an angle randomly selected within $\theta = -25°$ to 25°; (3) Random horizontal reflection i.e. flipped the frames in the left-right direction with a probability p=0.5.

V.5.2 Evaluation of Classification Performance

Class Activation Mapping

The class activation maps (CAMs), learnt from various models, are illustrated in Fig.V.10, Fig.V.11, Fig.V.12 and Fig.V.13 for all categories, which are generated at the last time step of test sequences. It can be found that Global Max Pooling (GMP) models, especially the AD-GMP generates the most discriminative CAMs (enclosed by green boxes), localizing precisely the target anatomy (even in some very challenging cases, like the heart examples with strong acoustic shadow and highly varied appearance and location), while Global Average Pooling (GAP) models perform less superior, the AD-GAP very often failed to capture the anatomy of interest in most cases (enclosed by red boxes), and although CAM-GAP localize

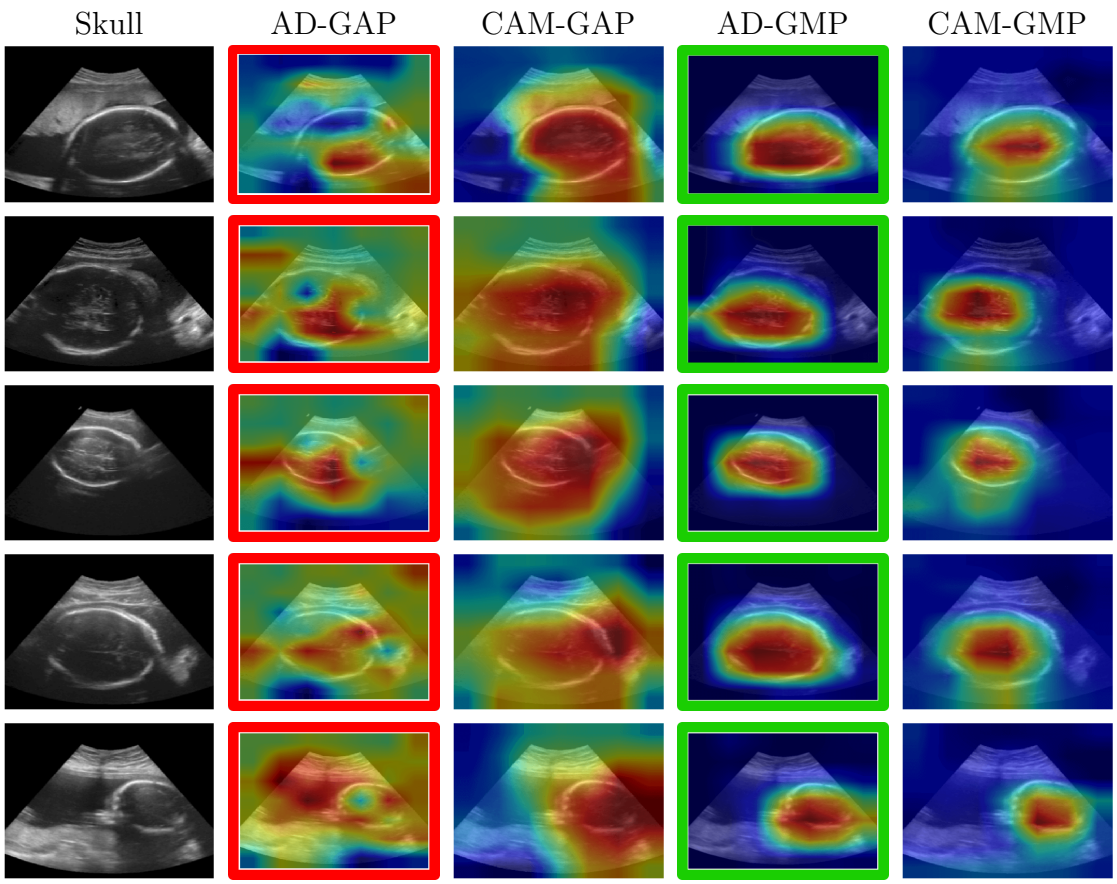


Figure V.10: Skull Class Activation Maps (CAMs). Skull examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization.

well the target structures but it tends to over-estimate its extent. The author argues that this is because, averaging a feature map is prone to incorporate the global context for the classification, while GMP only accounts for the receptive field of the most discriminative unit i.e. maximally activated neuron. Because of the recurrent nature of our model, each neuron has sufficiently large receptive field to capture anatomy of interest with any scale in our task, for instance, the receptive filed at the Conv3 last hidden state is already 484×484 (5 consecutive frames) which is more than twice the input size.

In Fig.V.10, it should be noted that although both AD-GMP and CAM-GMP localize the fetal skull correctly with high confidence, the AD-GMP model

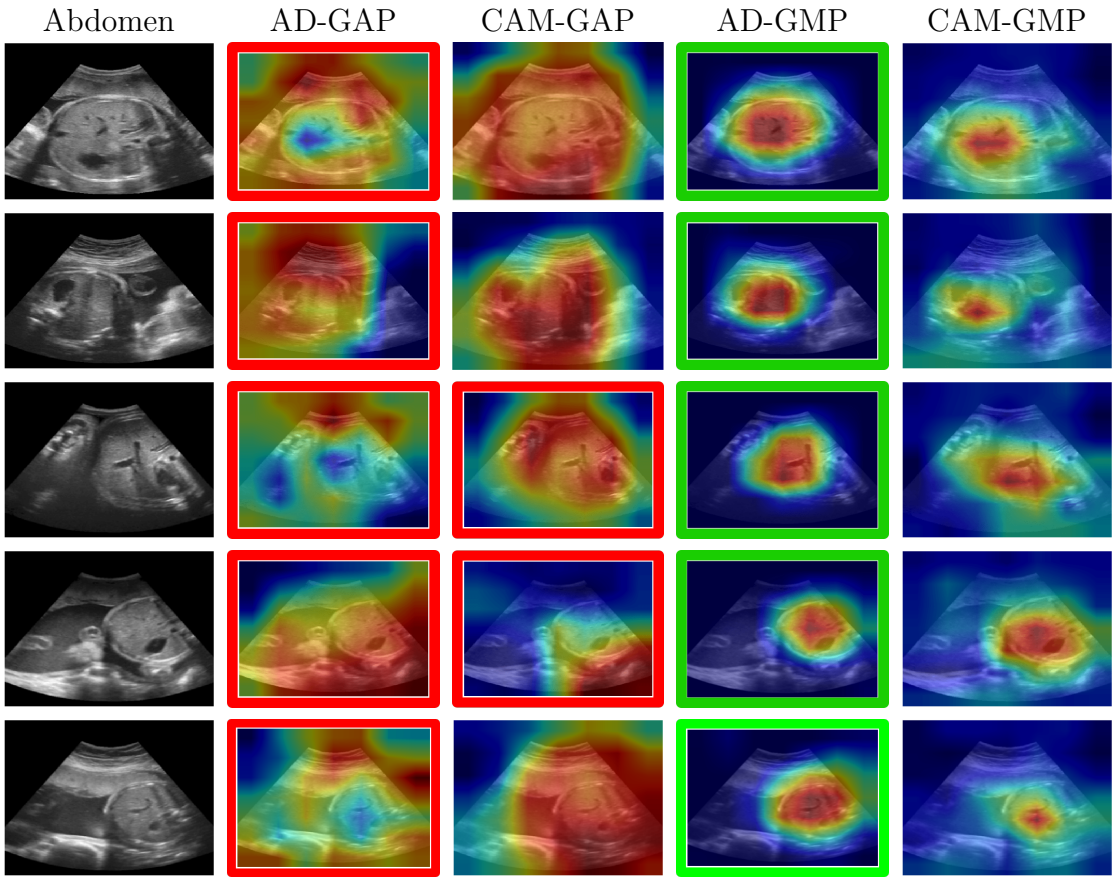


Figure V.11: Abdomen Class Activation Maps (CAMs). Abdomen examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization.

estimates more precisely the extent of fetal skull compared to the CAM-GMP model. GAP models can be seen either failed to localize the fetal skull (see AD-GAP) or overestimate the extent of fetal skull with significant offset in some of CAM-GAP examples.

In Fig.V.11, it can also be noted that both AD-GMP and CAM-GMP capture the whole abdomen area with high confidence, however, the prediction of CAM-GMP is based on the structures inside the abdomen, as can been seen the high confidence area focusing on the structures, such as stomach bubble and umbilical veins. As a consequence, it performs less superior than AD-GMP on estimating the extent of abdomen area. Likewise, GAP models perform poorly on localizing

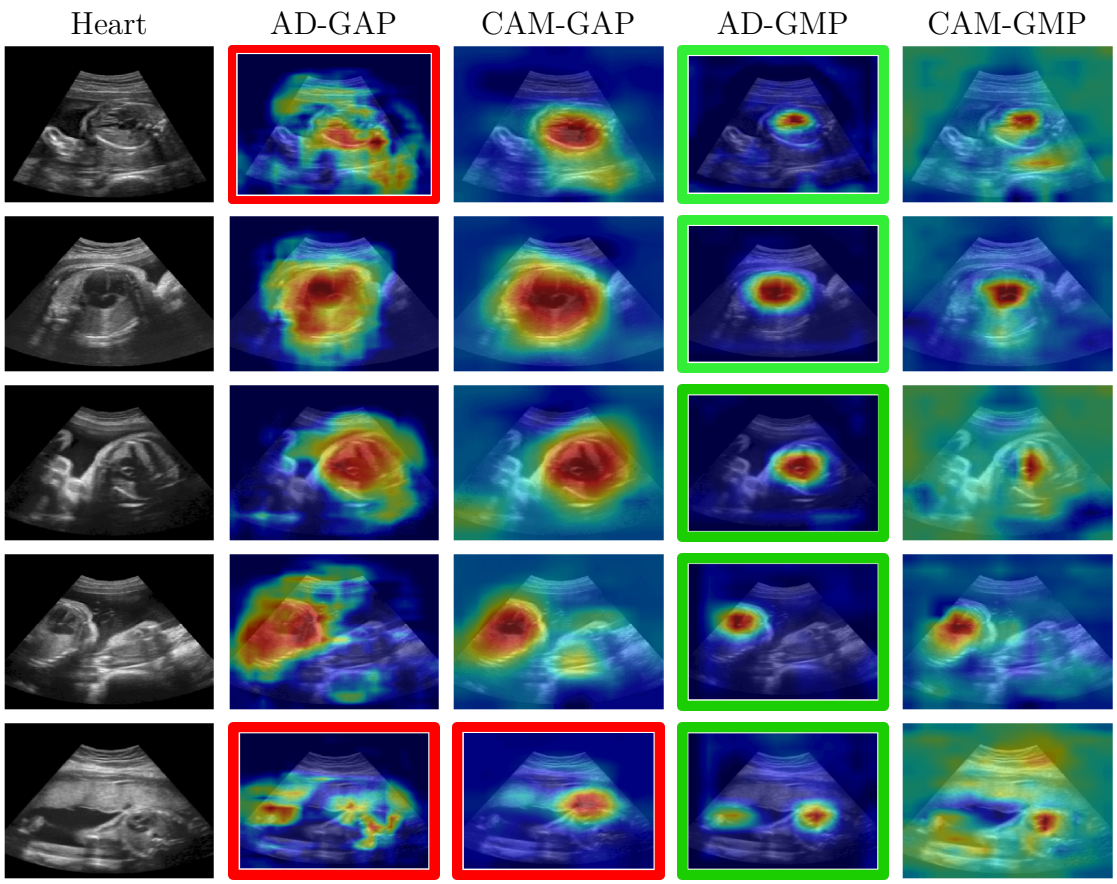


Figure V.12: Heart Class Activation Maps (CAMs). Heart examples of the high level CAMs generated from the four base-line models (see Section 5.1.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization.

the abdomen, as can be seen, it fails to capture the abdomen in most examples. Although, CAM-GAP captures the right region occasionally, it still fails to estimate the extent of abdomen that it very often oversegment the area of abdomen.

Fetal hearts are normally small which can be extremely difficult to recognize due to the reasons discussed earlier. However, it can been seen in Fig.V.12 that the AD-GMP and CAM-GMP models localize the fetal hearts precisely with high confidence, and more importantly it predicts the extent of fetal heart precisely. AD-GMP remains the best model that generates CAMs more discriminative than that of CAM-GMP. Although CAM-GAP spots the right location of fetal hearts in all examples, but as can be seen, it segments the whole abdomen area instead

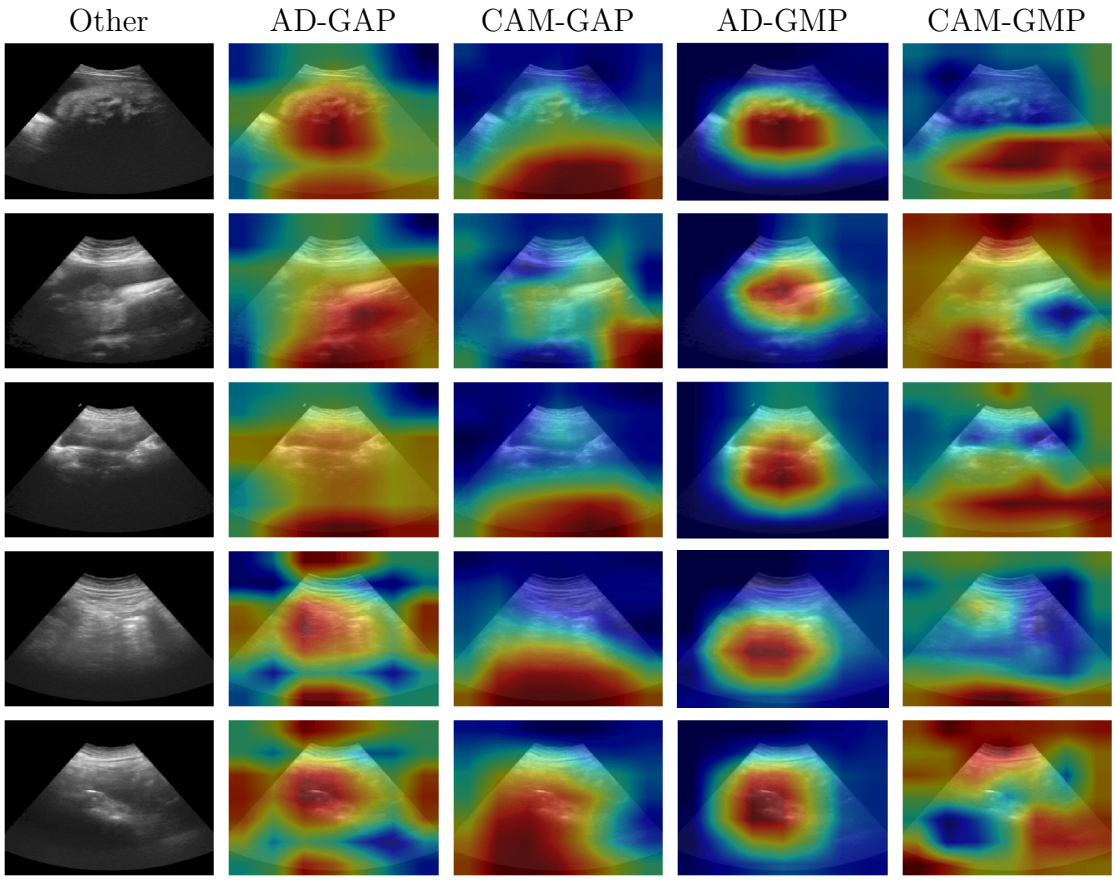


Figure V.13: Other Class Activation Maps (CAMs). Other examples (background frames) of the high level CAMs generated from the four base-line models (see Section III.A.). The maps highlight the discriminative regions for classification. Red boxes: failed in localization, Green boxes: the most discriminative localization.

of fetal hearts only. This also happens to the AD-GAP model that in some case it even failed to localize the fetal hearts (enclosed by red box).

In addition to the fetal structures of interest, the CAMs for the 'other' class i.e. background frames are as illustrated in Fig.V.13. It can be seen that the high confidence area sits right in the middle of field of view for AD-GMP model. In comparison, the activation is fairly random in other models that the high confidence areas are scattered and cover the whole frame.

Pattern Discovery

As a result of multi-scale learning, Fig.V.14 illustrates some examples of abdominal CAMs generated by AD-GMP model at different Conv feature levels. There is instance of the abdominal anatomy, such as stomach bubble (enclosed by blue box) or umbilical vein (enclosed by red box) in these examples, as shown in the first column. All the examples have shown the instance of both umbilical veins and stomach bubbles except for the second example (single instance of stomach bubble) and the last example (single instance of umbilical vein). The CAMs demonstrate the fine detection of the anatomies.

As can be seen, the Conv1 layer responds to edges but its corresponding CAMs are barely activated as the low level features are not discriminative at all. However, it should be noted that mid-low level i.e. Conv2 is already capable of capturing the umbilical veins, and mid-level Conv3 is able to detect the larger structure i.e. stomach bubbles. Conv4 captures both the umbilical veins and stomach bubbles with high confidence. Eventually, high-level Conv5 localizes well the entire fetal abdomen in each example.

Fig.V.15 compares the CAMs generated by AD-GMP and two-stream ConvNet[83] for fetal heart detection. It can be seen from that the two-stream model can be confused where the fetal heart is sometimes. As indicated in the second row, the CAMs can be not discriminative that there are multiple regions showing high confidence. And in some of the examples, such as the second and the fifth (counting from left), although the two stream network almost capture the fetal hearts, it shows a significant offset from their actual location. As a result, the two-stream model failed to localize the fetal heart in the examples (enclosed by red boxes). The reason for the poor localization has actually been discussed in Chapter V section 6 that the temporal network in [83] is too shallow to extract semantically-strong motion features, which makes it hard to discriminate the heart beating from other moving patterns. In all the failure cases, there is more or

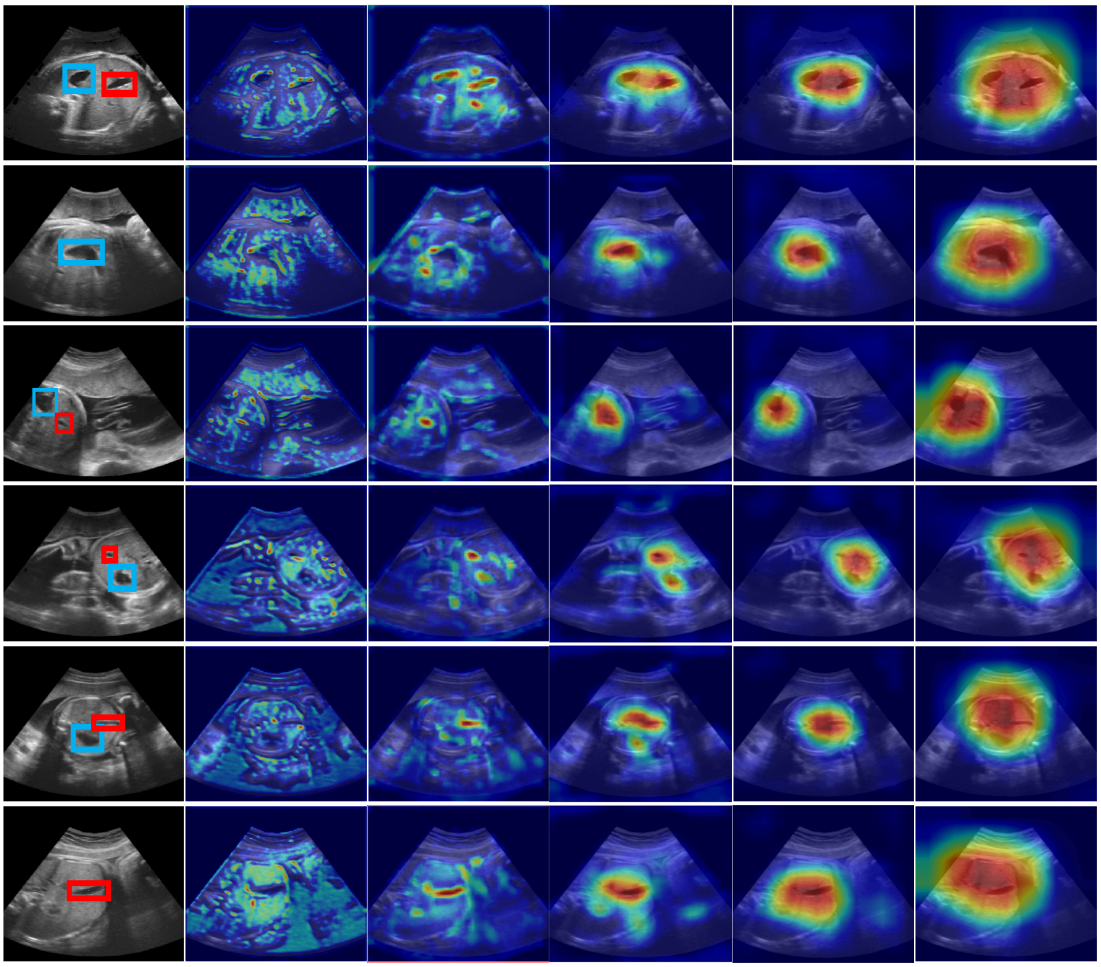


Figure V.14: Multi-scale CAMs for Fine-grained Detection. Fetal abdominal examples of CAMs learnt at different feature scale. Column-wise (left-right): Last frames in the abdominal video-clips, and CAMs (from low to high feature levels) at Conv1, Conv2, Conv3, Conv4, Conv5, respectively. Red boxes: umbilical vein, Blue boxes: stomach bubble.

less existence of the moving patterns that distract the localization of fetal hearts. In contrast, our AD-GMP model performs robustly in all cases (especially low contrast situation like the fourth example) and precisely captures the fetal heart in every example, as shown in the CAMs which are very discriminative highlighting where the fetal hearts are located. This demonstrates the superiority of exploring temporal context information from consecutive frames by learning spatio-temporal representations. In contrast to the two stream ConvNet, the proposed model learns the spatial-temporal representations directly from the ultrasound video, while the two stream model employs a pretrained temporal network that is untunable on

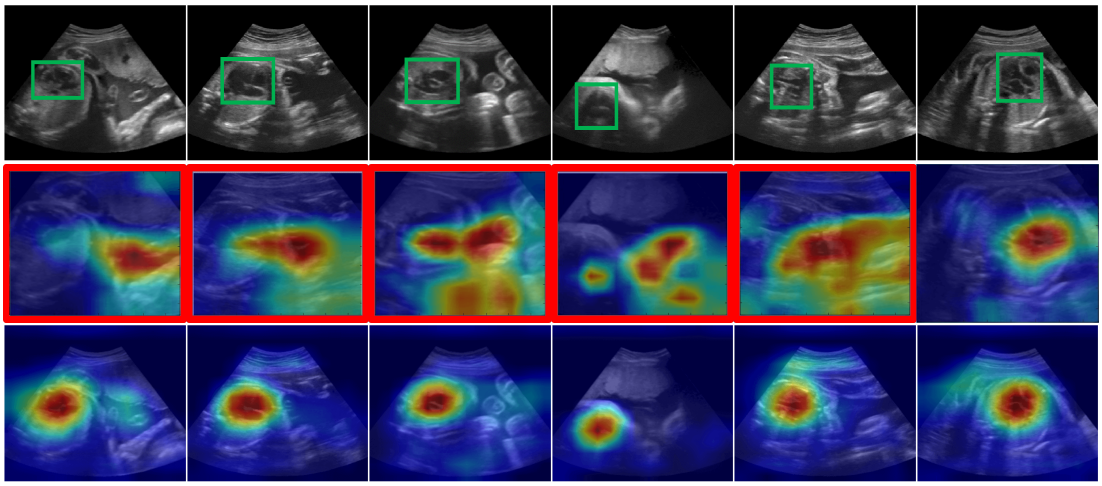


Figure V.15: Fetal Heart Detection. Fetal heart examples of CAMs generated from different models. Row-wise (top-down): examples of heart frames, **two-stream ConvNet** [83] and **AD-GMP**, respectively.

the ultrasound data (as discussed in Chapter V Section 6).

Quantitative Evaluation

Table.V.1 compares the classification performance of various models in terms of accuracy, precision, recall. It should be noted that GMP models generally achieved better classification performance for most metrics compared to GAP models. Particularly, CAM-GMP achieves the best performance in most of the metrics (highlighted in bold) among all models that, for instance, the class heart achieves the best accuracy of 96.1%, and the best precision and recall of 95.5% and 84.0%, respectively. It can also be noted that the attention gated models achieve very similar performance in every metric, and more importantly, the overall precision and recall are significantly improved in each category (especially the class heart) compared to the baseline models. It also compares to other state-of-the art models, like SonoNet, but it should be noted that both standard and non-standard frame are taken into account in this work, which is more challenging than the works [7] and [82] that only considers the standardized plane in each category and treats all non-standard frames as background. In spite of this, our proposed

Table V.1: CATEGORICAL CLASSIFICATION PERFORMANCE (%) OF VARIOUS MODELS ON THE TEST SET (5-fold cross validation). **FASP:**Fetal Abdominal Standard Plane. **FFVSP:**Fetal Four Chamber Standard Plane. **FHSP:**Fetal Heart Standard planes (all views).

Models	Fetal Skull			Fetal Abdomen			Fetal Heart		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
AD-GAP	98.8±0.8	81.3±2.8	73.8±5.6	92.0±2.4	77.4±4.5	68.6±5.9	92.5±5.6	73.6±3.8	68.0±6.0
AD-GMP	97.4±1.8	89.5±2.5	78.8±2.2	**97.4±3.9**	86.0±3.0	74.7±2.6	92.1±4.9	78.4±6.3	78.8±2.2
CAM-GAP	96.9±4.7	97.0±3.4	75.4±7.0	87.6±8.2	90.3±5.6	69.7±3.7	89.7±5.7	88.4±2.7	71.0±5.7
CAM-GMP	**99.5±0.4**	99.1±0.3	**85.1±6.0**	94.4±4.2	**95.5±1.9**	**82.4±6.1**	**96.1±5.1**	**95.5±1.8**	**84.0±4.8**
AG-AD-GAP	99.1±0.6	98.5±1.1	81.8±2.8	90.3±4.8	94.3±1.7	77.5±2.7	93.1±3.2	92.3±2.0	78.6±2.5
AG-AD-GMP	98.2±2.5	98.3±1.6	81.4±6.5	91.3±5.7	94.4±2.3	77.6±6.2	92.3±4.0	91.3±2.4	77.9±6.7
AG-CAM-GAP	99.1±0.8	98.8±0.4	81.9±4.9	90.9±5.6	93.6±1.3	76.9±3.9	91.0±5.6	88.7±1.3	76.9±4.6
AG-CAM-GMP	99.1±0.7	**99.4±0.5**	81.2±3.8	90.0±0.2	94.6±2.2	76.3±3.2	89.2±6.3	92.0±3.0	76.6±4.8
Two-stream [83]	97.4	91.1	70.3	93.7	90.6 FASP	71.5	90.3	85.3 FFVSP	79.2
RNN [87]	-	- Brain	-	90.8	74.8 Abdominal	74.7	86.7	77.0 FHSP	61.2
SonoNet [82]	-	88.0	97.0	-	93.0	90.0	-	72.8	66.0
AG-SonoNet [89]	-	98.4	98.6	-	96.3	96.1	-	80.2	83.3

model outperforms the state-of-the-arts, especially for the fetal heart detection, which our CAM-GMP (all views) achieves precision and recall of 95.5% and 84.0% respectively, compare to AG-Sononet (standardized heart planes only) that are 80.2% and 83.3%, and SonoNet that are 72.8% and 66.0%.

V.5.3 Visualization of Attention

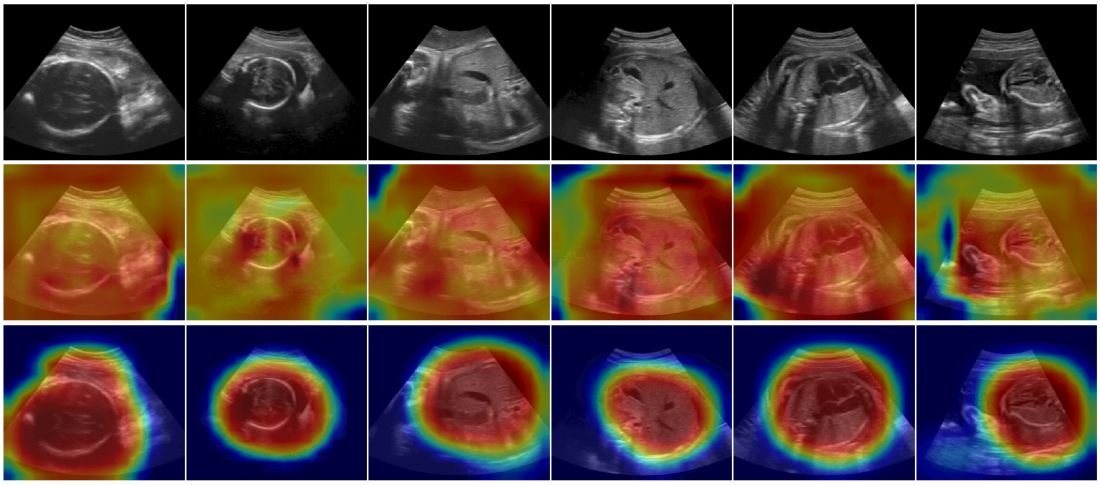


Figure V.16: Examples of CAMs generated by AD-GAP and attention gated AD-GAP. Row-wise (top-down): example frames, **AD-GAP** and **AG-AD-GAP**, respectively.

As discussed in last section, the AD-GAP model very often failed to localize the anatomy of interest. However, it can be found that attention helps to regularize the generation of CAMs. By introducing attention to the model, the CAMs generated become discriminative, which precisely capture the structures of interest, as illustrated in Fig.V.16. Again, because of the GAP, the AG-AD-GAP model overestimates the extent of small anatomy, like the fetal heart, which tends to capture the whole fetal abdomen instead of heart only.

Fig.V.17 gives examples of spatial-temporal attention captured by the AG-CAM-GMP model at different feature levels. It can be found that the **Conv3** layer captures a rich diversity of temporal dynamics of small anatomies, for instance, the appearing and disappearing of umbilical veins, and the periodic up-down movement of atrioventricular valves. A very tricky heart example (see the last column), the

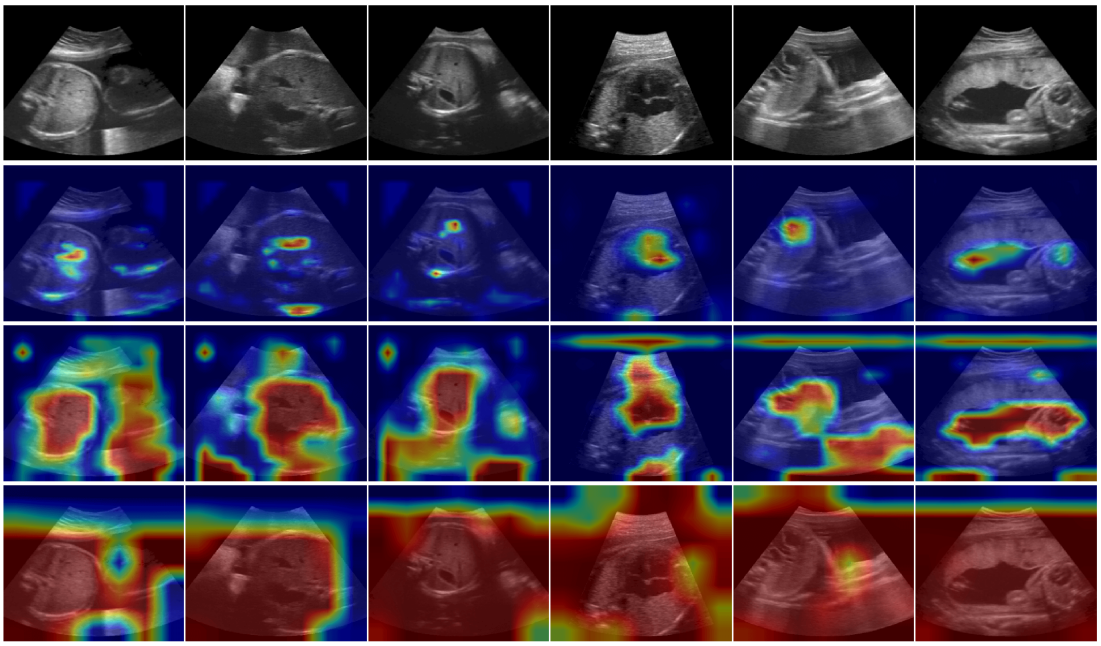


Figure V.17: Spatial-temporal Attention. **Row 1**: examples videos. **Row 2 to 4**: attention maps at **Conv3 to Conv5** levels, respectively. Best viewed in Adobe Reader. All videos should play automatically.

fetal abdomen is very small so that the neural network is struggling to find where the heart is. At the beginning, the attention is distracted by some unrecognizable moving patterns on the left-hand side of the frame, but it is progressively corrected towards end of the video, and eventually localize the fetal heart. Likewise, the attention is fairly random initially at **Conv4**, and because of the extremely large receptive field, **Conv5** begin with paying attention on the whole frame. With accumulation of contextual information, it learns to smoothly and progressively move to the anatomy of interest, and finally localize them. It should be noted that **Conv4** finally is more likely to focus on the anatomy inside the abdomen, such as the heart, stomach bubble and umbilical vein, and **Conv5** tends to focus on the fetal abdomen.

Figure.V.18 compares the attention learnt at multiple temporal scales, in which each row corresponds to one test video of the three anatomies of interest, sampled with the different temporal size of 3, 5 and 10 frames, respectively. The first column shows the attention learnt with a temporal window of 3 frames. It can be found that three frames is too short to capture the whole extent of the anatomy of interest,

like the skull and abdomen. As a result, the hot blobs on the corresponding CAMs are slightly off the center of the target anatomy. And it should also be noted from the third row that 3 frames attention failed to focus on the fetal heart, however, 5 and 10 frames learn to recognize the fetal heart successfully. Moreover, the CAMs generated at 5 frames correct the mis-localization (see the fourth column). Although the 10 frames managed to learn the attention, it can be found that it tends to over-segment the target anatomy. As a consequence, the resulting CAMs are not as discriminative as those of 3 and 5 frames, and this causes poorer localization.

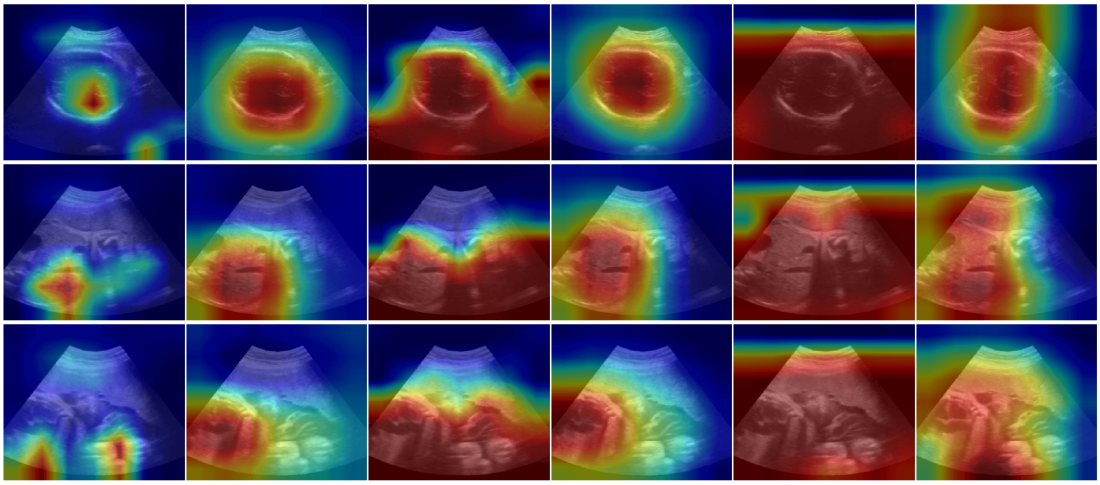


Figure V.18: Effect of temporal size. **Row 1 to 3**: Examples of skull, abdomen, and heart videos, respectively. **Column 1, 3 and 5**: attention learnt with a temporal scale of **3, 5 and 10** frames, respectively. **Column 2, 4 and 6** are CAMs generated at the last frame of the videos. Best viewed in Adobe Reader. All videos should play automatically.

V.5.4 Evaluation of Localization Performance

In order to evaluate localization performance quantitatively, a bounding box needs to be generated and its associated anatomy of interest. To generate a bounding box from the CAMs, a simple technique is used to segment the confidence maps. The regions were segmented of which the value is above a certain threshold of the max value of the CAMs. In our experiments, thresholds are chosen, ranging from 10% to 90% with an incremental of 10% (nine thresholds in total). The thresholds

are applied to both the **Conv4** and **Conv5** CAMs, then a bounding box is fitted to the largest connected component in the segmentation map.

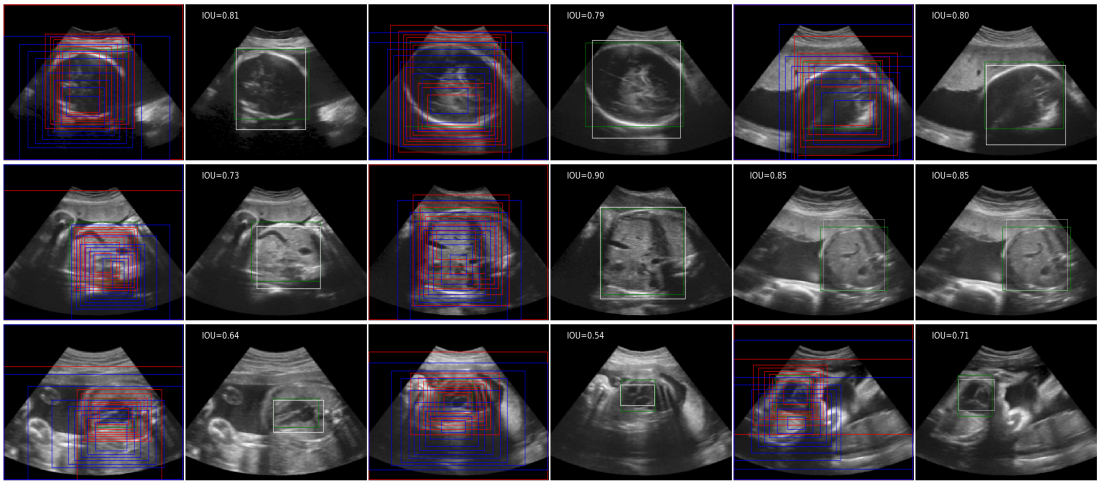


Figure V.19: Examples of predicted bounding boxes of target anatomy. **Row 1 to 3**: **Skull**, **Abdomen**, and **Heart** examples, respectively. **Red and blue boxes** are generated from **Conv4 and Conv5 CAMs** with different thresholds. **White boxes** are the predicted boxes that achieve the best **Intersection of Union (IOU)** against the **ground truth boxes (green boxes)**.

Fig.V.19 illustrates the generated bounding boxes–the red and blue ones are obtained from **Conv4** and **Conv5** CAMs respectively with different thresholds. Then, the Intersection Over Union (IOU) is measured between the generated boxes and the ground truth box (green boxes) to find the box that achieves the highest IOU i.e. the white boxes. The localization metrics, such as the categorical mean IOU and detection rate, are then evaluated based on the results of the IOU measurements.

Figure.V.20 compares the categorical mean IOU obtained from the four baseline models. It can be found that the 'other' category always achieves the highest IOU. This is because it consists of mostly background frames that the activation, as can be seen from the CAMs in Fig.V.13, is fairly random and large in terms of extent so that the generated boxes are very often as large as the ground truth boxes i.e. the entire frame. In contrast, the 'heart' category attains the lowest IOU in each model because the heart is normally much smaller than skull and abdomen which makes it difficult to achieve an IOU as good as the other categories (the

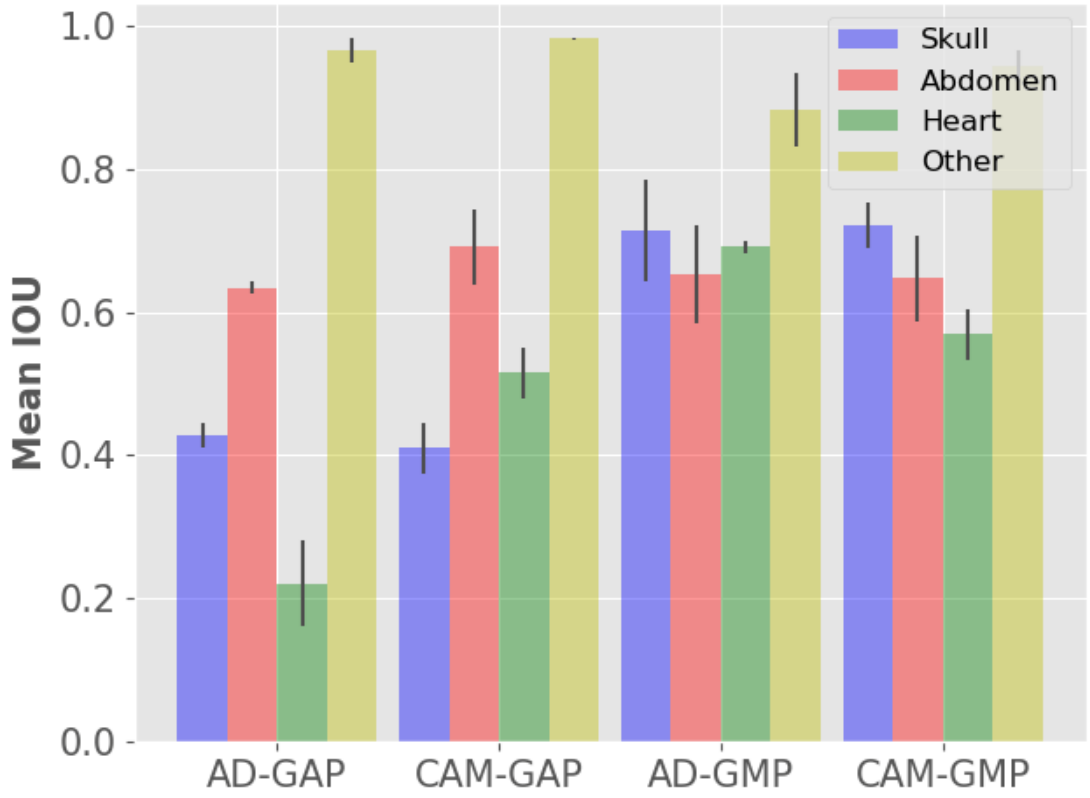
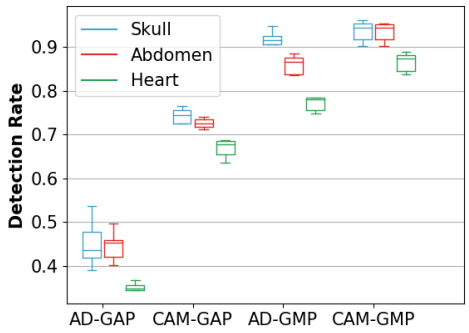


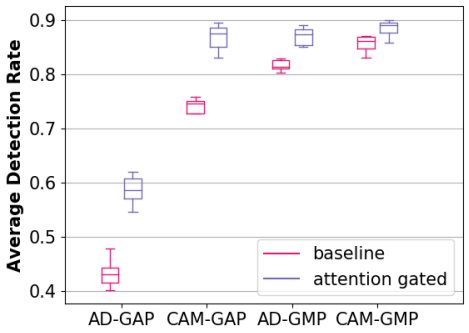
Figure V.20: Categorical Mean Intersection Over Union(IOU).

best is about 70% achieved by the model AD-GMP). It should also be noted that the GMP models i.e. AD-GMP and CAM-GMP achieve significantly better IOU in the 'heart' category compared to GAP models. This is because, as discussed in the previous sections, GAP models tend to over segment the heart, so instead of localizing the heart itself, they very often capture the entire abdominal plane resulting in the poorer heart IOU.

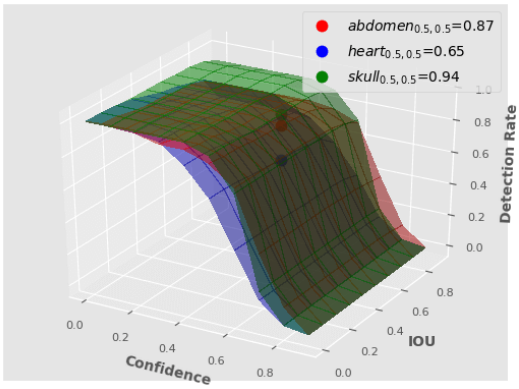
A simple metric based on the IOU and classification confidence was introduced to jointly evaluate the correctness of detection, namely the detection rate. Because of the bias of categorical IOU, a correct detection is counted if the IOU is above 50% of the maximum achievable IOU of its associated class, and it is a positive prediction with confidence above 80%. Figure.V.21a compares the categorical detection rate of the baseline models, in which it can be found the skull and abdomen category achieve comparable performance in each model, and slightly outperform the heart



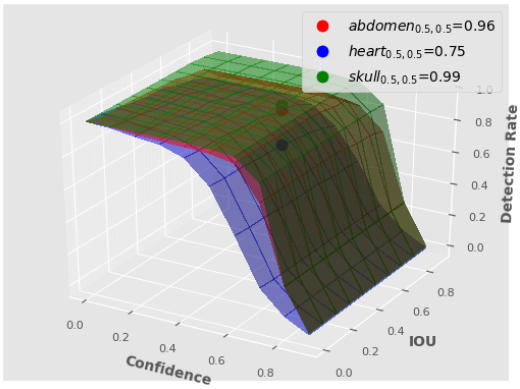
(a) Detection Rate



(b) Average Detection Rate



(c) CAM-GAP



(d) AG-CAM-GAP

Figure V.21: Quantitative results of localization (5-fold cross-validation). **(a)** Categorical detection rate is defined as $IOU > 0.5 \times max(IOU_{class})$ *and confidence* > 0.8 **(b)** Compare the average detection rate between the baseline and attention-gated models. **(c)** and **(d)** Plot detection rate against a range of IOU and confidence thresholds, for the model **CAM-GAP** and its attention-gated version i.e. **AG-CAM-GAP**.

category. The CAM-GMP model achieves the best detection rate of 0.94, 0.93 and 0.85 (median) in the category of skull, abdomen and heart, respectively. In Fig.V.21b, it should be noted that attention helps to improve the localization performance of the baseline models. Particularly, there is a substantial improvement of the average detection rate (more than 10%) in the GAP models i.e. AD-GAP and CAM-GAP. Although CAMs model already perform very well but there is still a moderate increase of average detection rate after incorporating attention into the models. To show the improvement more intuitively, the comparison is broken down into categories by plotting the categorical detection rate against different IOU and confidence thresholds (both ranging from 0 to 0.9), as shown in Fig.V.21c

and Fig.V.21d. As can be seen, the surfaces of each category in Fig.V.21d is relatively more expanded towards to 1, and the detection rate at the mid-point i.e. 0.5 is increased by 9%, 10% and 5% in the category of abdomen, heart and skull respectively, compared to the baseline model CAM-GAP in Fig.V.21c.

V.5.5 Evaluation of running time

In practice, computation complexity of the proposed model was studied that two metrics were considered: 1) the number of floating-point operations (FLOPs); 2) frame rate (speed) on a single GPU (FPS). FLOPs consist of convolution (multiply-adds) operations in the backbone (CNN) which is about 3.82 GFLOPs; Recurrent and attention modules is about 1.16GFLOPs and 0.29GFLOPs (here only Conv4 and Conv5 being considered at inference time, as conv4 and conv5 CAMs are used for the localization task); classifiers take a fractional computation that can be neglected compared to the above modules. So in total the proposed baseline model computation is about 4.98 GFLOPs (attention gated models slightly higher at 5.27GFLOPs) that is about one third the computation complexity of VGG16 [56] (15.3GFLOPs) , and is comparable to SonoNet-32 [82]. Frame rate accounts for not only computation complexity but also data I/O, memory access cost etc. and bounding box generation (post-processing). Frame rates were measured on a Nvidia Geforce GTX 1080 GPU for classification and localisation combined that base-line models achieve approx. 43.2 FPS, attention-gated models are slightly slower approx. 37.5 FPS. Videos used for evaluation were recorded at 30 FPS so all experimented models achieve real-time performance.

V.6 Discussion and Conclusion

A weakly supervised spatial-temporal neural network has been presented for detection of multiple fetal structures in free-hand ultrasound videos. In contrast to recent works, such as [82] and [89], in which they only learn to detect the standardized

fetal anatomical planes, and treat all non-standard planes as background class. This work values the importance of non-standard anatomical planes as they introduce more variability to the training data so as to learn a more expressive representation, and actually detection is learnt dominantly from the non-standard cases (especially the abdominal and heart planes). A flexible and computationally efficient soft attention-gated recurrent module was proposed which can be plugged-in any feature level of a ConvNet. As result of multi-scale learning, the model has demonstrated its capability of learning a rich diversity of spatial-temporal patterns at different feature levels. Moreover, class activation maps were evaluated qualitatively that GAP models were found being prone to over-segment the extent of structures of interest, especially the small anatomy like fetal heart, in contrast, GMP baseline models (both AD-GMP and CAM-GMP) are already good enough to learn a discriminative activation map that specific to the target anatomy. The classification performance was evaluated in terms of accuracy, precision and recall and to compare between models, the weighted average (over the categories skull, abdomen and heart) of the metrics was taken into account and the **CAM-GMP** achieved the best results except for classification precision as summarised in Table V.2. The localization performance was evaluated in terms of IOU and detection rate (accounting for both the IOU and the classification confidence) and the models were compared in terms of weighted average detection rate and mean IOU. The **AG-CAM-GMP** achieved the best localization results as summarised in Table.V.2. Moreover, Table.V.3 compares the attention-gated models and its associated baseline models, and the results (highlighted in bold) show that the soft attention mechanism helps to improve the classification precision and localization accuracy, and particularly the GAP models by a large margin. T-tests are carried out to corroborate the improvements are statically significant as indicated by the p-values. In addition, the attention mechanism also helps to regularize the learning of class activation mapping in the GAP models, as shown earlier in Fig.V.16.

	Classification (%)			Localization (%)	
	Acurracy	Precision	Recall	Mean IOU	Average Detection Rate
Baseline	CAM-GMP (96.3±2.1)	CAM-GMP (95.0±0.8)	CAM-GMP (82.3±2.9)	CAM-GMP (66.4±2.2)	CAM-GMP (85.7±1.6)
Attention Gated	AG-CAM-GMP (94.6±1.9)	AG-CAM-GMP (96.8±1.0)	AG-CAM-GMP (78.4±2.4)	AG-CAM-GMP (69.1±2.5)	AG-CAM-GMP (88.5±1.9)
Overall	CAM-GMP	AG-CAM-GMP	CAM-GMP	AG-CAM-GMP	AG-CAM-GMP

Table V.2: Summary of the best models in terms of different metrics. In terms of **classification**, CAM-GMP achieved the best results among the baseline models; AG-CAM-GMP achieved the best results among the attention gated models; Overall, CAM-GMP achieved the best results except for 'Precision' that AG-CAM-GMP achieved the best classification precision. In terms of **localization**, CAM-GMP achieved the best results among the baseline models so does their attention gated versions achieved the best results among attention gated models; Overall, AG-CAM-GMP achieved the best localization performance.

Table V.3: t-test for corroborating the improvement of **Average Classification Precision** and **Localization Accuracy (Average Detection Rate)** are statistically significant. The null hypothesis is that two sets of results (baseline models VS. attention gated models) come from independent normal distributions with equal means but unknown variances. The alternative hypothesis is that the two sets of results come from independent normal distributions with unequal means. **h (hypothesis test result):**-h is 1 if the test rejects the null hypothesis at the 5% significance level, and 0 otherwise. **p-value:**-a scalar in the range [0,1] is the probability of observing a test statistic as extreme as, or more extreme than, the observed value under the null hypothesis. p-value$<$ 5% results in rejection of the null hypothesis. **Classification Precision:** weighted average (over the classes skull, abdomen and heart) classification precision (5-fold cross validation). **Localization Accuracy:** weighted average (over the classes skull, abdomen and heart) detection rate (5-fold cross validation).

	Classification Precision (%)	p-value	h	Localization Accuracy (%)	p-value	h
AD-GAP	74.5±1.3	6.8×10^{-8}	1	43.4±2.9	3.8×10^{-5}	1
AG-AD-GAP	**94.0±1.9**			**58.7±2.9**		
AD-GMP	82.4±3.8	1.2×10^{-4}	1	73.3±3.2	8.9×10^{-5}	1
AG-AD-GMP	**94.7±1.4**			**86.8±2.7**		
CAM-GAP	90.2±2.5	0.013	1	81.7±1.1	3.4×10^{-4}	1
AG-CAM-GAP	**93.9±1.6**			**87.1±1.7**		
CAM-GMP	95.0±0.8	0.012	1	85.7±1.6	0.027	1
AG-CAM-GMP	**96.8±1.0**			**88.5±1.9**		

Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.

All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.

— Johann Wolfgang von Goethe [9]

VI

Summary and Future Work

Contents

VI.1 Contributions and Limitations 156
VI.2 Directions for future work 159
VI.2.1 Portable ultrasound for point-of-care 159
VI.2.2 Learning and Describing Actions 161
VI.2.3 Automation with Reinforcement Learning 162

In summary, we have been addressing the challenge of pattern recognition from fetal ultrasound videos acquired in limited-resources: (1) we presented a sufficiently simple scanning protocol, which non-expert sonographers can get on hand easily. Without standardization, there is a substantial variability in terms of the appearance of key anatomical structures. So the challenge here is to model the variability with limited amount of data; (2) Instead of annotating the video data explicitly (e.g. bounding box or segmentation mask), we did image-level annotation that extracts the frames containing anatomical structure of interest. So another challenge we have been addressing here is to learn fine-grained detection with the weak annotation.

In this thesis, we have demonstrated the feasibility of addressing the challenges

with deep learning. We have presented a number of deep learning models to automatically detect the fetal skull, abdomen and heart. This chapter summarizes the main contributions in this thesis and discusses the current limitations and potential opportunities for future improvements.

VI.1 Contributions and Limitations

In Chapter III, state-of-the-art CNNs have been investigated for fetal ultrasound image classification. We have demonstrated that the state-of-the-art CNNs should be tailored to learn on a much smaller ultrasound dataset (compare to natural image data) from scratch without overfitting. The key is to augment datasets properly and regularize the training carefully. We have also demonstrated optimization with batch normalization can actually accelerate the convergence of learning but there is no evidence showing it improves the generality of models. In addition, we have also demonstrated the cross-domain generality of CNNs that the distribution of low level features are actually similar between the natural image and ultrasound image, and it is a must to fine-tune the mid-high level feature representations on ultrasound data to have a decent classification performance. Furthermore, the trained classification CNNs have been visualized in both feature and image space for diagnosis purpose.

Although we have investigated transfer learning in this thesis, it is still an active area of research area, especially the cross-domain transfer learning, there is lack of evidence explaining how transfer learning works. A potential improvement should think about visualizing the adaptation (1) qualitatively, showing the evolution of patterns (2) quantitatively, showing the change of feature distributions over the course of adaptation.

In Chapter IV, video sequences that contain the fetal heart are further analyzed and a novel two-stream CNN model has been presented for automatically detecting and characterizing a fetal heartbeat. We have demonstrated the importance of incorporating motion features into representation learning for detecting the fetal

heart. Compare to the standard CNNs in Chapter III, we have made several distinctions in the proposed model: (1) it separates the encoding of motion and appearance representation; (2) it is a fully convolutional architecture that works on arbitrary size of input. Because of this, scale invariancy is able to be learned via a multi-scale training scheme; (3) kernels are rotated to make the representations rotation-invariant which is important for predicting the orientation of fetal heart. The classification results demonstrate that the proposed two-stream CNN outperforms the best standard CNN (in Chapter III) by a significant margin (accuracy 85.5% versus 74.9%), in terms of heart identification.

A potential limitation is the model was designed to be relatively shallow accounting for the limited computation resources (trained on CPU). This could compromise the discriminability of the network, especially the temporal network actually learned a motion detector rather than a motion discriminator. This could be improved by: (1) gathering more high quality fetal heart videos; 2) making a deeper network simultaneously; (3) and continuous learning on recognizing the cardiac activities, which is particularly important to draw attention of the network to learn motion features that relevant to the heart. In addition, because our data is acquired in a limited-resources setting, it is lack of the information (ground truth) of probe moving. As has been demonstrated, our proposed model actually captures both the global motion (caused by probe moving) and local motion (caused by heart beating). However, because lack of the ground truth, it is impossible to validate the motion of probe. A potential solution here would be mounting an accelerometer or setting up an optical tracking of the probe which allows to acquire the motion information of probe (orientation and magnitude). With this data, the motion of probe can be learned and validated given input ultrasound sequences. This is of great importance, in terms of heartbeat characterization that the true motion of the cardiac valves can be revealed by correcting the deviated local motion (valves moving, as described earlier in Fig.IV.14) with the validated global motion (probe moving).

In **Chapter V**, a deep learning based spatio-temporal model was proposed for detection and localization of the fetal skull, abdomen and heart in video sequences. Several distinctions have been made to the proposed model that (1) Instead of matrix multiplication, recurrent operation is implemented with convolution which saves the trainable parameters and computations; (2) Because of convolution, we are able to apply the recurrent module on a finer-resolution to learn temporal dynamics in mid-low level CNN representations; (3) the recurrent module is designed to be attention-gated so that localization of structures is attention-guided. With the proposed model, we have achieved several contributions in fetal ultrasound recognition: (1) to the best of our knowledge, we are the first to demonstrate what fetal patterns are learned at different CNN feature levels; (2) we are the first to visualize the temporal dynamics in the intermediate CNN representations; (3) we are the first to investigate the effect of global pooling with respect to the class activation mapping in medical image analysis. Furthermore, we have also demonstrated the usability of the non-standard fetal ultrasound planes that rich patterns have been discovered from the non-standard scans.

Despite the achievements, we have only explored the anatomy detection over a short temporal window. Future work will explore the importance of modelling long-term dependency for prediction of fetus presentation which needs to be determined by the appearing order of anatomies of interest over a longer temporal window. Modelling long-term dependencies in a video can be challenging that (1) gradient vanishing becomes prominent when interrogating through a lone video sequences and a practical solution is to incorporating gates into RNN to control the information flwoing through the hidden neurons, such as LSTM and GRU. This thesis has only studied the superiority of modelling spatial-temporal representations with LSTM and gradient vanishing is not a concern in this thesis but it may be important to be studied in the future when modelling long video sequences; (2) LSTMs and GRUs are still not power enough to learn long-term dependencies from sequences because they have only feedforward connections, which is different from the biological neural system that is typically composed of both feedforward and feedback connections. A

potential solution is to introduce the feedback connections to RNNs, for example, [125] introduces loop connection and processor-sharing mechanism to the network consisting of several processors, each of which is a GRU while associated with multiple groups of hidden states. All processors are loop connected to mimic the brain's feedforward and feedback connections, in which they are shared across multiple pathways in the loop connection. (3) gradient descent faces an increasing difficulty as the duration of the dependencies to be captured increases. [126] suggests to use alternative optimization approaches, such as multi-grid random search and discrete error propagation. In addition, we have demonstrated the effect of temporal scale on localizing the anatomies of interest. However, one limitation is our computation graph is fixed so that we can only train separate models for evaluating the effect of different temporal scales. An improvement would expect to build a dynamic computation graph that enables a multi-temporal scale learning so that we could train and test our model with arbitrary length of video sequences.

VI.2 Directions for future work

VI.2.1 Portable ultrasound for point-of-care

Emerging advances in semiconductor technology have paved way for smaller ultrasound machines with acceptable image qualities. This has subsequently led to the development of portable ultrasound devices to plug into laptops and smart phones, facilitating the use of ultrasound for point-of-care in different scenes. Recently, a start-up company called Butterfly Network announced a brand new portable ultrasound probe, as shown in Fig.VI.1, which is able to connect with smart phone to perform a whole body ultrasound scan [127]. Compare to conventional ultrasound probes, this probe replaced the piezoelectrics with a microchip to generate vibrations, which is flexible to be tuned with a dynamic range of frequencies for imaging at different depth. It is also worth noting that this portable ultrasound probe is able to acquire images with a quality as good as those acquired by a regular ultrasound



Figure VI.1: Portable ultrasound probe compatible with smart phones.

probe. This also facilitates the development of machine learning guided image acquisition. However, the machine learning application is very limited at this stage that in the case of Butterfly Networks, they only have simple image analysis functionality, such as differentiating a high- and poor-quality image for body parts like the heart. It is still an open area of research that incorporates machine learning into data acquisition to optimize the approach of fetal screening (improve efficiency) and also to train novice sonographers, even to enable non-experts to carry out the fetal examination themselves in the future. As discussed in Section II, the use of ultrasound imaging is far less common in the developing worlds because of the limited medical resources (lack of care providers and qualified sonographers). This kind of smart, portable, affordable ultrasound probe would be a potential solution to provide better pregnancy care in the future. With a global adoption of smart phones, hospitals could distribute the probe to the patients so that they would not need to queue for days to be screened anymore. Instead, machine learning applications could guide them to perform a standardized scan and automatically record the key information and generate analytic reports for clinicians to make a decision.

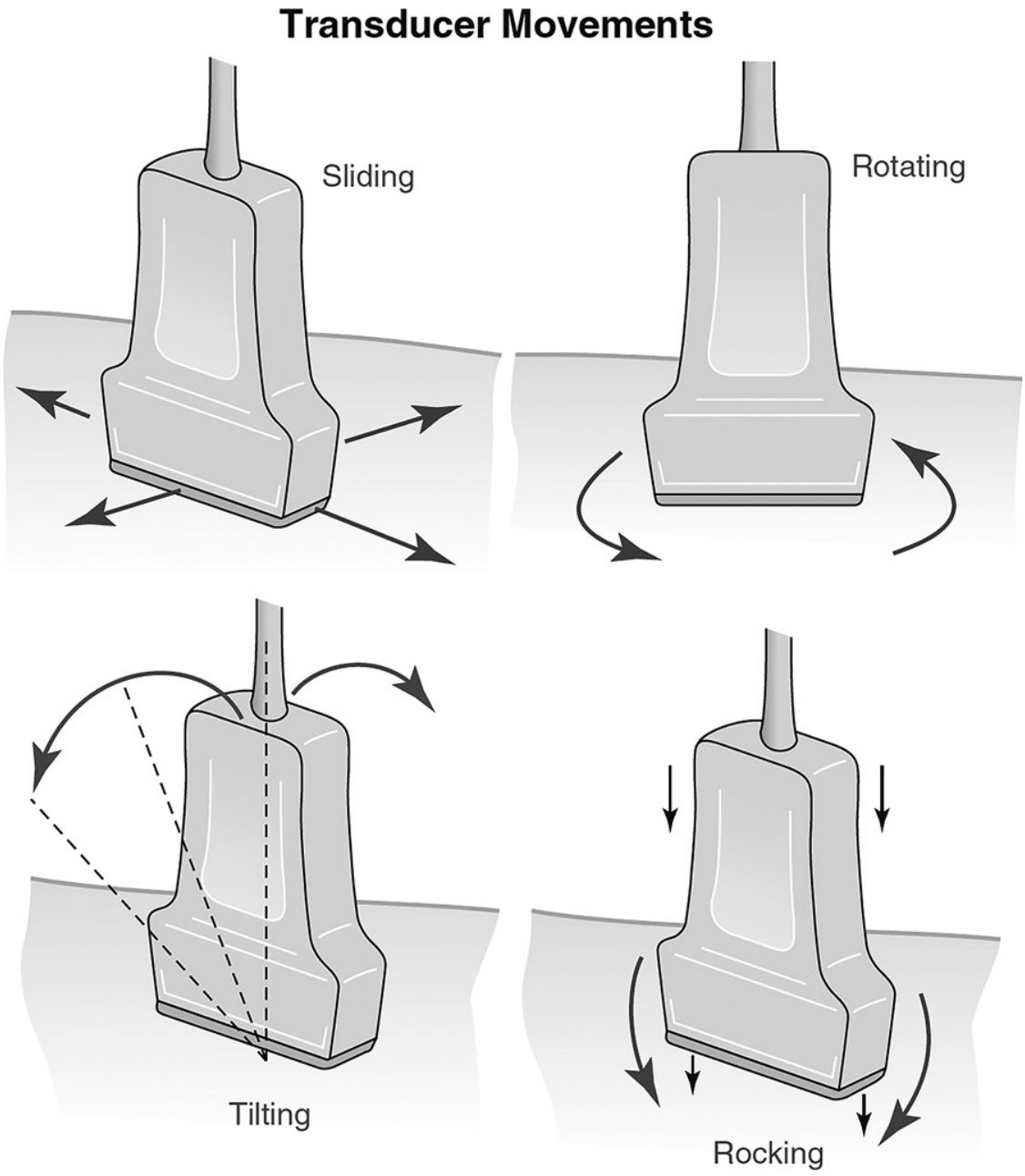


Figure VI.2: Ultrasound probe movement.

VI.2.2 Learning and Describing Actions

With advances in deep learning and GPUs, it has come to an unparalleled success of pattern recognition in fetal ultrasound images that now it is able to automatically differentiate standard and non-standard planes in real time [82] and also annotate the fetal structures of interest on-the-fly. We are working towards the harder

task of learning and describing the movement (action) of probe, to the best our knowledge, none of the relative research has been conducted in the past. In our research, we attempt to do such research, however, we do not have proper data to learn such information. As discussed earlier, our data used in this thesis was acquired with an extremely simple linear protocol that (1) there is no many actions involved in the scanning process. And (2) also there is no record (ground truth) of probe moving. These two limitations make it impossible to learn probe actions from our data. In order to carry out such research, it is required to discuss with an experienced sonographer to propose verbs describing the actions, such as "sliding", "rotating", "titling" and "rocking" (the probe), as illustrated in Fig.VI.2. Learning the probe actions can be a significant step towards the machine guided fetal screening. For example, when searching or fine-tuning for a certain fetal plane of interest, experts' and non-experts' actions (probe actions) can be recorded to train (1) a classification model on recognizing whether an acquisition is good or not (2) a regression model that scores the quality of screening. Such models take two sources of inputs i.e. image sequences and a series of probe actions that the quality of screening is not only assessed by the image quality but also the actions of moving probe. This will provide useful feedbacks for inexperienced sonographers to improve their skills. Moreover, reinforcement learning can be used to guide the fetal screening that the goal can be set to find a certain standard fetal plane as quick as possible then rewards can be measured when a series of probe actions are made. This provides a real-time feedback on how the probe should be moved according to the current environment and estimate how closed the current frame is to the target frame (e.g.a standard plane).

VI.2.3 Automation with Reinforcement Learning

Reinforcement learning (RL) has recently achieved a great success in gaming [128] and robotics [129], but has barely been researched in medical imaging. Reinforcement learning is an important step towards the automation of ultrasound fetal screening.

RL is based on the idea that an artificial agent learns by interacting with its environment [130]. In case of obstetric ultrasound, the agent could be a robotic arm holding the probe and it learns to interact with the environment i.e. video contents while scanning, as illustrated in Fig.VI.3. RL allows the agent to automatically determine the ideal behavior within a specific context that maximizes performance with respect to predefined measures. In case of obstetric ultrasound, the predefined measures could be, for instance, the similarity between the current frame and a target frame (e.g. standard diagnostic plane) and the time to localize the target. The RL agent is the decision-maker of the process and attempts to take an action (e.g. sliding, rotating the probe etc) recognized by the environment. It receives a reward or punishment from its environment depending on the action taken. The RL agents discover which actions bring more reward using exploration and exploitation [131] that the agent must make a trade-off between immediate and long-term return. It must also explore unseen states as well as the states which maximize its return by choosing what it already knows from the environment. Therefore, there needs to be a balance between exploration of unseen states and exploitation of familiar (rewarding) states. At the beginning of the learning process the RL agent does not have any knowledge about how promising taking different actions are [130]. It takes the various actions, and observes the results. After a while, the agent has explored many actions which bring the highest reward and gradually begins to exploit them. In fact, the agent acquires knowledge of the actions and eventually learns to perform the actions that are the most rewarding. During this process it tries to meet a certain goal relating to the state of the environment, in our case, we for example want to localize the standard diagnostic planes within the shortest time. Eventually, RL would learn an optimal policy of actions without any human data, guidance or knowledge. The most exciting thing about RL is it could learn a policy that is even superior than human's. This could also help humans to revise their policy, in case of fetal examination, hopefully it could help sonographers to find a more efficient approach of scan.

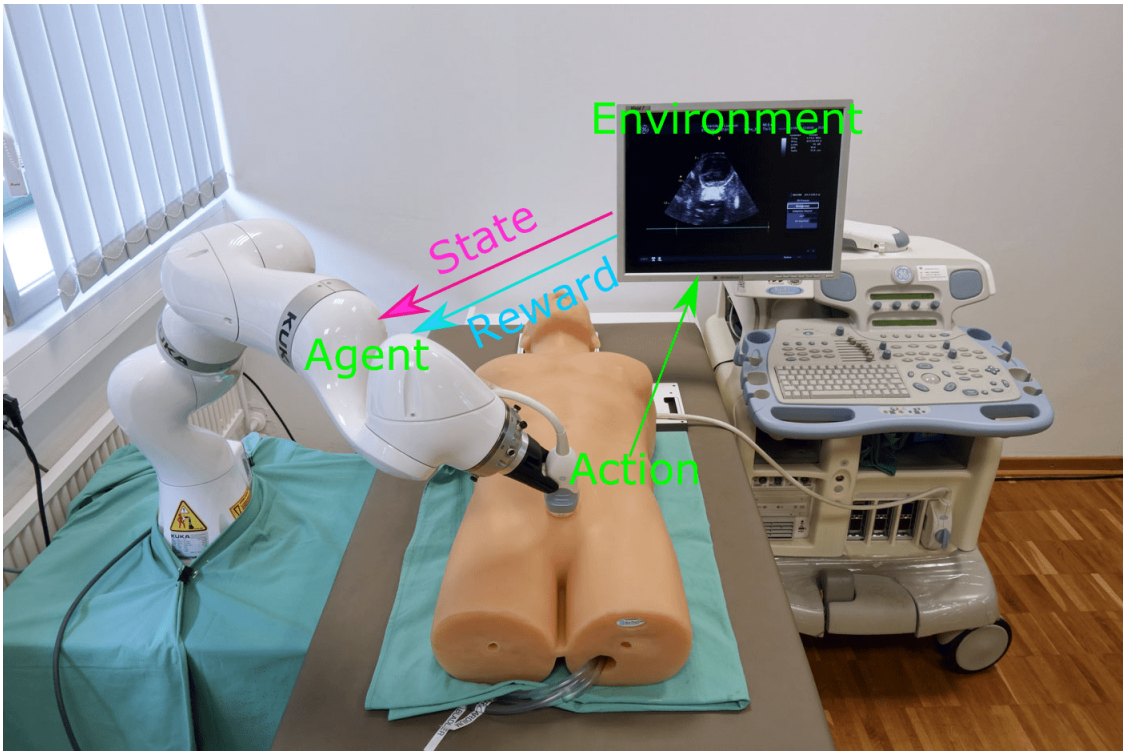


Figure VI.3: Reinforcement learning setting in Obstetric Ultrasound.

The first kind of intellectual and artistic personality belongs to the hedgehogs, the second to the foxes ...

— Sir Isaiah Berlin [132]

References

[1] Margaret B McNay and John E Fleming. "Forty years of obstetric ultrasound 1957-1997: from A-scope to three dimensions". In: *Ultrasound in medicine and biology* 25.1 (1999), pp. 3–56.

[2] Francis A Duck. "Is it safe to use diagnostic ultrasound during the first trimester?" In: *Ultrasound in Obstetrics and Gynecology* 13.6 (1999), pp. 385–388.

[3] Alfred Abuhamad et al. "Standardized six-step approach to the performance of the focused basic obstetric ultrasound examination". In: *American journal of perinatology* 2.01 (2016), pp. 090–098.

[4] Denise Tiran. "NICE guideline on antenatal care: routine care for the healthy pregnant woman—recommendations on the use of complementary therapies do not promote clinical excellence". In: *Complementary therapies in clinical practice* 2.11 (2005), pp. 127–129.

[5] LJ Salomon et al. "Practice guidelines for performance of the routine mid-trimester fetal ultrasound scan". In: *Ultrasound in Obstetrics and Gynecology* 37.1 (2011), pp. 116–126.

[6] Hao Chen et al. "Standard plane localization in fetal ultrasound via domain transferred deep neural networks". In: *IEEE journal of biomedical and health informatics* 19.5 (2015), pp. 1627–1636.

[7] Christian F Baumgartner et al. "Real-time standard scan plane detection and localisation in fetal ultrasound using fully convolutional neural networks". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (2016)*. Springer, pp. 203–211.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems (2012)*, pp. 1097–1105.

[9] Johann Wolfgang von Goethe. *Wilhelm Meisters Wanderjahre oder die Entsagenden*. de. Cotta, 1829.

[10] Leontine Alkema et al. "Global, regional, and national levels and trends in maternal mortality between 1990 and 2015, with scenario-based projections to 2030: a systematic analysis by the UN Maternal Mortality Estimation Inter-Agency Group". In: *The Lancet* 387.10017 (2016), pp. 462–474.

[11] Agustin Conde-Agudelo, José M Belizán, and Cristina Lammers. "Maternal-perinatal morbidity and mortality associated with adolescent pregnancy in Latin America: Cross-sectional study". In: *American journal of obstetrics and gynecology* 192.2 (2005), pp. 342–349.

[12] Agustin Conde-Agudelo, José M Belizán, and Cristina Lammers. "Maternal-perinatal morbidity and mortality associated with adolescent pregnancy in Latin America: Cross-sectional study". In: *American journal of obstetrics and gynecology* 192.2 (2005), pp. 342–349.

[13] Lale Say et al. "Global causes of maternal death: a WHO systematic analysis". In: *The Lancet Global Health* 2.6 (2014), e323–e333.

[14] Hannah Blencowe et al. "National, regional, and worldwide estimates of stillbirth rates in 2015, with trends from 2000: a systematic analysis". In: *The Lancet Global Health* 4.2 (2016), e98–e108.

[15] UNICEF WHO and C Mathers. "Global strategy for women's, children's and adolescents' health (2016-2030)". In: *Organization* 2016.9 (2017).

[16] Harald Ostensen. "Developing countries". In: *Ultrasound in Medicine and Biology* 26 (2000), S159–S161.

[17] Sachita Shah et al. "Perceived barriers in the use of ultrasound in developing countries". In: *Critical ultrasound journal* 7.1 (2015), p. 11.

[18] Clay Naomi. "Strategies for eliminating the sonographer shortage: recruitment, retention, and educational perspectives". In: *Journal of Diagnostic Medical Sonography* 20.6 (2004), pp. 408–413.

[19] Marveen Craig. "Sonographer Shortages: A Day Late and a Dollar Short?" In: *Journal of Diagnostic Medical Sonography* 19.4 (2003), pp. 261–271.

[20] Nigel Thomson. "Sonographer Workforce Survey Analysis". In: *Society of Radiographers* (2014), pp. 28–35.

[21] Julia Noble. "Towards engineering ultrasound image analysis solutions for resource constrained environments". Thesis. 2016.

[22] Melissa I March, Steven L Warsof, and Suneet P Chauhan. "Fetal biometry: relevance in obstetrical practice". In: *Clinical obstetrics and gynecology* 55.1 (2012), pp. 281–287.

[23] AT Papageorghiou et al. "International standards for early fetal size and pregnancy dating based on ultrasound measurement of crown–rump length in the first trimester of pregnancy". In: *Ultrasound in Obstetrics and Gynecology* 44.6 (2014), pp. 641–648.

[24] Steven R Goldstein. "Embryonic ultrasonographic measurements: crown-rump length revisited". In: *American Journal of Obstetrics and Gynecology* 165.3 (1991), pp. 497–501.

[25] Aris T Papageorghiou et al. "International standards for fetal growth based on serial ultrasound measurements: the Fetal Growth Longitudinal Study of the INTERGROWTH-21st Project". In: *The Lancet* 384.9946 (2014), pp. 869–879.

[26] Juozas Kurmanavicius et al. "Fetal ultrasound biometry: 2. Abdomen and femur length reference values". In: *BJOG: An International Journal of Obstetrics and Gynaecology* 106.2 (1999), pp. 136–143.

[27] Durlin E Hickok et al. "The frequency of breech presentation by gestational age at birth: a large population-based study". In: *American Journal of Obstetrics and Gynecology* 166.3 (1992), pp. 851–852.

[28] Mary Cheng and Mary Hannah. "Breech delivery at term: a critical review of the literature". In: *Obstetrics and gynecology* 82.4 Pt 1 (1993), pp. 605–618.

[29] Oya Demirci et al. "Pregnancy outcomes by mode of delivery among breech births". In: *Archives of gynecology and obstetrics* 285.2 (2012), pp. 297–303.

[30] Abhay Lodha et al. "Neonatal outcomes of preterm infants in breech presentation according to mode of birth in Canadian NICUs". In: *Postgraduate medical journal* (2011), pgmj. 2010.107532.

[31] Donna Kirwan. "NHS Fetal Anomaly Screening Programme". In: *18+ 0 to 20+ 6 Weeks Fetal Anomaly Scan National Standards and Guidance for England* (2010).

[32] I Sarris et al. "Standardization of fetal ultrasound biometry measurements: improving the quality and consistency of measurements". In: *Ultrasound in Obstetrics and Gynecology* 38.6 (2011), pp. 681–687.

[33] MJ Rijken et al. "Obstetric ultrasound scanning by local health workers in a refugee camp on the Thai–Burmese border". In: *Ultrasound in Obstetrics and Gynecology* 34.4 (2009), pp. 395–403.

[34] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[35] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision (2015)*, pp. 1026–1034.

[36] Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. "What is the best multi-stage architecture for object recognition?" In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, pp. 2146–2153.

[37] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml (2013)*. Vol. 30, p. 3.

[38] Ian J Goodfellow et al. "Maxout networks". In: *arXiv preprint arXiv:1302.4389* (2013).

[39] Meng Cai, Yongzhe Shi, and Jia Liu. "Deep maxout neural networks for speech recognition". In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pp. 291–296.

[40] Yi-Tong Zhou and Rama Chellappa. "Computation of optical flow using a neural network". In: *IEEE International Conference on Neural Networks*. Vol. 1998, pp. 71–78.

[41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), p. 533.

[42] D Randall Wilson and Tony R Martinez. "The general inefficiency of batch training for gradient descent learning". In: *Neural Networks* 16.10 (2003), pp. 1429–1451.

[43] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[44] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[45] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics (2010)*, pp. 249–256.

[46] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[47] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[48] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[49] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[50] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[51] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[52] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[53] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition (2014)*, pp. 1–9.

[54] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[55] Pierre Sermanet et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229* (2013).

[56] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[57] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition (2015)*, pp. 770–778.

[58] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision (2013)*. Springer, pp. 818–833.

[59] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. "Adaptive deconvolutional networks for mid and high level feature learning". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 2018–2025.

[60] Ken Chatfield et al. "Return of the devil in the details: Delving deep into convolutional nets". In: *arXiv preprint arXiv:1405.3531* (2014).

[61] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2014)*, pp. 1725–1732.

[62] Du Tran et al. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision (2014)*, pp. 4489–4497.

[63] Karen Simonyan and Andrew Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems (2014)*, pp. 568–576.

[64] Heng Wang and Cordelia Schmid. “Action recognition with improved trajectories”. In: *Proceedings of the IEEE international conference on computer vision (2013)*, pp. 3551–3558.

[65] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. “Convolutional two-stream network fusion for video action recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)*, pp. 1933–1941.

[66] Mohammad Ali Maraci et al. “Searching for structures of interest in an ultrasound video sequence”. In: *International Workshop on Machine Learning in Medical Imaging (2014)*. Springer, pp. 133–140.

[67] Mohammad Ali Maraci et al. “Fisher vector encoding for detecting objects of interest in ultrasound videos”. In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE, pp. 651–654.

[68] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[69] Mohammad Ali Maraci et al. “Object classification in an ultrasound video using lp-sift features”. In: *International MICCAI Workshop on Medical Computer Vision (2014)*. Springer, pp. 71–81.

[70] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[71] Josef Sivic and Andrew Zisserman. “Video Google: A text retrieval approach to object matching in videos”. In: *proc. ICCV (2003)*. IEEE, p. 1470.

[72] Florent Perronnin and Christopher Dance. “Fisher kernels on visual vocabularies for image categorization”. In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE, pp. 1–8.

[73] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. “Improving the fisher kernel for large-scale image classification”. In: *European conference on computer vision (2010)*. Springer, pp. 143–156.

[74] Relja Arandjelovic and Andrew Zisserman. “Three things everyone should know to improve object retrieval”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2911–2918.

[75] Gianfranco Doretto et al. “Dynamic textures”. In: *International Journal of Computer Vision* 51.2 (2003), pp. 91–109.

[76] SVN Vishwanathan, Alexander J Smola, and René Vidal. “Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes”. In: *International Journal of Computer Vision* 73.1 (2007), pp. 95–119.

[77] Roland Kwitt et al. “Localizing target structures in ultrasound video–a phantom study”. In: *Medical image analysis* 17.7 (2013), pp. 712–722.

[78] Mohammad Ali Maraci et al. “A framework for analysis of linear ultrasound videos to detect fetal presentation and heartbeat”. In: *Medical image analysis* 37 (2017), pp. 22–36.

[79] Christopher P Bridge, Christos Ioannou, and J Alison Noble. “Automated annotation and quantitative description of ultrasound videos of the fetal heart”. In: *Medical image analysis* 36 (2017), pp. 147–161.

[80] Antoni B Chan and Nuno Vasconcelos. “Layered dynamic textures”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.10 (2009), pp. 1862–1879.

[81] Y Gao, Mohammad Ali Maraci, and J Alison Noble. “Describing ultrasound video content using deep convolutional neural networks”. In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE, pp. 787–790.

[82] Christian F Baumgartner et al. “Sononet: real-time detection and localisation of fetal standard scan planes in freehand ultrasound”. In: *IEEE transactions on medical imaging* 36.11 (2017), pp. 2204–2215.

[83] Yuan Gao and J Alison Noble. “Detection and characterization of the fetal heartbeat in free-hand ultrasound sweeps with weakly-supervised two-streams convolutional networks”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (2017)*. Springer, pp. 305–313.

[84] Y Cai et al. “SonoEyeNet: Standardized fetal ultrasound plane detection informed by eye tracking”. In: *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*. IEEE, pp. 1475–1478.

[85] Hosuk Ryou et al. “Automated 3D Ultrasound Biometry Planes Extraction for First Trimester Fetal Assessment”. In: *International Workshop on Machine Learning in Medical Imaging (2016)*. Springer, pp. 196–204.

[86] 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* 47 (2018), pp. 127–139.

[87] Hao Chen et al. “Automatic fetal ultrasound standard plane detection using knowledge transferred recurrent neural networks”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (2015)*. Springer, pp. 507–514.

[88] Weilin Huang et al. “Temporal HeartNet: towards human-level automatic analysis of fetal cardiac screening video”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (2017)*. Springer, pp. 341–349.

[89] Jo Schlemper et al. “Attention-Gated Networks for Improving Ultrasound Scan Plane Detection”. In: *arXiv preprint arXiv:1804.05338* (2018).

[90] Maxime Oquab et al. “Is object localization for free?-weakly-supervised learning with convolutional neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)*, pp. 685–694.

[91] Bolei Zhou et al. “Learning deep features for discriminative localization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)*, pp. 2921–2929.

[92] Mingsheng Long et al. “Learning transferable features with deep adaptation networks”. In: *arXiv preprint arXiv:1502.02791* (2015).

[93] Ali Sharif Razavian et al. “CNN features off-the-shelf: an astounding baseline for recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (2014)*, pp. 806–813.

[94] Jeff Donahue et al. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *International conference on machine learning (2013)*, pp. 647–655.

[95] J Villar et al. “The objectives, design and implementation of the INTERGROWTH-21st Project”. In: *BJOG: An International Journal of Obstetrics and Gynaecology* 120 (2013), pp. 9–26.

[96] Jose Villar et al. “International standards for newborn weight, length, and head circumference by gestational age and sex: the Newborn Cross-Sectional Study of the INTERGROWTH-21st Project”. In: *The Lancet* 384.9946 (2014), pp. 857–868.

[97] International Society of Ultrasound in Obstetrics and Gynecology Education Committee. “Sonographic examination of the fetal central nervous system: guidelines for performing the’basic examination’and the’fetal neurosonogram’”. In: *Ultrasound in obstetrics and gynecology: the official journal of the International Society of Ultrasound in Obstetrics and Gynecology* 29.1 (2007), p. 109.

[98] Ian J Goodfellow et al. “Multi-digit number recognition from street view imagery using deep convolutional neural networks”. In: *arXiv preprint arXiv:1312.6082* (2013).

[99] Yoshua Bengio. “Deep learning of representations for unsupervised and transfer learning”. In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning (2012)*, pp. 17–36.

[100] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[101] Laurens van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[102] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. “Adaptive deconvolutional networks for mid and high level feature learning”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, pp. 2018–2025.

[103] Damien Teney and Martial Hebert. “Learning to extract motion from videos in convolutional neural networks”. In: *Asian Conference on Computer Vision (2016)*. Springer, pp. 412–428.

[104] JS Carvalho et al. “ISUOG Practice Guidelines (updated): sonographic screening examination of the fetal heart”. In: *Ultrasound in Obstetrics and Gynecology* 41.3 (2013), pp. 348–359.

[105] David J Heeger. “Model for the extraction of image flow”. In: *JOSA A* 4.8 (1987), pp. 1455–1471.

[106] Fabio Solari et al. “What can we expect from a V1-MT feedforward architecture for optical flow estimation?” In: *Signal Processing: Image Communication* 39 (2015), pp. 342–354.

[107] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *European conference on computer vision (2004)*. Springer, pp. 25–36.

[108] Thomas Brox and Jitendra Malik. “Large displacement optical flow: descriptor matching in variational motion estimation”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.3 (2011), pp. 500–513.

[109] Simon Baker et al. “A database and evaluation methodology for optical flow”. In: *International Journal of Computer Vision* 92.1 (2011), pp. 1–31.

[110] Daniel J Butler et al. “A naturalistic open source movie for optical flow evaluation”. In: *European Conference on Computer Vision (2012)*. Springer, pp. 611–625.

[111] Jost Tobias Springenberg et al. “Striving for simplicity: The all convolutional net”. In: *arXiv preprint arXiv:1412.6806* (2014).

[112] Ken Chatfield et al. “Return of the devil in the details: Delving deep into convolutional nets”. In: *arXiv preprint arXiv:1405.3531* (2014).

[113] Nicolas Ballas et al. “Delving deeper into convolutional networks for learning video representations”. In: *arXiv preprint arXiv:1511.06432* (2015).

[114] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[115] Kelvin Xu et al. “Show, attend and tell: Neural image caption generation with visual attention”. In: *International conference on machine learning (2015)*, pp. 2048–2057.

[116] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. “Unsupervised learning of video representations using lstms”. In: *International conference on machine learning (2015)*, pp. 843–852.

[117] Jeffrey Donahue et al. “Long-term recurrent convolutional networks for visual recognition and description”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (2014)*, pp. 2625–2634.

[118] Joe Yue-Hei Ng et al. “Beyond short snippets: Deep networks for video classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (2015)*, pp. 4694–4702.

[119] Min Lin, Qiang Chen, and Shuicheng Yan. “Network in network”. In: *arXiv preprint arXiv:1312.4400* (2013).

[120] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[121] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *Herbert Robbins Selected Papers (1951)*. Springer, 1985, pp. 102–109.

[122] Christian Darken, Joseph Chang, and John Moody. “Learning rate schedules for faster stochastic gradient search”. In: *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*. IEEE, pp. 3–12.

[123] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[124] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. "Neural networks for machine learning". In: *Coursera, video lectures* 264 (2012).

[125] Yemin Shi et al. "Learning long-term dependencies for action recognition with a biologically-inspired deep network". In: *Proceedings of the IEEE International Conference on Computer Vision (2017)*, pp. 716–725.

[126] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[127] Stephanie A Barnes and Gregory T Wing. *Handheld medical diagnostic ultrasound instrument*. Generic. 2002.

[128] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[129] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 3389–3396.

[130] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[131] Sebastian B Thrun. "Efficient exploration in reinforcement learning". In: (1992).

[132] Isaiah Berlin. *The Hedgehog and the Fox: An Essay on Tolstoy's View of History*. English. Ed. by Henry Hardy. 2nd. Princeton University Press, June 2013.