MACHINE LEARNING FOR NEURAL ACTIVITY VIDEO ANALYSIS AND FOR OBJECT TRACKING IN VIDEO

DISSERTATION

Submitted in Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Yilin Song

January 2018
MACHINE LEARNING FOR NEURAL ACTIVITY VIDEO ANALYSIS AND FOR OBJECT TRACKING IN VIDEO

DISSERTATION

Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY TANDON SCHOOL OF ENGINEERING

by

Yilin Song

January 2018

Approved:

______________________________
Department Head Signature

University ID#: __________________________
NetID#: __________________________
Approved by the Guidance Committee:

Major: Electrical Engineering

Yao Wang  
Professor of Electrical Engineering  
New York University

Jonathan Viventi  
Assistant Professor of Biomedical and Engineering  
Duke University

Anna Choromanska  
Assistant Professor of Electrical Engineering  
New York University

Yong Liu  
Professor of Electrical Engineering  
New York University
Microfilm or other copies of this dissertation are obtainable from

UMI Dissertation Publishing
ProQuest CSA
789 E. Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346


Vita

Yilin Song was born on August 13nd, 1988 at Dalian, China. He received his Bachelor of Engineering degree and master degree at Northeast University, China and New York University, United States, in 2011 and 2013 respectively. Since September 2013, he has been a Ph.D student at Department of Electrical and Computer Engineering in Tandon School of Engineering, New York University, Brooklyn, United State, under the joint supervision between Professor Yao Wang and Professor Jonathan Viventi.

From May 2015 to August 2015, he worked as intern at SRI international at NJ on robust tracking system with convolutional neural network and collaborative model for hand position estimation. From September 2014 to May 2017, he co-founded Wikkit Labs, which is a start up company targeting at computer vision applications located at Shenzhen China. He led several projects on large scale face detection, face recognition, pedestrian tracking and model compression. During his doctorate study, his research area includes neural activity prediction, object tracking, object detection and video content analysis.
Acknowledgements

First of all, I would like to take the chance to express my deepest appreciation to my adviser, Professor Yao Wang, whose advice, encouragement, enduring patience and constant support, give the successful outcome of this thesis. Her rigorous attitude towards details influences not only my Ph.D study but also my future career. I would like to thank Professor Jonathan Viventi for his invaluable input in numerous discussions and paper revision. I would also like to thank Professor Anna Choromanska and Professor Yong Liu for their precious time spent in reviewing this work and constructive advises during this development.

During my Ph.D study, I am blessed to work with Dr. Yuanyi Xue, Chenge Li, Ran Wang and Dr. Cagdas Bilen who participated in my research and gave valuable input. I am also grateful to other labmates: Dr. Xuan Zhao, Fanyi Duanmu, Shervin Minaee, Jen-wei Kuo, Dr. Eymen Kurdoglu, Anti-Chiang, Shihong Fang and Yuan Wang for the inspiring discussion with them.

I am also very lucky to work with Dr. Xiaoran Ning, Dr. Yuanyi Xue, Bo Yan, Dr. Xuan Zhao, Sixuan Liu and Chenge Li for three years building our startup. Although it has been a challenging journey, I learned quite a lot from this experience. I am also very grateful to Dr. Ajay Divakaran and Dr. Amir Tamrakar, both my mentors during the internship at SRI international. The knowledge and experiences I learned from them in the internship are of great help for my future career. I also wish to sincerely thank everybody not mentioned here, but contributed in one way or another to the thesis.

Ph.D study is an endured but a joyful journey, I thank my family and my girl friend, for their unconditioned love, support and encourage throughout this journey.
To my parents and Jiayi with affection
ABSTRACT

MACHINE LEARNING FOR NEURAL ACTIVITY VIDEO ANALYSIS AND FOR OBJECT TRACKING IN VIDEO

by

Yilin Song

Advisor: Prof. Yao Wang, Ph.D.
Co-advisor: Prof. Jonathan Viventi, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Electrical Engineering)

January 2018

For the past few years, flexible, active, multiplexed recording devices for high resolution recording over large, clinically relevant areas in the brain have been manufactured. While this technology has enabled a much higher-resolution view of the electrical activity of the brain, the analytical methods to process, categorize
and respond to the huge volumes of data produced by these devices have not yet been developed.

In the first part of this thesis, we propose an unsupervised learning framework for spike analysis, which reveals typical spike patterns of the \( \mu \)ECoG data. And we further explore using these patterns for seizure prediction and spike wavefront prediction. These methods have been applied to in-vivo feline seizure recordings and yielded promising results.

Being able to predict the neural signal in the near future from the current and previous observations has the potential to enable real-time responsive brain stimulation to suppress seizures. We explore two approaches for neural activity prediction using recurrent neural networks. Firstly, to exploit the multiple activity pattern clusters present in the signal, a multiple choice learning model is proposed. An ensemble-awareness loss is used to jointly solve the model assignment problem and error minimization problem. Secondly, to ensure an accurate prediction for a long time horizon, two multi-resolution networks are proposed. To overcome the blurring effect associated with video prediction in the pixel domain using standard mean square error (MSE) loss, energy based adversarial training is used to improve the long-term prediction.

We also looked at the problem of object tracking at the pixel level. We present a novel pixel-wise visual object tracking framework that can track any anonymous object in a noisy background. The framework consists of two submodels, a global attention model and a local segmentation model. The global model generates a region of interests (ROI) that the object may lie in the new frame based on the past object segmentation maps; while the local model segments ROI to detect pixels belonging to the object. Once the models are trained, there is no need to refine
them to track specific objects, making this method efficient compared to online learning approaches.
# Contents

Vita .......................................................... iv
Acknowledgements ................................. v
Abstract ...................................................... vii
List of Figures ........................................... xxi
List of Tables ............................................ xxiv

1 Introduction 1
  1.1 Motivation .............................................. 2
  1.2 Contribution ............................................. 4
  1.3 Outline .................................................. 6

2 Background on $\mu$ECoG data and seizure prediction 9
  2.1 Multiplexed, Active Electrode Arrays .................. 9
  2.2 Dataset .................................................. 10
  2.3 Preprocessing using graph filtering .................... 12

3 Spike Clustering and Seizure Prediction 17
  3.1 Spike Segmentation ...................................... 18
  3.2 Spike Clustering Through Manifold Learning .......... 21
    3.2.1 Delay Map and Wavefront Trajectory ............. 23
3.2.2 Raw Signal Correlation Map .................................. 24
3.2.3 Manifold Projection .............................................. 25
  3.2.3.1 Isomap .................................................. 25
3.2.4 Clustering Using the Dirichlet Process Mixture Model .... 27
3.2.5 Clustering Results Comparison .................................. 27
3.3 Wavefront Trajectory Prediction ................................... 29
  3.3.1 Trajectory Prediction with Mean-based Regularization ... 30
3.4 Seizure Detection and Prediction .................................. 33
  3.4.1 Using a Single Hidden Markov Models ....................... 34
  3.4.2 Using Multiple Hidden Markov Models plus a second stage
classifier (HMM+SVM) ........................................... 36
  3.4.3 Detection and Prediction Result ................................ 38
3.5 Conclusion ......................................................... 42

4 Neural Activity Prediction ........................................... 44
  4.1 Review on Video Prediction ....................................... 44
    4.1.1 LSTM .................................................. 46
    4.1.2 Convolutional LSTM ..................................... 47
    4.1.3 LSTM and ConvLSTM for Video Prediction ................. 47
  4.2 Diversity Encouraged Learning of Ensemble Models ............... 48
    4.2.1 Multiple choice learning (MCL) ............................. 49
    4.2.2 Training the LSTM ensemble using MCL .................... 51
    4.2.3 Model selection as classification ............................ 55
    4.2.4 Relationship between trained models with neural activity
patterns .......................................................... 56
  4.3 Long Term Prediction With Multiple Temporal Resolution ....... 58
4.3.1 Framework .............................................. 59
4.3.2 Generative Model ....................................... 60
4.3.3 Discriminative Model .................................... 61
4.3.4 Multi-resolution Representation ......................... 63
  4.3.4.1 Benchmark Network ................................. 63
  4.3.4.2 Multi-resolution LSTM ............................. 63
  4.3.4.3 LSTM with Multi-resolution layer ................... 65
4.3.5 Results ................................................ 67
4.4 Conclusion ............................................... 71

5 Pixel-wise Object Tracking ................................ 73
  5.1 Introduction ............................................. 74
  5.2 The overall framework ................................... 76
  5.3 Local Segmentation Network .............................. 78
    5.3.1 Framework .......................................... 78
    5.3.2 Memory Initialization ............................... 80
    5.3.3 Training .............................................. 83
    5.3.4 Comparison and analysis ............................ 85
  5.4 Global Attention Network ................................. 86
  5.5 Experiment ............................................... 88
    5.5.1 Iterative optimization ............................... 88
    5.5.2 Time complexity .................................... 90
    5.5.3 Quantitative analysis ............................... 91

6 Conclusion .................................................. 94
  6.1 Summary of Main Contribution ............................ 94
6.2 Future Research .................................................... 96

6.2.1 Model Compression for Real Time Application .......... 96

6.2.2 Long Term Prediction with External Memory Network ... 97
List of Figures

2.1 Photograph of a 360 channel, high density neural electrode array used in a feline model of epilepsy. The electrode array was placed on the surface of visual cortex. The electrode size and spacing was 300 $\mu$m x 300 $\mu$m and 500 $\mu$m, respectively. .......................... 10

2.2 Comparison of graph filtering and Gaussian smoothing. Subplots a,c are results using a 5 by 5 Gaussian smoothing kernel for spatial filtering, and subplots b,d are graph filtering results using 80 dominant graph coefficients. For Gaussian filtering bad channels are human labelled and interpolated with Gaussian distance kernel from good channels within a local region of 5 by 5. The results b,d preserved the original pattern as well as smoothed the signal. .... 13

3.1 Processing steps for clustering spike patterns. ....................... 18
3.2 Examples of spike segmentation. Subfigure a is for a spike with a spiral motion. Subfigure b is for a spike with a planar motion. Each subfigure shows the segmented volume in transparent blue overlaid with the μECoG signals captured at different times, with vertical axis (time axis) corresponding to frame number, while the other two axis represent the spatial arrangement of μECoG electrodes.

3.3 Comparison of fixed length segmentation [76] (subfigure a,c) against using region-growing based variable length segmentation (subfigure b,d). With fixed length segmentation, the entire 3D cube formed by extending from a selected center frame to both past and future frames by a fixed length. The high intensity voxels in all figures are indicated by transparent color. In subfigure a,c, all voxels include within each spike segment, whereas in subfigure b,d only the high intensity voxels are included within each spike segment. Subfigure a shows a falsely detected segment by the fixed length method, which includes a truncation on top and includes many extra voxels (mainly on the bottom) that have low intensities. The detected spike segment for the similar time duration by the proposed method is shown in subfigure b, which includes all high intensity voxels that are connected. Subfigure c shows a detected spike segment that includes three separate true spikes, which are correctly separated by the region-growing method show in subfigure d.
3.4 Examples of extracted features. Subfigures a and b are the delay and trajectory features of the 9 spikes closest to the centroids of the 9 dominant clusters identified by Isomap embedding (Section 3.2.3) and DPM clustering method (Section 3.2.4). In a, blue indicates areas of earliest activation while red indicates latest activation. White regions are not included in the active spike region. In b, the starting position of the wavefront is marked with a blue dot, while the ending position is marked with a red dot. The line indicates the path of the wavefront (the energy-weighted centroid). Subfigure c is the correlation map of all 741 spike segments. Similar column vectors indicate that the corresponding spike segments had a similar propagation pattern. For example, the similarity evident in column vectors 190 to 220 could be interpreted as forming a natural cluster.

3.5 Cluster result comparison with [76]. Subfigure a,b demonstrate two cases where several clusters in [76] should be merged into one. Subfigure c displays several unique clusters that [76] failed to discover. Note that the spike patterns in subfigure c are very rare in the datasets and tend to be missed with standard linear embedding and K-means clustering method.

3.6 Trajectory prediction with polynomial regression and mean-based regularization on coefficients of cluster.

3.7 Trajectories of spikes belonging to the same cluster. Each subfigure is a trajectory from one particular spike segment. Clustering is performed using Isomap and DPM. As can be seen the trajectories within this cluster has a highly consistent pattern.
3.8 Bar chart representing distribution of 20 identified clusters by applying DPM on Isomap projections. Using majority vote of spike label (seizure, none-seizure) of each cluster for seizure detection would render seizure detection sensitivity of: 0.5703, seizure detection specificity of: 0.9983 respectively.

3.9 Illustration of the HMM+SVM seizure prediction scheme. Input of the diagram is the raw µECoG data. Black bracket indicates the process of segmenting the raw data into spike segments. Green bracket generalizes the procedure of extracting features for each spike and classifying it into one of predetermined spike clusters using pre-trained DPM model. Red bracket generalizes the process of multiple HMM modeling for each overlapping sequence of 21 spike cluster labels. Three different colors in each box represent the likelihood of the sequence belonging to one out of three HMMs. Blue bracket represents SVM classification of the likelihood vector computed from a total of 21 overlapping sequences centered around the current spike. The output of the diagram is the classified seizure stage for each spike. Seizure onset is predicted when the current and previous 4 spikes are all classified as either pre-ictal or ictal spikes.
3.10 Histogram of seizure detection delay. Horizontal axis indicate the
detection delay. Negative delay means that the seizure is predicted
before the actual seizure onset. Vertical axis is the cumulative distribu-
tion of the delay. For datasets 2 and 3, all seizures were detected
with up to 2 seconds delay, and more than 70% of seizures were
predicted before the actual onsets. However, for dataset 1, 5 out of
7 seizures were predicted before the seizure onset, and all seizures
were detected within 10 sec. delay. .......................... 41

4.1 Sample clusters of neural activity patterns. This figure shows the
delay maps in several clusters identified using the method of [3]. The
delay map captures how the apparent wave in a neural recording
moves. Clusters 1-4 shown here correspond to upward, top-left to
bottom-right, counter-clockwise, and clockwise waves. ............ 50

4.2 Reconstruction and prediction results for three test sequences by
different methods. The top subfigure shows the original sequences.
Each remaining subfigure contains the reconstructed frames and
predicted frames for these three sequences by a particular model.
Model 8, 7 and 4 are 3 models out of 8 models with MCL training
that have lowest reconstruction error on these sequences respect-
ively. The comparison against single model with 1000 LSTM cells
and average prediction with 8 randomly initialized LSTM models are
shown below. The MCL training has led each model specializing at
one kind of sequences by having lower prediction error. The abso-
lute error plot against the ground truth demonstrates MCL training
have lower prediction error. ................................. 52
4.3 Peak signal noise ratio (PNSR) against prediction time with different methods of test set. .......................... 55
4.4 4 models prediction specialties in the ensemble trained with MCL. Each example sequence is shown with the vertical order of ground truth, reconstruction/prediction and absolute error. ............... 57
4.5 Video prediction framework. The generative model is built using convolutional LSTM [80]. The network flow is represented with solid arrow, whereas the losses for the generative model are represented with dashed arrows. ................................. 59
4.6 Understanding the benefit from adversarial training: The input to the discriminative model is either true history with true future or true history with predicted future. Third and fifth row of each example shows the activation of second to last layer output across all channel. The activation by true data is distributed almost evenly in both space and time domain to reconstruct the entire sequence. The activation by the sequence with predicted future however concentrates on spatial and temporal inconsistencies. For example, in the first sequence, the discriminative model finds the inconsistency in the last few frames. ................................. 61
4.7 Benchmark network, multi-resolution LSTM and LSTM with multi-resolution layers. The multi-resolution LSTM has two scales, and in each scale it has two layer structure. Only one layer is drawn per scale for simplicity. The dotted box represents the predicted frames and \( \hat{y} \) represents linear interpolated frames. ................................. 66
4.8 PSNR of predicted frames against prediction time. The benchmark model, benchmark model with adversarial, multi-resolution LSTM, LSTM with multi-resolution layer correspond to models 5,6,7,8 respectively in Tab. 4.2. LSTM with multi-resolution layer has a better long term prediction accuracy compared to other models. The PSNR is obtained by first computing MSE by averaging squared errors over all pixels over all frames and all sequences, and then converting the resulting MSE to PSNR.

4.9 Prediction result comparison between different methods: generative model, adversarial training, multi-resolution LSTM and LSTM with multi-resolution layers correspond to model 5,6,7,8 respectively in Tab. 4.2.

5.1 Pixel-wise object tracking framework. The network consists of two sub-modules: a local and a global model, working in a closed loop. At time step \( t \), a resized full resolution binary image \( \hat{Z}(t-1) \) is fed into the global model. In inference time, this binary image is the predicted segmentation map acquired from the local model at frame \( t-1 \). The global model then roughly predict where the object would appear in frame \( t \) based on past segmentation maps and generate a region of interests (ROI). The cropped image in the ROI at frame \( t \) is then fed into the local model for segmentation.

5.2 Local model for object segmentation in a ROI image. The M-CNN and F-CNN are feature normalization layers. \( h,c \) are hidden and memory states.
5.3 In each subfigure, each row in vertical order is: the segmentation result overlaid on top of the raw image, first layer convolutional LSTM cells, second layer convolutional LSTM cells. The displayed images are downsampled by 2. Row 1: Both true sequence and inserted frames comes from testing set. Row 2 and 3, we show the top 16 activations out of 256 cells. Note: (i) For ConvLSTM even with memory regression there is still a burn in time for memory to converge. (ii) Memory cells get far noisier in subfigure(a) compare to subfigure(b) after several step. (iii) There is memory cells co-adapt with noisy sequences, which act as action detection (encircled with red rectangle in figure).

5.4 Comparison between ConvLSTM and framewise segmentation. **Left:** ConvLSTM 1 and 2 represents training strategies with randomly replaced frame and without randomly replaced frame respectively. **Right:** IOU comparison per frame for convolutional LSTM training with randomly place frames vs segmentation.

5.5 Demonstration of observation difference between testing set observation and ground truth. Each row shows a sequence temporally downsampled by 4. From top to bottom: input to global in testing sequence, ground truth mask and predicted bounding box location.

5.6 Training framework for global attention model

5.7 Quality results for 8 videos. The prediction are overlaid on top of the original image. The results are based on 2-stage models. The failure case at sequence 2 and 8 are mostly due to a large camera motions.
List of Tables

2.1 Description of our datasets ........................................ 11

3.1 Trajectory Prediction Error (pixel) .......................... 34

3.2 Spike classification accuracy for dataset 2 using the single HMM classifier. The observation sequence length is fixed at 21 spike segments, spike segment to be predicted is either located in the middle of the sequence (delay approximately 1 sec) or in the end (no delay). ”Stationary initial” means that we used the stationary state distributions to set the initial state of each observation sequence. ”Recursive initial” means that we use the state assigned to the first spike segment by the classifier for the previous observation sequence. 40

3.3 Spike classification accuracy for dataset 2 using the HMM+SVM classifier. Spike segment to be predicted is located in the middle of the observation ........................................ 40
3.4 Seizure prediction accuracy for three datasets using the HMM+SVM approach. Seizure onset is predicted if 5 consecutive spikes are classified to either pre-seizure or seizure stage. The time between this prediction and the actual onset time is defined as the delay. Seizure prediction is considered correct if $-8 \text{ sec} \leq \text{delay} \leq 0 \text{ sec}$. The last column reports the average of the negative delays among all predictions (including those with positive delays).

4.1 PSNR over all predicted frames with different methods for test set. The MCL training and random initialization both have 8 models. Each ensemble models have 1000 nodes.

4.2 Comparison of the accuracy and number of parameters of all models. 64-64 represents the number of convolution LSTM cells in layer 1 and 2 are both 64. All the convolution LSTM cells uses $5 \times 5$ kernel. The multi-resolution LSTM structure has two scales, each scale has two layer convolution LSTM cells. 64-64, 64-64 means each scale uses a 64-64 two layer LSTM. Model 1 and 5 are trained with $l_2$ loss alone.
5.1 Number of filters for each modules in local segmentation network and global attention network. Notation $\times 2$ represents two identical layers that are connected. **Local model:** In M-CNN and F-CNN internal activation functions uses rectified linear unit (relu), where as the output activation function is $tanh$. The internal activation function in deconvolution is leaky-relu, the last activation function is sigmoid. **Global model:** every two convolution layer are followed by a pooling operation to reduce the spatial dimensionality. The input to fully connected layer is vectorized output of convolutional LSTM. For the second fully connected layer the input dimension is 1025, where we concatenate the feature from last layer with aspect ratio of the current video clip.

5.2 Average IOU at different frame length with template cropped close to ground truth location. For sequences shorter than preset length, we upsample the testing sequences to the fixed length.

5.3 Average IOU at different frame length.
Chapter 1

Introduction

Over 1/3 of the worlds 60 million people with epilepsy (3 million in the U.S.) have seizures that cannot be controlled by medication. First generation antiepileptic devices are promising, but efficacy is modest. Epilepsy surgery outcomes (e.g. 50-59% seizure freedom) have not improved in over 20 years. Currently many existing neurological data analyses rely on manual inspection. With new high-density electrode arrays that provide dramatically enhanced spatial resolution, the data volume is too large for manual review. Further, manual inspection can miss subtle features that automated machine learning techniques can detect. There is an urgent need for efficient and sensitive automated methods that can analyze the large volumes of data produced by next generation neurologic devices. In this thesis, I take a voyage of building efficient and automated tools for spike segmentation, seizure prediction and neural activity prediction by combining recent exciting developments in machine learning and with traditional signal processing methods.
1.1 Motivation

There are two major problems to solve for neural activity prediction. On the macro-scale, we focus on discovering the underlying structure in the repetitive spatial temporal patterns of individual spikes using manifold learning. We believe different wave patterns correspond to different neuron states. A thorough study of the wave patterns would give a better understanding towards the true nature of neuron activities. Most current seizure detection/prediction methods rely on clever designed features like spectral power, wavelet energy, spike rate and so on [10, 12, 14, 42, 48, 51]. Although they may have good performance in seizure prediction, these hand crafted features certainly would likely lose detailed information about the signal. One major issue with supervised learning using hand crafted features is that manual labeling of seizure datasets may not reveal the true nature of neuron activities. It is very challenging for human to give a meaningful label to intracranial EEG or ECoG data for a short time horizon (milliseconds). It is more feasible for a human expert to give a more general label, such as whether each long duration corresponds to an interictal stage or ictal stage. Given the fact that there is almost no way for human to label the data in precise time point or even a short time horizon, an unsupervised technique to analyze the data seems to be a better choice. In chapter 3, we present an unsupervised spike clustering framework to reveal activities within and between spike segments. In this work, a spike segment refers to a consecutive set of frames in which one or several adjacent channels have high amplitude.

On the micro-scale, being able to predict the neural signal in the near future from the current and previous observations has the potential to enable real-time responsive brain stimulation to suppress seizures. By electrically stimulating the
future location that a spike will propagate to, neural tissue refractory could be rendered, thus preventing the continued propagation of the spike. This would terminate an ongoing seizure and potentially prevent a new seizure from starting. Recognizing that there exist multiple activity pattern clusters, in the first half of chapter 4, we first explore an ensemble of LSTM models so that each model can specialize in modeling certain neural activities, without explicitly clustering the training data. We train the ensemble using an ensemble-awareness loss, which jointly solves the model assignment problem and the error minimization problem. However, predicting the raw video is very challenging because the algorithm must be able to predict the neural signals in a sufficiently long time horizon to allow enough time for medical intervention. In the latter part of chapter 4, a multi-resolution representation is proposed to accomplish long term prediction. The novel LSTM structure with multi-resolution layers could significantly outperform the single-resolution benchmark with similar number of parameters. To overcome the blurring effect associated with video prediction in the pixel domain using standard mean square error (MSE) loss, we use energy based adversarial training to improve the long-term prediction.

Other than application of medical imaging, I further propose a novel pixel-wise visual object tracking framework that can track any anonymous object in a noisy background. Provided with an object of interest at the first frame, visual object tracking is a problem of building a computational model that is able to predict the location of the object in consecutive frames. Tracking frameworks could be mainly categorized into detection based tracking and segmentation based tracking. Segmentation-based tracking algorithm have advantage over detection-based algorithm for handling a target undergoing substantial no-rigid motion. Yet many of
these segmentation based method have limited speed around 1 fps to achieve good prediction accuracy. Besides most of them fail to demonstrate their performance in challenging dataset like VOT [38]. We consider the pixel-wise object tracking as a time series prediction problem and propose a novel two-stage model handling micro-scale appearance change and macro-scale object motion separately. Our model performs pixel-wise object tracking at a reasonable accuracy in real time.

1.2 Contribution

The first part of this thesis focuses on macro-scale of neural signal and our main contributions include:

1. Build a graph-based signal processing framework for automatically locating bad channels for $\mu$ECoG datas either caused by manufacturing defects or loss of contact between channel and tissue. It also removes noisy spatial patterns and improves performance of spike segmentation. We segment spikes using 3D region growing to harvest clean spike.

2. Design a pairwise metric for spike segments to model the manifold spike segments living in. An Isomap [72] and Drichlet Process Mixture Model [8] based clustering scheme is proposed, which can automatically determine the number of clusters for spike patterns.

3. I further demonstrate the benefit and potential applications of spike clustering by performing wavefront prediction and seizure detection and prediction. The wavefront at each frame refers to an intermediate point inside spike region that has a strong signal and is moving in the same direction as the whole
region. A Tikhonov regularized wavefront prediction method is proposed for each spike, which restricts the trajectory pattern to be close to the mean trajectory of each spike pattern cluster. This method produced reasonably good performance on wavefront prediction.

4. A seizure prediction framework which uses a two stage classifier of hidden markov model (HMM) and support vector machine (SVM) is proposed. The framework has demonstrated the potential for seizure prediction by analyzing the temporal variation of the spike labels, although the prediction time is only within seconds of the actual seizure onset.

The second part of the thesis focuses on micro-scale of neural signal prediction and our main contributions include:

1. An auto-encoder model consisting of long short term memory (LSTM)[25] cells is proposed for neural signal prediction. Recognizing that there exist multiple activity pattern clusters, we build an ensemble of LSTM models so that each model can specialize in modeling certain neural activities, without explicitly clustering the training data. An ensemble-awareness loss is used to jointly solve the model assignment problem and the error minimization problem.

2. To enable prediction over long time horizon, two encoder-decoder-predictor structures using multi-resolution representation are presented. The novel LSTM structure with multi-resolution layers could significantly outperform the single-resolution benchmark with similar number of parameters.

3. To overcome the blurring effect associated with video prediction in the pixel
domain using standard mean square error (MSE) loss, an energy based adversarial training method is proposed to improve the long-term prediction. We propose a discriminative model with an encoder-decoder structure using 3D CNN model, which has been demonstrated to improve long term prediction accuracy.

The third part of the thesis focuses on object tracking in natural scene and the main contributions are as follows:

1. A novel object tracking framework consisting of two models is proposed. The global model learns the global motion pattern of the object and predicts the objects likely location in a new frame from its past locations. The global model employs a convLSTM structure to generate the latent feature characterizing the object motion, which is fed to a spatial transformer network to determine the location and size of the ROI in the new frame.

2. The local model performs object segmentation in a ROI identified by the global model, based primarily on the appearance features of the object in the new frame. The local model uses a convLSTM based structure whose memory state evolves to learn the essential appearance features of the object, enabling the segmentation of the object even under significant appearance shifts and occlusion.

1.3 Outline

The thesis is organized as follows:

In chapter 2, the device and data currently available from picrotoxin-induced
seizures are introduced. A primer on problem formulation and prior studies using machine learning on seizure detection and prediction are reviewed and discussed.

In chapter 3, the preprocessing algorithm for the μECoG data using graph filtering is first presented. Spike segmentation result using region growing techniques is then presented. A framework for spike pattern clustering is built on unraveling the manifold the spike segment lives on. To demonstrate the merits of clustering spike pattern for helping to solve clinical problems, results on wavefront prediction and seizure detection are shown.

In chapter 4, the micro-scale neural signal prediction problem is tackled. Based on the multi-cluster nature of aforementioned neural activities, a multiple choice learning framework is proposed. Unlike most ensemble models that enhance performance by averaging independently trained models with random initializations, an ensemble-awareness loss function is proposed to jointly solve the assignment problem and prediction problem. A parallel track is explored to solve the long term prediction problem. To enable realtime responsive brain stimulation to suppress or prevent seizures entirely at sufficiently long time horizon, we further propose two types of network structures, both employing a multi-resolution representation of the signal. To overcome the blurring effect associated with video prediction in the pixel domain using standard mean square error (MSE) loss, energy based adversarial training is also introduced to improve the long-term prediction.

In chapter 5, a novel object tracking framework consisting of two models are proposed. The global model learns the global motion pattern of the object and predicts the objects likely location in a new frame from its past locations. The local model performs object segmentation in a ROI identified by the global model, based primarily on the appearance features of the object in the new frame. The
local model uses a convLSTM based structure whose memory state evolves to learn the essential appearance features of the object, enabling the segmentation of the object even under significant appearance shifts and occlusion. The LSTM output further goes through a deconvolution layer to generate the segmentation map. The global model also employs a convLSTM structure to generate the latent feature characterizing the object motion, which is fed to a spatial transformer network to determine the location and size of the ROI in the new frame.

Finally, chapter 6 summarizes this thesis and presents future directions of research.
Chapter 2

Background on $\mu$ECoG data and seizure prediction

This chapter reviews the current devices and techniques for harvesting $\mu$ECoG data. This chapter also gives definition to the technical terms for the readers to understand the thesis work.

2.1 Multiplexed, Active Electrode Arrays

This section give a brief introduction of the device for recording $\mu$ECoG data. Over the past few year, Viventi, et al. [76] have adapted active, flexible electronics technology [35] into implantable neurological devices. These devices have advantages over current electrode technologies. (1) They are extremely thin, conform to the brains irregular geometry, yield higher fidelity signals, and appear to cause less tissue trauma. (2) They use active electronics, enabling in-situ amplification and multiplexing, and eliminate a 40 year constraint that implanted electrodes
be individually wired to remote electronics. And (3) they enable high resolution recording over large, clinically relevant areas (25,000 electrodes within the area of a typical 64-electrode grid). Fig. 2.1 shows a photograph of this flexible electronic neurological device.

\section{2.2 Dataset}

The analyzed micro-electrocorticographic ($\mu$ECoG) data used for this thesis comes from an acute in vivo feline model of epilepsy. Adult cats were anesthetized with a continuous infusion ($3 \sim 10 \text{ mg/kg/hr}$) of intravenous thiopental. Craniotomy and durotomy were performed to expose a $2 \times 3 \text{ cm}$ region of cortex. The high resolution electrode array was then placed on the surface of the brain over
Table 2.1: Description of our datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>sampling frequency (Hz)</th>
<th>recording length (min)</th>
<th>labeled seizures</th>
<th>detected spike segments</th>
<th>seizure spike segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1 (cat 1)</td>
<td>277.778</td>
<td>53.67</td>
<td>7</td>
<td>1685</td>
<td>254</td>
</tr>
<tr>
<td>Dataset 2 (cat 2)</td>
<td>925.925</td>
<td>32.33</td>
<td>27</td>
<td>3706</td>
<td>3191</td>
</tr>
<tr>
<td>Dataset 3 (cat 2)</td>
<td>925.925</td>
<td>26.16</td>
<td>27</td>
<td>2380</td>
<td>1930</td>
</tr>
</tbody>
</table>

Dataset 2 and Dataset 3 are from the same animal, with different implant position primary visual cortex, localized by electrophysiological recordings of visual evoked potentials. Picrotoxin, was topically applied adjacent to the anterior-medial corner of the electrode array in an amount sufficient to induce abnormal electrical spikes and seizures from the covered region[76].

The active electrode array placed on the cortex was used to record data from 360 independent channels arranged in 20 columns and 18 rows, spaced 500µm apart. Each electrode contact was composed of a 300µm × 300µm square of platinum. Two flexible silicon transistors for each electrode buffered and multiplexed the recorded signals [76]. The total array size was 10mm × 9mm. Three datasets were acquired from two different animals, as summarized in Table 2.1. In this work, a spike segment refers to a consecutive set of frames in which one or several adjacent channels have high amplitude.

Two datasets were acquired from the same animal, with the electrode located on different areas of the cortex. The patterns observed from the second placement were distinct from the patterns observed during the first placement, motivating splitting the dataset. For seizure prediction problem in chapter 3, the goal is to detect and predict seizure onset base on spike pattern variation. Testing set is formulated by using observation sequences extracted during one seizure period, pre-seizure period (8 sec) before this seizure, and the non-seizure period leading
up to this seizure. The rest data is considered as training set. The same approach
are iterated for each seizure.

For neural activity prediction in chapter 4, one seizure period and its leading
non-seizure period from dataset 1 (Tab. 2.1) is selected as testing dataset. The
rest of the raw video is considered as training set so as to get disjoint subset for
training and testing.

2.3 Preprocessing using graph filtering

All channel recordings are individually band-pass filtered between 1 and 50 Hz
with a 6th order butterworth filter in the forward and reverse direction. This is to
remove the high frequency noise in the temporal domain. In order to remove the
noise in the spatial domain and fill in missing channels caused by manufacturing
defects or loss of contact on human membrane, graph filtering based technique
is developed. Specifically, we represent a μECOG dataset as a graph, where each
vertex of the graph represents one signal channel, and the edge between two vertices
embeds the relational information between two channels. The weighted graph
$G = \{V, E, W\}$ consists of a finite set of vertices and a weighted adjacency matrix
$W$. We want the adjacency matrix $W$ not only reflects the spatial relationship
between two channels but also reflects the signal correlations. The adjacency
matrix is defined as:
Figure 2.2: Comparison of graph filtering and Gaussian smoothing. Subplots a,c are results using a 5 by 5 Gaussian smoothing kernel for spatial filtering, and subplots b,d are graph filtering results using 80 dominant graph coefficients. For Gaussian filtering bad channels are human labelled and interpolated with Gaussian distance kernel from good channels within a local region of 5 by 5. The results b,d preserved the original pattern as well as smoothed the signal.
\[ W(i, j) = D(i, j) * C(i, j) \]

\[
D(i, j) = \begin{cases} 
\exp\left(-\frac{\text{dist}(i, j)^2}{2\theta^2}\right) & \text{if } \text{dist}(i, j)^2 \leq 32 \\
0 & \text{otherwise} 
\end{cases} \tag{2.1}
\]

\[
C(i, j) = \begin{cases} 
\text{Corr}(i, j) & \text{if } \text{Corr}(i, j) \geq 0.8 \\
0 & \text{otherwise} 
\end{cases}
\]

In here, the distance \( \text{dist}(i, j) \) is the Euclidean distance in the \( x, y \) coordinates between channel \( i \) and \( j \). The hard thresholding of distance and correlation aims to generate a sparse adjacency matrix and potentially a disconnected graph. We have experimented with various threshold values, and through trial and error, we found that setting distance, correlation threshold and \( \theta \) at 32, 0.8 and 4, respectively, generates the disconnected graph quite well. The disconnected graph serves the purpose of filtering out the noisy channels caused by manufacturing defects or lose of contact and relieves the burden of human inspection. If all channels work properly, then the graph \( G = \{V, E, W\} \) would be a connected one. Otherwise, the graph would have a bunch of disconnected subgraphs with each subgraph consisting of the electrode channels with strong connections. The largest connected subgraph represents the good channels as we assume the majority of our electrode array are functioning properly and are correlated with each other. Then the set of bad channels are interpolated from good channels as \( f_{\text{bad}} = Mf_{\text{out}} \). Here \( f_{\text{bad}} \) at any frame is a vector consisting of the signals of bad channels, \( f_{\text{out}} \) is a vector consisting of the filtered signals of good channels, and \( M \) is a matrix with elements defined
in Eq (2), in which $i$ stands for one bad channel, $j$ stands for a good channel.

$$M(i,j) = \frac{\exp\left(-\left(\frac{\text{dist}(i,j)^2}{2\theta^2}\right)\right)}{\sum_j \exp\left(-\left(\frac{\text{dist}(i,j)^2}{2\theta^2}\right)\right)}$$

(2.2)

To filter the good channel signals, we form the non-normalized graph laplacian of the largest connected subgraph $G_{sub}$ as $L_{G_{sub}} = D_{G_{sub}} - W_{G_{sub}}$. $W_{G_{sub}}$ is the adjacency matrix for the subgraph, and the degree matrix $D_{G_{sub}}$ is a diagonal matrix whose diagonal element in each row is the sum of all the weights in the same row of matrix $W_{G_{sub}}$. Because graph Laplacian matrix $L_{G_{sub}}$ is a real symmetric matrix, it has a set of orthonormal eigenvectors represented by $U = \{u_n\}_{n=0,1,\cdots,N-1}$, where $N$ is the number of good channels. Since the graph is connected, the ordered eigenvalue associated with the eigenvectors has the following traits $0 = \lambda_0 \leq \lambda_1 \leq \lambda_2 \cdots \leq \lambda_{N-1}$. The spectrum of the graph laplacian is defined as $\sigma(L) \equiv \{\lambda_0, \lambda_1, \cdots, \lambda_{N-1}\}$. Once we have a graph spectral representation, in order to filter out high frequency component in the spatial domain, we can generalize filtering in signal processing to graph spectral filtering [62] as:

$$f_{out} = \hat{h}(L_{G_{sub}})f_{in}$$

$$\hat{h}(L) = U \begin{pmatrix} \hat{h}(\lambda_0) & 0 \\ \vdots & \ddots \\ 0 & \hat{h}(\lambda_{N-1}) \end{pmatrix} U^T$$

(2.3)

In our case, a simple thresholding of the graph spectrum could render good reconstruction result. This is done by setting $\hat{h}(\lambda_t) = \lambda_t$ for $t \leq T$ and 0 otherwise. It resembles low pass filtering in traditional signal processing. The threshold $T$
is selected to be the smallest number to preserve at least 90% of original energy. Figure 2.2 shows a comparison between graph filtering and gaussian spatial filtering. The blocking artifact caused by Gaussian filtering is removed by using graph filtering.
In this chapter, we present an unsupervised learning framework for spike analysis, which by itself reveals spike pattern. By applying advanced video processing techniques for separating a multichannel recording into individual spike segments, unfolding the spike segments manifold and identifying natural clusters for spike patterns, we are able to find the common spike motion patterns. And we further explored using these patterns for more interesting and practical problems as seizure prediction and spike wavefront prediction. These methods have been applied to in-vivo feline seizure recordings and yielded promising results. The overall spike pattern clustering involves three layers as shown in Fig. 3.1.
3.1 Spike Segmentation

In the acquired datasets, the multi-channel $\mu$ECoG signal has high spatial and temporal correlation, and therefore each spike could be treated as a spatial-temporally connected set of voxels that have high intensity values. A simple scheme to detect spike segment is by thresholding. However, this can separate a single spike into multiple disconnected segments, or generate isolated scattered points due to the sensor noise. In [76], a spike segment is detected when a single channel has a negative value exceeding a preset threshold (0.5 mV), and then all frames that are within a certain time window from this point both in the past (60 ms) and future (100 ms) are included in the spike. Because each spike lasts for different durations depending on its initial location and motion pattern, this simple scheme sometimes includes more than one spike or an incomplete spike in a detected segment, as shown in Fig. 3.3. This false segmentation of a spike makes the wavefront prediction within a spike almost impossible. Besides only spatially connected voxels that have high intensity signal should be included in one spike, whereas this prior method
Figure 3.2: Examples of spike segmentation. Subfigure a is for a spike with a spiral motion. Subfigure b is for a spike with a planar motion. Each subfigure shows the segmented volume in transparent blue overlaid with the $\mu$ECoG signals captured at different times, with vertical axis (time axis) corresponding to frame number, while the other two axis represent the spatial arrangement of $\mu$ECoG electrodes;
Figure 3.3: Comparison of fixed length segmentation [76] (subfigure a,c) against using region-growing based variable length segmentation (subfigure b,d). With fixed length segmentation, the entire 3D cube formed by extending from a selected center frame to both past and future frames by a fixed length. The high intensity voxels in all figures are indicated by transparent color. In subfigure a,c, all voxels include within each spike segment, whereas in subfigure b,d only the high intensity voxels are included within each spike segment. Subfigure a shows a falsely detected segment by the fixed length method, which includes a truncation on top and includes many extra voxels (mainly on the bottom) that have low intensities. The detected spike segment for the similar time duration by the proposed method is shown in subfigure b, which includes all high intensity voxels that are connected. Subfigure c shows a detected spike segment that includes three separate true spikes, which are correctly separated by the region-growing method show in subfigure d.
includes all pixels in the chosen duration in the detected spike. To overcome these limitations, a 3D region growing technique is applied to detect spike region.

The region growing algorithm first detects pixels with intensity greater than $P_t$. Each such pixel becomes the seed of a region. The regions are then repeatedly expanded by including their neighboring pixels that have intensity greater than $\mu - \alpha * \sigma$, with $\mu$ and $\sigma$ being the mean and standard deviation of all current pixels included in the regions. The algorithm converges until no more pixels can be included. We applied the region growing algorithm to the data (after graph filtering) captured from an acute in vivo feline model of seizures, using the parameters $P_t = 0.5mV$, $\alpha = 0.8$ and $T_t = 40ms$, with $T_t$ being the minimum span of each spike segment in time. The parameters are selected based on experiments to remove short spike segments and noise. Fig. 3.2 shows two detected spike segments from one dataset. Fig. 3.3 compares fixed length segmentation against region growing. The result of fixed length segmentation in Fig. 3.3(a) shows an incomplete spike segment, and Fig. 3.3(c) shows multiple true spikes included within one detected spike segment. These false segmentations would make spike movement prediction impossible. The new region growing method clearly overcomes these problems as shown in Fig. 3.3(b,d) respectively.

3.2 Spike Clustering Through Manifold Learning

For each of the detected spike segment, we want to characterize its spatial-temporal characteristics relying on pairwise metric that describes the relationship between spikes. Even with the most cleverly designed feature descriptor, spikes still
Figure 3.4: Examples of extracted features. Subfigures a and b are the delay and trajectory features of the 9 spikes closest to the centroids of the 9 dominant clusters identified by Isomap embedding (Section 3.2.3) and DPM clustering method (Section 3.2.4).

In a, blue indicates areas of earliest activation while red indicates latest activation. White regions are not included in the active spike region. In b, the starting position of the wavefront is marked with a blue dot, while the ending position is marked with a red dot. The line indicates the path of the wavefront (the energy-weighted centroid). Subfigure c is the correlation map of all 741 spike segments. Similar column vectors indicate that the corresponding spike segments had a similar propagation pattern. For example, the similarity evident in column vectors 190 to 220 could be interpreted as forming a natural cluster.
live in a high dimensional Euclidean space that has a non-linear lower dimensional structure. For any type of clustering algorithms to work properly, a clever way to unfold this manifold is necessary. In this section, we will first describe the pairwise metric to describe the manifold that spike segments live in. And then we present algorithms to unfold this structure and cluster patterns.

3.2.1 Delay Map and Wavefront Trajectory

Before we delve deep into pairwise metric, we quickly describe two simple features for visualizing spike patterns. Following [76], for each spike segment we computed a delay map, which has the same dimension as the 2D sensor array, and each element indicates the delay of the signal at that sensor with respect to a reference signal. For the channel that is not in the segmented region over the entire duration of the spike segment, a 360 ms delay is assigned, same as the longest duration of spike segment in the dataset. The reference channel is the channel that has the highest sum of energy during this particular spike. Fig. 3.4(a) shows sample delay maps for 9 representative spike segments. As can be seen the delay map is an effective and efficient way to characterize the spike motion.

We also determined the wavefront position at each instant in time by determining the energy-weighted centroid of the spike region. Denote the set of pixels that belong to the spike region at time $t$ by $S_n(t)$, and the signal of the channel $(x, y)$ at time $t$ by $p(x, y, t)$. The $x$-position of the wavefront at time $t$ was determined by

$$
\bar{x}_n(t) = \frac{\sum_{(x,y,t) \in S_n(t)} |p(x, y, t)|^2 x}{\sum_{(x,y,t) \in S_n(t)} |p(x, y, t)|^2}.
$$
The y-position was determined similarly. Fig. 3.4(b) shows the wavefront trajectories of the same set of 9 representative spike segments. The wavefront trajectory (a vector consists of the x and y positions from the beginning to the end frame of a spike) can also be used as the descriptor of a spike.

3.2.2 Raw Signal Correlation Map

Since the number of spikes we got from our dataset is large, instead of coming up with hand crafted features, a better way is to let the data speak for itself. As the spike segments lives in a high dimensional space, one way of finding a low dimensional manifold for this kind of data is through a pairwise relationship description. We used the correlation of spike segments as the pairwise relational metric. Assuming that the signals corresponding to two spike segments have duration \( n \) and \( m \) with \( n \geq m \). Denote their corresponding video signals by \( X \in P^{N_1,N_2n}, Y \in P^{N_1,N_2m} \), where the intensity values at voxels not belonging to the spike segment are set to zero. \( N_1,N_2 \) represent channel dimension of the 2D sensor array. To compute the correlation between \( X \) and \( Y \), we found a length-\( m \) segment in \( X \) that has the highest correlation coefficient with \( Y \), and used this maximum as the correlation between \( X \) and \( Y \). The raw signal correlation matrix \( W \) has a dimension of \( N \times N \), \( N \) is the total number of segments in the training set. The motivation for using the correlation as descriptor is that similar segments should have similar correlations with respect to other segments. Correlation may not depict the spatio-temporal characteristics of the spike segment directly, but it is crucial for unveiling the nonlinear subspace that similar spike patterns live on as we will mention next. An example raw correlation map is shown in Fig. 3.4(c).
3.2.3 Manifold Projection

Since the spike segments live on a high dimensional space, the goal of manifold learning is to construct a low dimensional embedding that maximally preserves the local structure after unfolding. Unlike its counterpart principle component analysis (PCA) [31], which tries to find a linear mapping, manifold embedding usually assumes nonlinear structure. In our case, the nonlinear manifold is constructed through the correlation matrix $W$ from the previous paragraph. In the following paragraphs, we briefly describe one method to unveil the manifold. This approach starts by building a graph adjacency matrix with each vertex of the graph represents a spike, each edge represents certain relationship between two spikes.

3.2.3.1 Isomap

Isomap [72] preserves the geodesic distance between each pair of points after unfolding the data. In our case, adjacency matrix $M$ of the dataset is constructed by using correlation matrix $W$ as:

$$M_{ij} = \begin{cases} 
(1 - W_{ij})/2 & \text{if i,j: k-nearest neighbors} \\
\infty & \text{otherwise} 
\end{cases} \quad (3.1)$$

The geodesic distance matrix $D$ is defined by the shortest path between each two vertices using the adjacency matrix $M$. The normalized geodesic distance
matrix $\tau(D)$ is defined as:

$$
\tau(D) = -\frac{HSH}{2} \\
S_{ij} = D_{ij}^2 \\
H_{ij} = \delta_{ij} - 1/N
$$

(3.2)

Finding the $d$ dimension embedding then can be resolved by finding the $d$ eigenvectors of matrix $\tau(D)$ with smallest eigenvalues. Given a spike segment, its manifold feature vector consists of the projections its geodesic distance vector onto these eigenvectors. The geodesic distance is calculated using Eq. 3.1. The detailed algorithm to solve Eq. 3.2 is described in [72].

The nearest neighborhood number $k$ in Eq. 3.1 is selected to be $\log(N)$ as suggested in [77], and the adjacency graph does not have to be fully connected. In fact by setting $k$ around $\log(N)$, the adjacency graph we built has a few subgraphs which only has one single vertex. We have observed that constructing the adjacency matrix this way also serves the purpose of removing the impact of some irregular/uncommon segments that is caused by noise or false segmentation. The larger subgraphs have interesting and clean spike patterns and some of which highly correlate with seizure onset. Each large subgraph may consist of more than one spike pattern, therefore further clustering is necessary in order to separate the spike patterns within each subgraph. For our dataset, we have found that besides those subgraphs with with one node, there are a few small subgraphs (each with less than 20 nodes) plus one large subgraph. We ignore the single node subgraph, and treat each of the remaining small subgraphs as one cluster, and further separate the large subgraph into multiple clusters using manifold-based clustering. That is, we determine the eigenvectors for the large subgraph and project the correlation
vectors of the spike signals in this subgraph into the eigenvectors to form their embedded features. We then apply an unsupervised clustering method on these embedded features as described below. The projection dimension $d$ in Isomap is selected by finding the largest gap between eigenvalues, and is set to 20 in our experiment.

### 3.2.4 Clustering Using the Dirichlet Process Mixture Model

To find spike patterns from low dimensional embeddings, we used Dirichlet Process Mixture (DPM) model for clustering. Unlike k-means and the Gaussian Mixture Model, the DPM model does not require that the number of clusters in the dataset is known a priori. This nonparametric Bayesian model learn the number of clusters from the dataset. DPM is a special mixture model, where the mixing coefficient is a random variable that follows a certain distribution. A special case of the DPM has the Gaussian distribution as its base distribution and uses the Beta distribution with a concentration parameter to model the mixing coefficient. We used Mean-Field Variational Inference\cite{8} to learn the DPM parameters for a given set of samples. Once the parameters were determined, a given sample was assigned to the mixture with which it has the highest likelihood.

### 3.2.5 Clustering Results Comparison

We applied our manifold clustering techniques onto 741 spike segments obtained from in-vivo recordings of a feline model of epilepsy. To compare with prior work \cite{75, 76}, we used the same portion of dataset 1 (Table 2.1). A detailed description of the dataset is given in Section V.

We have found that DPM on isomap projections have derived cleaner spike
patterns within each cluster compared with the clusters derived in [76]. In this prior work, we first use fixed length segmentation to find spikes and then used delay and energy as our feature representation. PCA is applied on concatenated feature maps to find a linear low dimensional embedding and then adopt K-means algorithm for spike clustering. The region based spike segmentation in this work could adaptively find neural activity restricted to a local region and allow variable length of spike segments. This ensures cleaner spike segmentation. Furthermore, the number of raw pixels belonging to each spike segment generally exceed tens of thousands, and it is highly likely that these spike segments lie on a non-linear low dimensional manifold than a linear one. PCA as a linear dimension-reduction method used in [76] fails to capture the non-linearity in the dataset. One limitation of K-means is that it requires the knowledge of the number of clusters, which is hard to determine in the case of spike clustering, even with human inspection. One observation we have is that some types of spikes are much more frequently observed than others. K-means clustering usually works well only when the clusters have comparable sizes. On the other hand, DPM, as a special mixture model, does not require the knowledge of the number of clusters a priori and can generate unevenly distributed clusters with the proper selection of the concentration parameters. We present visualization results of spike pattern clusters in Fig. 3.5. Fig. 3.5(a) and (b) demonstrate two cases where clusters derived in [76] should be merged into one. Fig. 3.5(c) shows new clusters that [76] failed to discover.
Figure 3.5: Cluster result comparison with [76]. Subfigure a,b demonstrate two cases where several clusters in [76] should be merged into one. Subfigure c displays several unique clusters that [76] failed to discover. Note that the spike patterns in subfigure c are very rare in the datasets and tend to be missed with standard linear embedding and K-means clustering method.

3.3 Wavefront Trajectory Prediction

As described earlier, the wavefront at any particular frame time is obtained by averaging the locations of the channels weighted by the channel signal magnitude. Because the signal of the leading edge of the spike usually has much higher intensity compared to the rest of the spike, this serves as an approximation of the wavefront. Let \((x_t, y_t)\) denotes the wavefront location at frame \(t\). The problem of wavefront prediction is to predict future locations of the wavefront at frames \(t > k\), given their locations in \(k\) beginning frames \(t = 1, 2, \ldots, k\). One way to predict this time series is by polynomial regression, where we model the trajectory as a degree-\(n\) polynomial function of \(t\):

\[
x_t = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_0
\]

\[
y_t = b_n t^n + b_{n-1} t^{n-1} + \cdots + b_0
\]

(3.3)

Because \(x_t\) and \(y_t\) are individually parameterized by polynomial function, without loss of generality, we give the mathematic formulation of solving \(a_n, \ldots, a_0\) as
the polynomial regression problem in Eq. 3.4. The parameters of this polynomial function are regressed from the first \( k, k \leq n \) observations of the trajectory \( x_t, t = 1, \ldots, k \) and then we use the coefficients found to generate wavefront positions in the future. Specifically, we determine the coefficients by solving the following problem:

\[
\begin{align*}
\minimize_a ||x - Ta||^2 \\
where \ x = [x_1, x_2, \ldots, x_k]^T \\
a = [a_0, a_1, \ldots, a_n]^T \\
T = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & 2 & \cdots & 2^n \\
1 & 3 & \cdots & 3^n \\
\vdots & \vdots & \ddots & \vdots \\
1 & k & \cdots & k^n
\end{pmatrix}
\end{align*}
\]

(3.4)

The least square solution is \( a = (T^T T)^{-1} T x \). As we can only use a small initial portion of the trajectory to estimate the future trajectory, it would yield unreliable result as shown in Fig. 3.6(a).

### 3.3.1 Trajectory Prediction with Mean-based Regularization

As shown before, predicting the wavefront trajectory of a spike from a few samples at the beginning of the trajectory is hard, as the wavefront trajectory varies from one spike to another in a highly non-linear way. This is an extremely challenging task, without any prior knowledge of the spike pattern. By recognizing
Figure 3.6: Trajectory prediction with polynomial regression and mean-based regularization on coefficients. The spike trajectories are predicted given five observations by fitting a degree-4 polynomial. Red curve is the ground truth, blue curve is the prediction results. Subfigure (a) shows the results using polynomial regression. The overlapped 5 points of prediction and ground truth is the first five observation of that particular trajectory. Subfigure (b) demonstrates the predictions by further adding mean-based regularization.
Figure 3.7: Trajectories of spikes belonging to the same cluster. Each subfigure is a trajectory from one particular spike segment. Clustering is performed using Isomap and DPM. As can be seen the trajectories within this cluster has a highly consistent pattern.

As shown in Fig. 3.7, the spikes in the same cluster have consistent trajectory patterns. If we know the cluster that a spike belongs to, and we also know the mean of the polynomial coefficients of the spike trajectories in this cluster based on training data, we can modify the previous least squares fitting problem with an additional constraint. Given the cluster index $i$ of a spike, we determine the polynomial coefficients using the beginning of the trajectory, with additional constraint that the coefficients should be close to the coefficient mean of that particular cluster. Let $\bar{a}$ denotes the mean of the coefficients and $Q$ is the inverse covariance matrix of $a \in$ cluster $i$, $\|a\|_Q^2$ stands for the weighted norm $a^T Q a$. This formula-
tion is equivalent to find the maximum a posterior estimation of $a$ by assuming a
gaussian prior distribution.

$$\min_{a} \|x - Ta\|^2 + \|a - \bar{a}\|^2_Q$$ (3.5)

The optimization problem has a closed form solution.

$$a^* = \bar{a} + (T^T T + Q)^{-1} (T^T (x - T \bar{a}))$$

Fig. 3.6(b) shows two of the prediction results using mean-based regularization. In addition to using mean-based regularization, we also compared merely using mean of the polynomial coefficients of the cluster as a strong benchmark. The root mean square error between the predicted wavefront locations and the actual locations using regularization, mean and simple least squares fitting are summarized in Table 3.1. To make the benchmark method of only using polynomial fitting stronger, we further restrict all predicted positions of x,y coordinates using different methods to be within the grid size. As shown in Table 3.1 prediction of mean-based regularization is within 4 pixels of the true location, on average, which is much better than prediction using polynomial fitting just based on the first 5 observations.

### 3.4 Seizure Detection and Prediction

In this section, we investigate how to detect and predict seizure onset based on the spike pattern label variation. Note that seizure prediction relying on one spike segment is unreliable. Figure 3.8 displays the distribution of nonseizure vs.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cluster-based polynomial fitting with mean-based regularization</th>
<th>Prediction using cluster mean fitting</th>
<th>Polynomial fitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>7.62</td>
<td>8.10</td>
<td>20.21</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>8.31</td>
<td>9.07</td>
<td>23.52</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>8.13</td>
<td>8.49</td>
<td>21.30</td>
</tr>
</tbody>
</table>

Root mean square error of trajectory prediction for all detected spikes in different datasets. Wavefront locations in all remaining frames of a spike segment are predicted from the polynomial coefficients derived from the wavefront locations in the first five frames.

Seizure spikes over the clusters found by the DPM clustering method on isomap projections. Each bar represents one of the 20 different spike patterns identified. It can be seen that ictal (during a seizure) and inter-ical (between seizure) spikes are well separated. In particular, cluster 7, 11, 12, 13, 18 and 19 mainly contain spikes that occur during seizures whereas cluster 1, 2, 3, 4, 5, 6, 14, 15, 16 are mostly spikes that occur between seizures. This result reveals a cleaner separation between seizure cluster and non-seizure clusters against [75]. But there are certain clusters like cluster 9 that mixes seizure patterns and non-seizure patterns. Therefore for a more robust seizure detection and prediction one has to take into account of spike pattern variation in a longer time horizon.

### 3.4.1 Using a Single Hidden Markov Models

We have explored two different ways of using HMM for seizure stage classification. In this single HMM approach, hidden states in the HMM correspond to all possible seizure stages (i.e., interictal, pre-ictal, and ictal). Given a sequence of $T$ observations (observation at time $t$ is the cluster label of the spike at time $t$), the
Figure 3.8: Bar chart representing distribution of 20 identified clusters by applying DPM on Isomap projections. Using majority vote of spike label (seizure, none-seizure) of each cluster for seizure detection would render seizure detection sensitivity of: 0.5703, seizure detection specificity of: 0.9983 respectively.
most likely state transition path then tells the classification results for all $T$ spikes. During the training stage, we collect observations and their corresponding state labels to derive the transition and emission probabilities for HMM. Inferring the hidden state label in this case corresponds to finding the Viterbi path. We found that prediction accuracy begins to decrease as the observation sequence length reached beyond a certain point. Therefore instead of using all past observations up to time $t$, we use only a fixed number of past observations. Through trial and error, we found that using 20 observations yielded the best result. We have experimented with two ways for assigning the initial state of HMM, 1) using the stationary probability, 2) using the state assigned to the first spike segment when applying HMM classifier to the previous overlapping observation sequence.

### 3.4.2 Using Multiple Hidden Markov Models plus a second stage classifier (HMM+SVM)

In this approach, we built a separate HMM for each individual seizure stage. $\lambda_i = (A_i, B_i, \pi_i)$ denotes the parameters for the $i$-th stage. During the training stage, the goal is to adjust model parameters $\lambda_i = (A, B, \pi)$ to maximize $P(O|\lambda_i)$ using observation sequences in the training data for stage $i$. Model parameters are updated iteratively through Baum-Welch algorithm. Since we have no prior knowledge of how many states to choose, the number of hidden states $n$ is chosen to maximize $P(O|\lambda_i)$. We found that four hidden states work well for all three different HMMs.

We have found that using the HMM trained separately for each seizure stage cannot accurately classify a spike during the transition period from non-seizure to seizure (or vice versa). We suspect that this is because, during the transition
Figure 3.9: Illustration of the HMM+SVM seizure prediction scheme. Input of the diagram is the raw µECoG data. Black bracket indicates the process of segmenting the raw data into spike segments. Green bracket generalizes the procedure of extracting features for each spike and classifying it into one of predetermined spike clusters using pre-trained DPM model. Red bracket generalizes the process of multiple HMM modeling for each overlapping sequence of 21 spike cluster labels. Three different colors in each box represent the likelihood of the sequence belonging to one out of three HMMs. Blue bracket represents SVM classification of the likelihood vector computed from a total of 21 overlapping sequences centered around the current spike. The output of the diagram is the classified seizure stage for each spike. Seizure onset is predicted when the current and previous 4 spikes are all classified as either pre-ictal or ictal spikes.
period, the observation sequence contains spikes from both the non-seizure period and the seizure period. If we record the actual likelihood values for all three seizure stages in time, it is likely that, during the transition period, the likelihood for the non-seizure stage decreases, and the likelihood for the seizure stage increases. To exploit the likelihood variation pattern over time, we adopted a sliding window approach and find the likelihood vector (consisting of the likelihood values of the three seizure stages) over each sliding window. As shown in Fig 3.9, to classify the observation $O_t$ as indicated by the red double dashed vertical lines on the top, we collect a total of $2n + 1$ observations. For each sliding window containing $n$ observations, we derive the log likelihood that it belongs to each of the three seizure stages using the 3 trained HMM models. Because there are a total of $n + 1$ sliding windows over $2n + 1$ observations, we form a new $3n + 3$ feature vector $G_t$ that contains the log likelihood values for the three seizure stages of each over the $n + 1$ sliding windows. We apply a trained classifier to this feature vector to determine the seizure stage of the spike $O_t$. For this second stage classifier, we use the multi-class support vector machine with linear kernel and radial basis function (RBF) kernel.

### 3.4.3 Detection and Prediction Result

We analyzed micro-electrocorticographic ($\mu$ECoG) data from an acute in vivo feline model of epilepsy. Adult cats were anesthetized with a continuous infusion (3 ~ 10 mg/kg/hr) of intravenous thiopental. A craniotomy and durotomy were performed to expose a 2 x 3 cm region of cortex. The high resolution electrode array was then placed on the surface of the brain over primary visual cortex, localized by electrophysiological recordings of visual evoked potentials. Picrotoxin, a GABA-A
receptor antagonist that blocks inhibition, was topically applied adjacent to the anterior-medial corner of the electrode array in an amount sufficient to induce abnormal electrical spikes and seizures from the covered region[76].

The active electrode array placed on the cortex was used to record data from 360 independent channels arranged in 20 columns and 18 rows, spaced $500\mu m$ apart. Each electrode contact was composed of a $300\mu m \times 300\mu m$ square of platinum. Two high-performance, flexible silicon transistors for each electrode buffered and multiplexed the recorded signals [76]. The total array size was $10\text{mm} \times 9\text{mm}$.

Because of the variation of implant position and time effect, the predictor we built is dataset specific. But to avoid overfitting, we use leave one out cross validation. Namely we use observation sequences extracted during one seizure period, pre-seizure period (8 sec) before this seizure, and the non-seizure period leading up to this seizure as our testing data, use the rest of the data for training. We repeat this approach for each seizure.

We compared the performances of both the single HMM classifier and the HMM+SVM classifier (Fig. 3.9) for determining the seizure stage of a spike. The spike classification performance using the single HMM classifier is given in Table 3.2, results obtained using the HMM+SVM approach are summarized in Table 3.3. Here the accuracy is defined as the percentage of spikes that are correctly identified into the three seizure stages as compared to the manually labeled ground truth. We can see that the HMM+SVM approach achieves significantly higher accuracy than the single HMM approach.

Tables 3.2 and 3.3 considered the spike classification accuracy. Even though the HMM+SVM classifier was able to achieve very high spike classification on average, the classification accuracy for the pre-ictal spike is much lower, because pre-ictal
Table 3.2: Spike classification accuracy for dataset 2 using the single HMM classifier. The observation sequence length is fixed at 21 spike segments, spike segment to be predicted is either located in the middle of the sequence (delay approximately 1 sec) or in the end (no delay). "Stationary initial" means that we used the stationary state distributions to set the initial state of each observation sequence. "Recursive initial" means that we use the state assigned to the first spike segment by the classifier for the previous observation sequence.

<table>
<thead>
<tr>
<th>Delay</th>
<th>Stationary initial</th>
<th>Recursive initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>no delay</td>
<td>0.765</td>
<td>0.770</td>
</tr>
<tr>
<td>10 segments (≈ 1 sec delay)</td>
<td>0.779</td>
<td>0.784</td>
</tr>
</tbody>
</table>

Table 3.3: Spike classification accuracy for dataset 2 using the HMM+SVM classifier. Spike segment to be predicted is located in the middle of the observation

<table>
<thead>
<tr>
<th>Observation length</th>
<th>svm-rbf</th>
<th>svm-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>21 segments (≈ 1 sec delay)</td>
<td>0.920</td>
<td>0.846</td>
</tr>
<tr>
<td>41 segments (≈ 2 sec delay)</td>
<td>0.940</td>
<td>0.852</td>
</tr>
</tbody>
</table>

spikes occur much less frequently than inter-ictal and ictal spikes. Furthermore, we cannot simply consider any spike classified as pre-ictal as the onset of a new seizure, as that will yield very noisy and inconsistent prediction. Instead, we predict the onset of a new seizure if the current spike and 4 previous spikes are classified to either pre-seizure or seizure spike. If the seizure actually happened after the predicted onset within 8 sec, we consider the prediction accurate. Table 3.4 summarizes the seizure prediction accuracy using the HMM+SVM approach. Figure 3.10 shows the distribution of the prediction delay.
Table 3.4: Seizure prediction accuracy for three datasets using the HMM+SVM approach. Seizure onset is predicted if 5 consecutive spikes are classified to either pre-seizure or seizure stage. The time between this prediction and the actual onset time is defined as the delay. Seizure prediction is considered correct if \(-8 \text{ sec} \leq \text{delay} \leq 0 \text{ sec}\). The last column reports the average of the negative delays among all predictions (including those with positive delays).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>prediction accuracy</th>
<th>number of false positive</th>
<th>average time before seizure</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat 1</td>
<td>5/7: 0.7143</td>
<td>0</td>
<td>1.86 sec</td>
</tr>
<tr>
<td>cat 2</td>
<td>20/27: 0.7407</td>
<td>1</td>
<td>2.22 sec</td>
</tr>
<tr>
<td>cat 2</td>
<td>24/27: 0.8889</td>
<td>0</td>
<td>2.33 sec</td>
</tr>
</tbody>
</table>

Figure 3.10: Histogram of seizure detection delay. Horizontal axis indicate the detection delay. Negative delay means that the seizure is predicted before the actual seizure onset. Vertical axis is the cumulative distribution of the delay. For datasets 2 and 3, all seizures were detected with up to 2 seconds delay, and more than 70% of seizures were predicted before the actual onsets. However, for dataset 1, 5 out of 7 seizures were predicted before the seizure onset, and all seizures were detected within 10 sec. delay.
3.5 Conclusion

In this chapter, we have combined advanced video analysis and machine learning algorithms to analyze μECoG datasets in novel ways. We have developed efficient methods for identifying and localizing the spatial and temporal extent of inter-ictal and ictal spikes through region growing. For those identified spikes we designed a pairwise metric that characterize the similarity between the movement patterns of different spikes. Using this similarity metric, we discovered the lower-dimension manifold features using the Isomap and furthermore discovered different spike patterns using the DPM clustering method on the manifold features. The clustering result reveals cleaner patterns that have not been seen before.

We further demonstrate the benefits of using the clustering pattern to solve wavefront prediction and seizure prediction problems. We casted the wavefront prediction into a polynomial regression problem by using wavefront coordinates of the first few frame to estimate the parameters. We then used the polynomial model to predict future wavefront trajectories. We found that by identifying the spike cluster that a spike belongs to and requiring the polynomial coefficients to be close to the mean coefficients of the trajectories in that cluster, we are able to achieve more accurate trajectory prediction. The mean error of predicted wavefront locations is only around 4 channels away from the true locations. The error is caused mainly by two reasons. One is the error in spike segmentation using 3D region growing, which would consider merging / splitting spikes as the same one as long as these two or more spikes were spatially connected at one time. This issue makes trajectory of a spike less meaningful when there are multiple spikes that intersecting each other. The second is that polynomial fitting is one simple way of describing wave pattern, more advanced techniques like recurrent neural network
or its variants could be a better model. Another interesting issue would be how to predict spike pattern of a new spike based on its first few frames as well as the past temporal variation of spike patterns. These problems remain to be explored in the next chapter.

We further investigated seizure detection problem using each spike segment’s cluster label. The proposed two stage HMM+SVM classifier yielded accurate seizure detection and possibly prediction, by detecting 49 out of 61 seizures before the seizure onset. Our framework has demonstrated the potential for seizure prediction by analyzing the temporal variation of the spike labels, although the prediction time is only within seconds of the actual seizure onset. This is due to the fact that our datasets consisted of induced seizures that occur very frequently. As such, there does not appear to be a pre-seizure stage that occurs far ahead of the seizure onset.
Chapter 4

Neural Activity Prediction

This chapter describes two different approaches for predicting the raw neural signal. Being able to predict the neural signal in the near future from the current and previous observations has the potential to enable real-time responsive brain stimulation to suppress seizures. However, this problem is very challenging because the algorithm must be able to predict the neural signals in a sufficiently long time horizon to allow enough time for medical intervention. We consider how to accomplish long term prediction using a LSTM based encoder-decoder-predictor structure.

4.1 Review on Video Prediction

Studies have focused on seizure prediction for decades, but reliable prediction of seizure activity many minutes before a seizure has been elusive. Constructed features like wavelet, energy of spike and spectral power [1, 5, 10, 12, 14, 42, 51, 66, 71] are applied on electroencephalogram (EEG) or electrocorticographic (ECoG) data with coarse resolution for most of current neurological analysis works.
Prior work has focused on predicting seizures minutes or hours in advance of the seizure, using supervised datasets with labeled examples of seizures. However, with the rich spatial and temporal patterns unveiled by high resolution micro-electrocorticographic (µECoG) [76] which is very similar to a high-frame rate video signal, accurate prediction of neural activities at the sub-second level could become a very interesting and tractable problem. Neural signal prediction on this time frame would allow responsive stimulation to suppress seizures. This kind of neural signal prediction could also find a compact representation for neural activity which could lead to understanding non-pathologic neural activity. To capture the highly non-linear dynamics in neural activities, deep learning neural networks appear to be a promising solution. But, learning a compact representation for neural video prediction gets more challenging when trying to predict in a longer future and the deep neural network model develops a more severe vanishing gradient problem.

To model long term dependencies, Long Short Term Memory (LSTM) units [25] were proposed as an improvement over vanilla RNN to solve vanishing gradients problem by introducing gate functions. Gated Recurrent Unit (GRU) [9] as simplified version of LSTM units has achieved better performance in a number of applications [4, 34, 73]. Even though LSTM and GRU tries to solve vanishing gradient problem by preserve long term dependency in their cells, modeling long term dependencies is still difficult. For neural language translation, instead of decoding a sequence from a compact feature learnt through an encoding network, [4, 58] use word-by-word attention mechanism which allow direct connection between premise and hypothesis sentences. By using such direct connections, it alleviates the vanishing gradient problem for long sentence translation. Another different approach is to use memory augmented neural networks as Neural Turing Machines [19]. By
using external memory to store information, the explicit storage of hidden states creates a shortcut through time. [20, 61] both achieve good performance by using external memory network.

To help the reader to gain a better understanding of this thesis, in the following subsections, a review of time series modeling using LSTM is given.

4.1.1 LSTM

For general purpose of sequence modeling, LSTM as a special RNN structure has proven to be capable of modeling long range dependency in many applications [18, 68, 81]. The crucial feature of LSTM compared to the classical RNN is the memory cell noted by $c_t$, which serves as a conveyor belt connecting time series and acts as an accumulator. The input gate $i_t$ controls the extent that current input $x_t$ and past hidden states $h_{t-1}$ have on affecting the current cell state. Simultaneously a sigmoid layer called forget gate $f_t$ decides what information is going to be thrown away or dampened from the current cell state. Finally the output of the LSTM is a filtered version of the current cell state controlled by the output gate $o_t$ and pushed through a tanh function so that the output has values between -1 and 1.

The basic LSTM cell structures are summarized as follows, where $\circ$ denotes the Hadamard product:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (4.1)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$
4.1.2 Convolutional LSTM

The major drawback of LSTM in handling spatiotemporal data is its usage of full connections in input-to-state and state-to-state transitions in which no spatial information is encoded. To overcome this problem, [80] proposes to replace fully connected matrix of $W_{x*}$ and $W_{h*}$ with a convolutional operator shown in Eq. 4.2.

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf} * X_t + W_{fi} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \text{tanh}(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \quad (4.2)$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o)$$

$$H_t = o_t \circ \text{tanh}(C_t)$$

In equation 4.2, the hadamard product $\circ$ between $W_{c*}$ and $C$ are crucial for learning long term dependencies. It restricts cross-channel information exchange and overcomes vanishing gradient problem. Replacing hadamard product with convolution would not achieve similar performance for time sequence model.

4.1.3 LSTM and ConvLSTM for Video Prediction

LSTM based recurrent neural network has been widely applied in the field of neural machine translation[4, 11, 68], video analysis[30, 81, 84, 86], etc. In these tasks formulated as a supervised learning problem, the goals are to match a set of observation sequences to the correct target sequences or labels. However in many applications correspondences between videos or detailed labels are not available, exploring the spatial-temporal structure of the raw video sequences would be more appealing.
For μECoG prediction, we used [67] as the baseline model. The baseline model has an encoder, a decoder and a predictor. The encoder learns a compact representation for a certain number of observed frames, and the decoder reconstructs these observed frames from the encoded feature. The predictor then predicts future frames of the given sequence based on the encoder feature. The entire system can be learnt in an end-to-end manner based on the training sequences. To enhance the performance, instead of using one layer of LSTM/convLSTM for each submodule (encoder/decoder/predictor), multiple LSTMs/convLSTMs are stacked to form more complex structures by adding nonlinearity.

### 4.2 Diversity Encouraged Learning of Ensemble Models

In this section, we describe the first approach for predicting neural activity signal in μECoG video. The μECoG dataset have been observed to form multiple clusters, each with a distinct neural activity pattern, as shown in Fig. 4.1. One way to exploit this multi-cluster nature of the μECoG videos is by fitting a model for each cluster of sequences. This approach would require one to segment a long video into short sequences and furthermore, classify each sequence to one of predefined clusters. Such an approach is highly limited by the sequence segmentation and clustering. Besides, this pipelined framework is against the common approach of deep learning where one usually trains in an end-to-end manner. Another alternative is by adding more LSTM cells. As the number of parameters grows in $O(n^2)$, $n$ being the number of LSTM cells in each layer, adding more LSTM cells is not efficient to fully exploit the clustered nature of the underlying signals. In
the following subsection we propose a new approach to solve the assignment and optimization problem in an end-to-end manner.

4.2.1 Multiple choice learning (MCL)

Typically, ensemble models are trained independently under different random initializations and prediction results are averaged during test time [24, 41]. These models are commonly viewed as experts or specialists in the literature, although they are rarely trained to encourage diversity and specialization. In [24], the authors distilled information from ensembles of convolutional neural network (CNN) by pre-clustering data. They pre-cluster the images in the dataset based on image categories, and each CNN specialist is only trained on a subset/cluster of images. During inference time, the prediction of label is by having a generalist model first determines the potential subcategory that input image might belong to and letting the ensemble models trained with this sub-category further determine the label of the image. Although this approach is sound for image classification where one typically has many labelled images, such an approach leveraging label-based pre-clustering is not feasible when facing unlabelled video dataset. Instead, we adopt the framework of Multiple Choice Learning (MCL) [21, 41], where the assignment of a training sample to each model is jointly solved with finding the optimal parameters for all models.

In the video prediction setup, we have a set of models $\theta_1, \theta_2, \ldots, \theta_M$ such that $\theta_m : x_{(1:t-n)} \rightarrow \hat{x}_{(1:t)}$. $x_{(1:t-n)}$ are input frames in time 1 to $t-n$. $\hat{x}_{(1:t-n)}$ are the reconstructed frame from the input and $\hat{x}_{(t-n+1:t)}$ are the predicted frames in time $t - n + 1$ to $t$. The loss for a sequence $x_{(1:t)}$ is defined in Eq. 4.3, where $l(\theta(x_{(1:t-n)}), x_{(1:t)})$ is the mean square error loss between $\hat{x}_{(1:t)}$ and $x_{(1:t)}$. The goal
Figure 4.1: Sample clusters of neural activity patterns. This figure shows the delay maps in several clusters identified using the method of [3]. The delay map captures how the apparent wave in a neural recording moves. Clusters 1-4 shown here correspond to upward, top-left to bottom-right, counter-clockwise, and clockwise waves.
of our MCL setup is to find the assignment variable \( p_{im} \) and parameters for \( \theta_m \) by solving the optimization problem defined in Eq. 4.3.

\[
\min_{\theta_{(1:M)}, p_{im}} \sum_{i=1}^{I} \sum_{m=1}^{M} p_{im} l(x_i, \theta_m(x_i))
\]

s.t. \( \sum_{m=1}^{M} p_{im} = 1, p_{im} \in \{0, 1\} \) \( \theta_m(x) : x_{1:t-n} \rightarrow \hat{x}_{1:t} \)
\[
l(x, \theta_m(x)) : ||x_{(1:t)} - \theta_m(x)||^2
\]

(4.3)

Note that in the training stage, at each iteration, we know the reconstruction and prediction accuracy of each current ensemble model on one instance \( x_i \). Therefore, we can assign a training instance \( x_i \) to the model that has the minimal reconstruction and prediction error. The optimization problem in Eq. 4.3 could be solved with a coordinate descent algorithm [41] with stochastic gradient descent (SGD) shown below. The solution alternates between finding the assignment and optimizing the corresponding model’s parameter.

4.2.2 Training the LSTM ensemble using MCL

We train a LSTM ensemble with 8 models. For parameter initialization, we first try random initialization for all 8 models. But we find once the gradient descent is made for the first mini-batch, one model is much better updated comparing to the rest and this model would have the lowest error for the majority of the remaining mini-batches. This causes only one model gets updated during training most of time. To overcome such problem, we randomly divide the training set into 8 non-overlapping subsets. Initialize one model with one of subset. We train
Figure 4.2: Reconstruction and prediction results for three test sequences by different methods. The top subfigure shows the original sequences. Each remaining subfigure contains the reconstructed frames and predicted frames for these three sequences by a particular model. Model 8, 7 and 4 are 3 models out of 8 models with MCL training that have lowest reconstruction error on these sequences respectively. The comparison against single model with 1000 LSTM cells and average prediction with 8 randomly initialized LSTM models are shown below. The MCL training has led each model specializing at one kind of sequences by having lower prediction error. The absolute error plot against the ground truth demonstrates MCL training have lower prediction error.
Algorithm 1 Coordinate Descent for MCL training of LSTM

1: Dataset $D = \{x_i\}$, SGD parameters $\lambda, \eta$
2: LSTM model parameters $\theta_1, \cdots, \theta_M$
3: Initialize $\theta_1, \cdots, \theta_M$ with pre-trained models
4: $t \leftarrow 0$
5: while not converged do
6:   $t \leftarrow t + 1$
7:   sample batch $B \subset D$
8:   procedure FORWARD PASS
9:      for $x_i \in B$, compute each model’ loss:
10:     $l(x_i, \theta_m(x_i))$ defined in Eq. 4.3
11:     Update assignment variable $p_{im}$ as:
12:     $p_{im} = 1[[m = \arg\min_m l(x_i, \theta_m(x_i))]]$
13:     $L_m = \sum_i p_{im} l(x_i, \theta_m(x_i))$
14:   end procedure
15:   procedure BACKWARD PASS
16:      for each $\theta_m$ apply gradient descent as :
17:      $\theta_m \leftarrow \theta_m - \eta \nabla L_m - \lambda \Delta \theta_m$
18:   end procedure
19: end while

all models using each subset by minimizing the mean square error loss using back propagation through time and SGD with a learning rate of $2 \times 10^{-3}$ and momentum of 0.9. Dropout is applied only on non-recurrent connection as suggested [87]. We only train one epoch for each to ensure sufficient diversities between models. We then train all 8 models jointly using the MCL method described in Section 4.2.1 and perform early stopping base on error of the validation set.

Each LSTM model has the same structure as [67], with two LSTM layers each with 1000 nodes. For MCL training, we use 4 Nvidia k80 GPUs in a cluster for training. Since the loss function is coupled with all models and could not be trained in a sequential manner. To enable our experiment scale, we use Message Passing Interface (MPI) standard to enable high speed GPU communication. Each GPU loads two models. As a comparison to MCL training, we also train three benchmark
models. The first benchmark model consists of two LSTM layers, each with 1000 nodes. The second benchmark model has 3000 nodes each layer. The second benchmark model has roughly similar amount of parameters as the ensemble with 8 LSTM models. We also train another benchmark of 8 random initialized 1000 nodes LSTMs and use the average of the prediction results by all 8 models as the final predicted signal.

Sample prediction sequences of testing datasets are shown in Fig. 4.2. Model 8, 7 and 4 are models that have the lowest reconstruction errors on those sequences respectively, and the best model in terms of prediction accuracy also have the lowest reconstruction error in the case shown in here. This shows the model diversity trained with MCL. The prediction accuracy against time comparison is shown in Fig. 4.3. The PSNR is defined as:

$$PSNR = 10 \times \log_{10} \left( \frac{(max_I^2)}{MSE} \right)$$

Where MSE is the mean square error of prediction frames against ground truth frames and $max_I$ is the maximum intensity of the dataset. The oracle selection shown in Fig. 4.3 uses the model that has the lowest prediction error. Since ground truth future frames are not available during inference, such selection mechanism is not practical in reality. The reconstruction-error based model selection chooses the model that has the lowest reconstruction error. The short term prediction accuracy between oracle selection and reconstruction-error based selection are roughly the same, but the accuracy of the latter drops faster than oracle selection as the prediction horizon increases. Even so the reconstruction-error based selection still beats the closest benchmark of average prediction with randomly initialized ensemble by
Figure 4.3: Peak signal noise ratio (PNSR) against prediction time with different methods of test set.

a large margin.

From Table 4.1, it is clear that the 3000 nodes LSTM model is worse than other benchmarks. Because the model does not have any structure to exploit the multi-cluster nature of neural activities, simply adding more nodes makes the number of parameters to be trained grow in an exponential manner. It is less likely to converge to a good local minimum as such model is prone to overfit the training set.

4.2.3 Model selection as classification

To further enhance from reconstruction-based selection, we train a multilayer perceptron (MLP) classifier to select which model to use for prediction. The classifier takes the concatenated LSTM hidden features at the last input frame from all models as input and output the probability of the best LSTM model to use as predictor. The input to MLP classifier is 8000 (1000 dimension feature per model).
<table>
<thead>
<tr>
<th>model</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCL with oracle selection</td>
<td>32.2626</td>
</tr>
<tr>
<td>MCL with classifier selection</td>
<td>31.2767</td>
</tr>
<tr>
<td>MCL with reconstruction-error based selection</td>
<td>31.0636</td>
</tr>
<tr>
<td>Using average of prediction by separately initialized ensemble</td>
<td>29.0722</td>
</tr>
<tr>
<td>Single model with 1000 nodes</td>
<td>28.3495</td>
</tr>
<tr>
<td>Single model with 3000 nodes</td>
<td>25.8128</td>
</tr>
</tbody>
</table>

Table 4.1: PSNR over all predicted frames with different methods for test set. The MCL training and random initialization both have 8 models. Each ensemble models have 1000 nodes.

We use batch normalization[27] as regularization and used three fully connected layers. By using the model that is predicted to have the highest probability by the classifier, we obtain a slight improvement compared to using the reconstruction-error based selection shown in Table 4.1.

### 4.2.4 Relationship between trained models with neural activity patterns

In this section, we analyze the potential relationship between different models in the learned ensemble and different neural activity patterns during seizure and non-seizure durations. For each testing sequence, we assigned the model based the oracle selection (i.e. the model with the least prediction error). The difference between seizure and non-seizure stage shows there are essentially different neural activities in these stages. We see that most neural activity patterns during non-seizure periods can be captured by model 3, whereas there are several different clusters of activity patterns during seizure periods, mainly captured by models 3, 4, 6 and 8. We further investigate the types of neuron activities captured by these
Figure 4.4: 4 models prediction specialties in the ensemble trained with MCL. Each example sequence is shown with the vertical order of ground truth, reconstruction/prediction and absolute error.

models. We find that model 3 is good at predicting silent neural activity namely most of neurons will be at resting potential (hence this model is used in both non-seizure and seizure periods). Model 4 is good at predicting neural activities restricted to a small region. Model 6 is good at predicting when most neurons are going into refractory period after action potential. And model 8 is good at predicting moving neural activity patterns. Such patterns are more common in seizure stage, which explains why model 8 are selected more often during seizure stage. Those patterns are shown in Fig. 4.4.
4.3 Long Term Prediction With Multiple Temporal Resolution

Last section describes an ensemble learning approach for neural activity prediction. However the ensemble approach poses a lot of burden on computation resources. And for brain stimulation to suppress seizure, the ensemble prediction result could not render an accurate prediction for sufficiently long time horizon. Here we propose two encoder-decoder-predictor structures, both using multi-resolution representation.

Inspired by diluted convolution in [74], we propose a LSTM network that uses multi-resolution layers. The higher layer skips each temporal connections to create a shortcut, while lower layer is temporally connected and preserves the fine grained information. We also experiment with an explicitly multi-resolution LSTM structure that resembles a temporal pyramid. We demonstrate both multi-resolution representations improve long term prediction compare to a benchmark LSTM.

Learning long term dependencies not only needs an appropriate network structure but also needs a suitable loss function. For video prediction, to overcome blurry predictions caused by using pixel-wise mean square error (MSE), [47] added total variation loss. Video prediction could be consider as a special case of domain transfer, where the past observed frames lies on one data manifold and future frame lies on another one. Adversarial training finds the relationship between these two manifolds. [45, 47] add adversarial loss on top of MSE. But how video prediction benefits from adversarial training are not fully understood. To further understand how adversarial training benefits video prediction, we use a encoder-decoder 3D CNN structure for the discriminative model. The discriminative model uses re-
Figure 4.5: Video prediction framework. The generative model is built using convolutional LSTM [80]. The network flow is represented with solid arrow, whereas the losses for the generative model are represented with dashed arrows.

construction error as its loss rather than KL-divergence measure. This resembles energy-based GAN in [90] versus GAN [17].

4.3.1 Framework

In this section, we describe the general structure of our neural video prediction model. The structure consists of two different models, a generative model and a discriminative model. The generative model first takes past observations of video sequences as input and learns a compact feature representation, from which the generative model then reconstructs the past frames and predicts future frames. We explore different model structures during the experiments, and use convolutional LSTM [80] as basic building block for generative model. The discriminative model structure is nearly the same in all experiments. Its main goal is to determine whether the future frames are generated conditioned on the true past frames. Together with the generative model, these two models are considered as adversarial training [17]. The general structure is shown in Fig. 4.5.
4.3.2 Generative Model

Let \( X = \{x_1 \cdots x_{t+n}\} \) denotes a video sequence, where \( x_t \) denotes current observation, \( x_{t+n} \) denotes the \( n \)th frame in the future to predict. The generative network takes \( \{x_1 \cdots x_t\} \) as input and outputs a sequence \( Y = \{y_1 \cdots y_{t+n}\} \).

In our approach the generative model has an encoder network, a decoder network and a predictor network similar as [67]. These networks all use convolutional LSTM [80] as basic computation module. The encoder network takes \( \{x_1 \cdots x_t\} \) as input and generates a representation \( l \). The decoder network reconstructs \( \{y_1 \cdots y_t\} \) from \( R_{x_1 \cdots x_t} \). The decoder LSTM is set to be a conditional model namely the decoder reconstructs \( y_{t-m} \) from \( y_{t-m+1} \). The predictor network generates \( \{y_{t+1} \cdots y_{t+n}\} \) from \( R_{x_1 \cdots x_t} \). The predictor model is also a conditional model and it conditions on \( y_{t+m} \) to predict \( y_{t+m+1} \). The loss for the generative model consists four parts:

\[
\mathcal{L}_G = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{pred}} \mathcal{L}_{\text{pred}} + \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}
\]

\[
\mathcal{L}_{\text{rec}} = \sum_{i=1}^{t} ||x_i - y_i||_2^2
\]

\[
\mathcal{L}_{\text{pred}} = \sum_{i=t+1}^{t+n} ||x_i - y_i||_2^2
\]

\[
\mathcal{L}_{\text{adv}} = ||\text{Dec}(\text{Enc}(Z)) - Z||_2^2
\]

\( Z \) is a four dimensional tensor of size \( c \times (t+n) \times h \times w \), with \( c \), \( h \) and \( w \) represents channel, height and width of the frame respectively. \( Z \) is constructed by stacking \( \{x_1 \cdots x_t\} \) and \( \{y_{t+1} \cdots y_{t+n}\} \) in time order. \( \mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{pred}} \) are the pixel domain loss for the reconstructed frames and predicted frames respectively. Whereas \( \mathcal{L}_{\text{adv}} \) is the adversarial loss from the discriminative model.
Figure 4.6: Understanding the benefit from adversarial training: The input to the discriminative model is either true history with true future or true history with predicted future. Third and fifth row of each example shows the activation of second to last layer output across all channel. The activation by true data is distributed almost evenly in both space and time domain to reconstruct the entire sequence. The activation by the sequence with predicted future however concentrates on spatial and temporal inconsistencies. For example, in the first sequence, the discriminative model finds the inconsistency in the last few frames.

4.3.3 Discriminative Model

Generative adversarial network (GAN) were introduced by [17], where image is generated from random noise by using two networks trained in a competing manner. The discriminative model in [17] minimize the KL-divergence between true image distribution and generated image distribution. The original GAN structure suffers from convergence problem and collapsing mode [3, 60]. To solve those problems, [60] introduce several techniques including feature matching, minibatch discrimination and historical averaging. [90] use image reconstruction loss instead of KL-divergence loss. [55] derive a stable deep convolutional GAN structure by modifying modules in both generator and discriminator. [3] show the true data distribution and generative data distribution manifolds in high dimensional space hardly have any overlap, and Wasserstein distance is better compare to other distance measures for non-overlapping distribution. [3] achieved the state of art performance for image generation.
For adversarial training for domain transfer problem, where one generates a sample in target domain condition on the data in source domain. In domain transfer unlike GAN, whose generative model could easily suffer from mode collapsing [55, 60], overlaps between source domain and target domain manifold is easier to find. [40, 45, 47, 69] all used modest model structure to perform domain transfer task and achieved good performance. Video prediction could be considered as a domain transfer problem, where the past frames embedding lies on one manifold and future embedding lies on another manifold. [45] concatenates the LSTM features of past frames and CNN feature of generated frame to train a separate multilayer perceptron. [47] uses a multi-scale 2d convolutional network, the discriminative model stacks all input frames in the channel dimension and output a single scalar indicating whether the video frames are generated or from ground truth future. But both networks fail to model the temporal correlation between frames explicitly.

More importantly, it is not fully understood how adversarial training benefits video prediction. To exploit the temporal dependancies, we use an auto encoder and decoder 3D CNN structure as our discriminative model. The discriminative model uses the energy as the loss function. Energy-based model finds compact representation for the sequence which lives on a low dimension manifold. [90] demonstrate the energy-based GAN training has advantage over GAN for image generation. Another benefit of using encoder-decoder structure is by mapping the activation into pixel space, it helps understanding how adversarial training benefits video prediction. Figure 4.6 shows the activation in the second to last layer of the discriminative model when provided different input. The loss for discriminative
model is:
\[
\mathcal{L}_D = \|\text{Dec}(\text{Enc}(X)) - X\|^2_2 - \|\text{Dec}(\text{Enc}(Z)) - Z\|^2_2
\] (4.5)

The Dec and Enc in Eq. 4.5 refers to the encoder and decoder in the discriminative model.

### 4.3.4 Multi-resolution Representation

#### 4.3.4.1 Benchmark Network

First we introduce the benchmark ConvLSTM model, which is a two layer ConvLSTM structure shown in Fig. 4.7(a). In the benchmark model, each convolution LSTM layer uses a convolution kernel of size $5 \times 5$. For the predictor and decoder convLSTM in the model, the output of both layers go through a deconvolution layer. The deconvolution layer uses kernel size of $1 \times 1$ and outputs a frame, which is essentially a weighted average of all input feature maps followed by a \text{tanh} function.

#### 4.3.4.2 Multi-resolution LSTM

In this approach, in general, we generate $k$ temporal scales of the training sequences. The original sequence constitutes scale 0, and the upper scales are recursively down-sampled from the lower scale by a factor of 2. The top scale (coarsest resolution) works in the same way as the benchmark network over its samples only. The lower scale considers both the samples in that scale as well as the interpolated samples from the upper scale. We only present the 2-scale case here for simplicity. In order to avoid delay, we use the simple averaging of the
current and the previous sample in the lower scale as the anti-aliasing filter for downsampling. Specifically let $x_0^i$ represents the true video frame at time $i$. Scale 1 signal at even time samples is produced by:

$$x^1_i = \frac{x_{i-1}^0 + x_i^0}{2}, i = \text{even}$$ (4.6)

To interpolate the odd samples at scale 1 from even samples, we use simple averaging interpolation filter. The interpolated signal from scale 1 is

$$u(x^1)(i) = \begin{cases} 
  x^1_i & i = \text{even} \\
  \frac{x^1_{i-1} + x^1_{i+1}}{2}, & i = \text{odd}
\end{cases}$$

As shown in Fig. 4.7(b), we first predict even samples at scale 1. We then interpolate the odd future samples from the predicted even samples, to generate all predicted samples, $u(y^1)(t + i)$, from scale 1. We then predict samples at scale 0 using both past samples at scale 0 and current predicted sample at scale 1. Specifically, we predict samples at $t+\hat{i}$ using the features learned up to time $t+i-1$ (from both scales), the actual or predicted sample at $t+i-1$ at scale 0, as well as the predicted sample at $t+i$ at scale 1, i.e., $y_{t+i}^0 = G(f_{t+i-1}, y_{t+i-1}^0, u(y^1)(t+i))$. The two inputs $y_{t+i-1}^0$ and $u(y^1)(t+i)$ to the ConvLSTM predictor are simply stacked as two channels at the same time. In each scale, the generative model is trained by minimizing the loss function at that scale ($k = 0, 1$):
\[ \mathcal{L}^k_G = \lambda^k_{\text{rec}} \mathcal{L}^k_{\text{rec}} + \lambda^k_{\text{pred}} \mathcal{L}^k_{\text{pred}} + \lambda^k_{\text{adv}} \mathcal{L}^k_{\text{adv}} \]

\[ \mathcal{L}^k_{\text{rec}} = \sum_{i \in \text{scale } k} || x_i^k - y_i^k ||^2_2 \]

\[ \mathcal{L}^k_{\text{pred}} = \sum_{i \in \text{scale } k} || x_{t+i}^k - y_{t+i}^k ||^2_2 \]

\[ \mathcal{L}^k_{\text{adv}} = || \text{Dec}(\text{Enc}(Z^k)) - Z^k ||^2_2 \]  

(4.7)

where \( Z^k \) is a four dimensional tensor by stacking the true past frames and predicted frames for scale \( k \) in time order. The illustration of multi-scale structure is shown in Fig. 4.7(b). In each scale, the LSTM network has exactly the same two-layer ConvLSTM structure as the benchmark model, except the input frame of scale 0 have twice the number of channels compared to scale 1: half from the current scale and another half from the upper scale.

### 4.3.4.3 LSTM with Multi-resolution layer

For time series prediction because the temporal correlation is high, in order to achieve a larger receptive field with the same amount of parameters, [74] uses diluted convolution where the convolutional filter in higher layers of the CNN network are structured with zero coefficients every other connections.

Inspired by [74], we propose a LSTM network that has multi-resolution layers. The network have a higher layer and a lower layer. The fine-grained temporal resolution is preserved by the lower layer. The higher layer of the convolutional LSTM model use a skip temporal connection shown in Fig. 4.7(c). Compare to the lower layer, the higher layer creates a temporal highway, which alleviates the vanishing gradient problem. Different from the benchmark network shown in Fig. 4.7(a), the deconvolution layer in multi-resolution layer network (Fig. 4.7(c)) use different pa-
Figure 4.7: Benchmark network, multi-resolution LSTM and LSTM with multi-resolution layers. The multi-resolution LSTM has two scales, and in each scale it has two layer structure. Only one layer is drawn per scale for simplicity. The dotted box represents the predicted frames and $\hat{y}$ represents linear interpolated frames.
<table>
<thead>
<tr>
<th>Generative model</th>
<th>ConvLSTM 64-64</th>
<th>ConvLSTM 64-64</th>
<th>Multi-resolution LSTM 64-64, 64-64</th>
<th>LSTM with multi-resolution layer 64-64</th>
<th>ConvLSTM 128-128</th>
<th>ConvLSTM 128-128</th>
<th>Multi-resolution LSTM 128-128, 128-128</th>
<th>LSTM with multi-resolution layer 128-128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters in the generative model</td>
<td>4123266</td>
<td>4123266</td>
<td>4123266 and 8265732</td>
<td>4123254</td>
<td>15619330</td>
<td>15619330</td>
<td>15619330 and 31277060</td>
<td>15649844</td>
</tr>
<tr>
<td>Discriminative model number of feature maps per layer</td>
<td>None</td>
<td>32,32,4,32,32</td>
<td>32,4,32 and 32,32,4,32,32</td>
<td>32,32,4,32,32</td>
<td>None</td>
<td>32,32,4,32,32</td>
<td>32,4,32 and 32,32,4,32,32</td>
<td>32,32,4,32,32</td>
</tr>
<tr>
<td>PSNR of all frames</td>
<td>27.8737</td>
<td>28.3426</td>
<td>28.8903</td>
<td><strong>29.0372</strong></td>
<td>27.9942</td>
<td>28.5317</td>
<td>29.0931</td>
<td><strong>29.1741</strong></td>
</tr>
</tbody>
</table>

Table 4.2: Comparison of the accuracy and number of parameters of all models. 64-64 represents the number of convolution LSTM cells in layer 1 and 2 are both 64. All the convolution LSTM cells uses $5 \times 5$ kernel. The multi-resolution LSTM structure has two scales, each scale has two layer convolution LSTM cells. 64-64, 64-64 means each scale uses a 64-64 two layer LSTM. Model 1 and 5 are trained with $l_2$ loss alone.

Parameters to predict. In our implementation, the deconvolution layer is performing $1 \times 1$ spatial convolution on the feature map outputs, and the increase number of parameter compare to Fig. 4.7(a) is almost negligible. The number of parameters used in different models are shown in Tab. 4.2.

### 4.3.5 Results

For the discriminative model, we use a 3D convolutional neural network with an encoder-decoder structure. The encoder uses three 3D strided convolution layers with all layers using $7 \times 7 \times 7$ convolutional kernel. The decoder uses three 3D strided deconvolution layers with all layers with the same receptive field. We use batch normalization [27] and leaky ReLU [23] except for the last layer. For multi-resolution network, the discriminative for the higher scale uses only two 3D convolution layer and deconvolution layer as the sequence length is reduced by 2.

We present results for several models. The weights for reconstruction error and prediction error are set to be $\lambda_{rec} = 1, \lambda_{pred} = 1$ in all models. For models
Figure 4.8: PSNR of predicted frames against prediction time. The benchmark model, benchmark model with adversarial, multi-resolution LSTM, LSTM with multi-resolution layer correspond to models 5,6,7,8 respectively in Tab. 4.2. LSTM with multi-resolution layer has a better long term prediction accuracy compared to other models. The PSNR is obtained by first computing MSE by averaging squared errors over all pixels over all frames and all sequences, and then converting the resulting MSE to PSNR.
use adversarial training, we set the weight $\lambda_{adv} = 0.1$. For multi-resolution LSTM network, the weights are set the same in different scale. The discriminative model and generative model are trained both use Adam algorithm[36] both with learning rate of 0.001 decreasing by a factor of 10 halfway through training. To avoid exploding gradient for generative model, we perform gradient clipping by setting the $l_2$ norm maximum at 0.001. In all adversarial training case, the discriminative model is updated once every two iterations. During testing stage, the observation sequences lasts 16 frames and the predictor generates 16 future frames based on the observed frames. Sample prediction comparison are shown in Tab. 4.2. The adversarial training brings improvement compared to using $l_2$ loss alone. It is interesting to note that even the PSNR is based on $l_2$ metric, adding adversarial training into the loss function gets better prediction accuracy. Discriminative model helps generative model to learn the long term dependencies. The further the prediction the more significant is the accuracy gain from adversarial training, which is shown in Fig. 4.8. Comparing among all structures using adversarial learning, LSTM with multi-resolution layer and multi-resolution LSTM both have a significant gain compared to the benchmark model. Even though the multi-resolution LSTM achieved more gains for prediction up to 10 frames ahead, the LSTM with multi-resolution layers take over after 10 frames. PSNR increase by using LSTM with multi-resolution layer at $16th$ frames gets as high as 1.92 dB compared to the benchmark model. This is remarkable as the LSTM with multi-resolution layers have about the same number of parameters as the benchmark model. In Fig. 4.9, we show sample results of different models.
Figure 4.9: Prediction result comparison between different methods: generative model, adversarial training, multi-resolution LSTM and LSTM with multi-resolution layers correspond to model 5,6,7,8 respectively in Tab. 4.2.
4.4 Conclusion

In this chapter, we have described two different approaches for neural activity prediction. Observing that there are multiple clusters of neural activities, we propose an extension of MCL from CNN to LSTM models. The MCL solves the assignment problem jointly with the loss minimization problem. The MCL has enabled a significant improvement in video prediction accuracy compared to averaging the predictions by separately trained LSTM models. Some of the models indeed are found to be able to model different motion patterns in the neural dataset. The second approach uses two different multi-resolution representations for long term video prediction. The proposed LSTM model with multi-resolution layers creates a temporal highway in the upper-layer to capture the long-term dependencies between video frames. The second option uses two scale multi-resolution LSTM. We compare the performance of these two approaches against single resolution benchmark model and demonstrate the advantage of using multi-resolution representation of LSTM. We also demonstrate that all models benefit from energy-based adversarial training which is accomplished by using a 3D CNN based encoder-decoder structure. Comparing these different two approaches, because the backbone model for multi-resolution representation uses convLSTM as build block, there is a significant performance advantage vs. the earlier ensemble learning work which use LSTM. But those two works could be combined to get better prediction accuracy.

There are several avenues for future work. In the first part, we focused on training ensemble of LSTM models. The ensemble has M times more parameters than a single network. Work in [24] has shown how to transfer knowledge from an large model into a smaller, distilled model. Yet how to distill knowledge in the specialists back into a single network has not been shown. On the second part,
we have demonstrated the long term prediction accuracy could benefit from multi-resolution representation. It is also interesting to see whether the accuracy could be further boosted by using external memory network, which has been shown to have the ability to preserve information for a much longer time horizon. [33, 61].
Chapter 5

Pixel-wise Object Tracking

Independent from neural activity prediction, in this chapter, we describe novel pixel-wise visual object tracking framework based on convolutional neural network and recurrent neural networks. The proposed framework is trained to track any anonymous object in a noisy background. The framework consists of two submodels, a global model and a local segmentation model. The global model generates a region of interests (ROI) that the object may lie in the new frame based on the past object segmentation maps; while the local model segments the new image in the ROI. Each model uses a LSTM structure to model the temporal dynamics of the motion and appearance, respectively. To circumvent the dependency of the training data between the two models, we use an iterative update strategy. Once the models are trained, there is no need to refine them to track specific objects making our method efficient compared to online learning approaches.
5.1 Introduction

Provided with a object of interest at the first frame, visual object tracking is a problem of building a computational model that is able to predict the location of the object in consecutive frames. A robust tracking algorithm should be able to tackle some of the common issues including: target deformation, motion blur, illumination change, partial occlusion and background clutters. Most existing algorithms uses online learning by building a discriminative model to separate object from background. The feature extractor is the most important component of a tracker, using appropriate features could drastically boost the tracking performance. Many recent tracking-by-detection approaches [26, 52, 78] are inspired by methods for object detection [16, 56, 57] and fully embrace the features learnt from deep convolutional neural network. We recognize that existing CNN based feature extractor increases the performance and robustness of the tracking system, yet how to extend the deep neural network for visual object tracking has not been fully investigated. In this chapter, we tackle object tracking as a time-series prediction problem, in particular we want to give a pixel-wise foreground-background label for consecutive frames.

Segmentation-based tracking algorithms [2, 6, 29, 64, 85] have advantage over detection-based algorithm for handling a target that undergoes substantial non-rigid motions. Many of them [2, 6, 64] rely only on pixel-level information and hence fail to consider semantic structure of the target. [85] uses Markov Chain on superpixel graph, but information propagation through a graph could be slow depending on the structure. [29] uses a encoder-decoder structure which shares some similarity with ours, but they rely on optical flow and markov random field, which limits the segmentation speed to around 1 fps and the method has only
been demonstrated effective on high image quality dataset as DAVIS[53]. The encoder-decoder structure is widely used in deep learning systems [37, 46, 54, 82]. [54, 82] uses deconvolution for image segmentation and contour detection. We use a decoder to directly perform pixel-wise classification (object or not) on a video from features learnt from an encoder.

To consider the target appearance variation in object tracking, several recent trackers embed CNN into their frameworks. Specifically [7, 70] modify siamese network structure for visual tracking purpose. [79] trains a CNN using Imagenet data and transfers rich features learnt to a new object sequence by updating the network in an online manner. [50] trained a multi-domain network and have separate branches for different domain sequences. The domain specific layer needs to be refined for each sequence. One major limitation of the aforementioned methods is that they lack mechanism to jointly model spatial-temporal traits of the object. [15, 32, 83] propose to solve tracking problem as sequential position prediction by training RNN to model the time series. [15, 32] uses RNN to model the temporal relationship among frames, but they only conducted experiments on synthesized data and did not demonstrate competitive result on challenging dataset like VOT [38]. [59] uses convolutional LSTM [80] (convLSTM) to perform instance segmentation on single image. By spatial inhibition with an attention mechanism, they demonstrate compelling result on VOC Pascal dataset [13]. [83] uses convLSTM to model object feature variations, and their object detection mechanism use similar convolution structure as [7]. In particular, they perform convolution between current frame region of interest (ROI) with past frame predicted exampler frame, which is a tight bounding box prediction. The highest correlation region is the predicted result (i.e. exampler frame) for current frame. By performing convolutional
Figure 5.1: Pixel-wise object tracking framework. The network consists of two sub-modules: a local and a global model, working in a closed loop. At time step $t$, a resized full resolution binary image $\hat{Z}(t - 1)$ is feed into the global model. In inference time, this binary image is the predicted segmentation map acquired from the local model at frame $t - 1$. The global model then roughly predict where the object would appear in frame $t$ based on past segmentation maps and generate a region of interests (ROI). The cropped image in the ROI at frame $t$ is then fed into the local model for segmentation.

operation between exemplar frame with a region of interest (ROI), the output of their system is too coarse for finer-grained pixel labeling.

## 5.2 The overall framework

Our goal is to build a pixel-wise object tracking framework for all possible image resolutions and aspect ratios. Models will be trained in an offline supervised setting. Once the offline training process is finished, there is no need to retrain the network. Segmentation of the full-resolution image would require a large amount of computation resources for real time application. In addition, scaling original image to a fixed size could destroy the semantic information and appearance features of
a small object relative to the image size. To overcome these difficulties, a global model is used to predict the rough location of the object based on the past object segmentation maps. We then crop a region of interest (ROI) from the original image and perform segmentation on the ROI. The network structure is shown in Fig. 5.1.

The model runs in a close loop during inference time. At time step $t$ the global model takes a fixed size segmentation map as input, which is the resized version of a predicted full resolution segmentation map derived at $t - 1$. We use several layers of convolution and pooling to reduce the dimensionality of the image. The resulting features are fed into a convolutional LSTM [80] to fully exploit the temporal variation characteristics of the past segmentation maps. To allow different ROI sizes for the local model (necessary to handle different object sizes and object size variation due to motion in the depth direction), another fully connected layer takes convolution LSTM output as its input to estimate the spatial transformation parameter $\theta$ (including translation and scaling) for the ROI locator, which applies the transformation on a reference anchor box $G$ to generate the ROI in the raw frame $X_t$. As the input segmentation map to global model is resized to $224 \times 224$, we inject the aspect ratio of the original image $\gamma$ into the last fully connected layer of the global model for generalization power among video sequences with different aspect ratios.

At time step $t$ the local model receives a ROI image $x_t$ cropped from the full resolution image $X_t$. A pretrained VGG is used to extract features from the ROI image. These features are then fed into a convLSTM to model for appearance shift. Then the output of convLSTM goes through a deconvolution layer to generate the local segmentation map (which is a gray scale image, with the value at
Figure 5.2: Local model for object segmentation in a ROI image. The M-CNN and F-CNN are feature normalization layers. $h, c$ are hidden and memory states.

each pixel proportional to the estimated likelihood that the pixel belongs to the tracked object). Based on how the ROI is cropped from the full resolution image, the full resolution segmentation map is interpolated accordingly from the ROI segmentation map. Here we assume the ROI encloses the entire object hence all pixels outside the ROI are set to zero.

Global model and local model are trained alternatively in an end-to-end supervised manner. Once the offline training process is finished, there is no need to online finetune the network based on the appearance of the target object, as in some prior work [49, 50, 79].

5.3 Local Segmentation Network

5.3.1 Framework

The network structure for our local model is shown in Fig. 5.2. We consider the pixel-wise object tracking as a time series prediction problem. At each time $t$,
an input ROI $x_i$ is first processed by a pretrained convolutional network. As [88] has shown, high layer features of a trained CNN bears more semantic information whereas low layer outputs bears more appearance information. For genetic visual tracking, the features should be robust enough to work with many different object categories, and also be able to discriminate object instances from the same object class. Lower level features could be more helpful for such a task. However, using very low level features from a pretrained network could drastically increase the computation cost for the following layers. Based on these consideration, we use pool4 features from a VGG network [63] pretrained for image segmentation dataset. The weights of this feature extractor are kept the same during training.

The features are then fed into convolutional LSTMs. However, we have found that the resulting network is hard to train because pool4 features are not confined in a certain range. Therefore, we use another small network consisting of two convolutional layers to normalize the VGG features, which use $tanh$ as the last activation function. These parts are denoted as F-CNN in Fig. 5.2. The normalized features then go through a two layer convLSTM. Intuitively the first convLSTM layer models the dynamics of the foreground object as well as the background. And the second convLSTM layer mostly address appearance shift of the target object. The equation we use for ConvLSTM are shown in Eq. 4.2. To get the segmentation map, the output of second layer convolutional LSTM features are then fed into a deconvolution layer.

In equation 4.2, the hadamard product $\circ$ between $W_{cs}$ and $C$ are crucial for learning long term dependencies. It restricts cross-channel information exchange and overcomes vanishing gradient problem. Replacing hadamard product with convolution would not achieve similar performance for time sequence model. On
| Local model   | filter size | channels | stride | | Global model | filter size | channels | stride |
|--------------|-------------|----------|--------| | | | | |
| M-CNN ×2     | 3 × 3       | 1024     | 1      | | layer 1 ×2   | 3 × 3       | 8         | 1      |
| F-CNN ×2     | 3 × 3       | 512      | 1      | | layer 2 ×2   | 3 × 3       | 16        | 1      |
| ConvLSTM ×2  | 3 × 3       | 256      | 1      | | layer 3 ×2   | 3 × 3       | 32        | 1      |
| Decov -1     | 5 × 5       | 128      | 2      | | layer 4 ×2   | 3 × 3       | 64        | 1      |
| Decov -2     | 5 × 5       | 64       | 2      | | ConvLSTM×2   | 3 × 3       | 64        | 1      |
| Decov -3     | 5 × 5       | 32       | 2      | | full 1       | 1024       |           |        |
| Decov -4     | 5 × 5       | 1        | 2      | | full 2       | 3          |           |        |

Table 5.1: Number of filters for each modules in local segmentation network and global attention network. Notation ×2 represents two identical layers that are connected. **Local model:** In M-CNN and F-CNN internal activation functions uses rectified linear unit (relu), where as the output activation function is \( \text{tanh} \). The internal activation function in deconvolution is leaky-relu, the last activation function is sigmoid. **Global model:** every two convolution layer are followed by a pooling operation to reduce the spatial dimensionality. The input to fully connected layer is vectorized output of convolutional LSTM. For the second fully connected layer the input dimension is 1025, where we concatenate the feature from last layer with aspect ratio of the current video clip.

On the other hand, ConvLSTM is not equivariant to translation particularly because of the hadamard product. This means a spatially shifted version of the input image may not lead to an equally shifted segmentation map. As the global model may not always generate the ROIs centered around the object at different frame times, it would be preferred that the local segmentation network has a certain degree of translation equivariance. Although this is one major drawback of using ConvLSTM for object tracking, we have found that with the ROI chosen by the global model, the object tends to fall near the centers of the ROIs in all frames, and our local model can perform well even with small spatial shift between consecutive frames for unseen objects. The detailed number of parameters are shown in Tab. 5.1.

### 5.3.2 Memory Initialization

To start ConvLSTM, we need to initialize the memory and hidden state. Initializing the memory cell to be zeros is one option. But a major drawback of such
approach is the memory cell of recurrent network would need multiple time steps to converge. During this time its hidden connection $h$ is also drastically different from its true distribution. And segmentation could easily fail because deconvolution is directly applied on $h$. A wrongfully predicted local segmentation map would further affect the global model. Moreover, within the first ROI there could be more than one salient object. Without differentiating between these salient objects, the tracking system would not know which object to track and is likely to fail.

Instead of arbitrarily initializing the memory with zero, we train an initialization module that takes the object mask, and the image in a manually chosen ROI in the first frame and generates the initial memory cell state and the hidden state which ideally should capture the appearance features of the object. To overcome the boundary artifact, we use a dilated mask to generate the masked image. In our experiment, we find that instead of applying the object mask in the image domain, applying the mask on the layer right before the pool1 layer in the VGG network would render better performance. We then regress the initial memory and hidden states of ConvLSTM using the concatenated feature. This is done by using another two convolution layers denoted by M-CNN in Fig. 5.2. Similar as [83], we find using a $tanh$ function as the last activation function for M-CNN stabilizes the memory, even though the numerical value of memory cell could go beyond the range of $[-1,1]$. Ideally we want the memory cell to slowly adapt to appearance drift meanwhile while being able to ignore false objects. In Fig. 5.3 we show the memory state evolution under different training strategies. The training strategies would be discussed in the following subsection.
(a) Sequential segmentation result visualization without randomly inserting noisy frames during training.

(b) Sequential segmentation result visualization with randomly inserting noisy frames during training.

Figure 5.3: In each subfigure, each row in vertical order is: the segmentation result overlaid on top of the raw image, first layer convolutional LSTM cells, second layer convolutional LSTM cells. The displayed images are downsampled by 2. Row 1: Both true sequence and inserted frames comes from testing set. Row 2 and 3, we show the top 16 activations out of 256 cells. Note: (i) For ConvLSTM even with memory regression there is still a burn in time for memory to converge. (ii) Memory cells get far noisier in subfigure(a) compare to subfigure(b) after several step. (iii) There is memory cells co-adapt with noisy sequences, which act as action detection (encircled with red rectangle in figure).
5.3.3 Training

Visual object tracking (VOT) [38] dataset is considered one of the hardest dataset for object tracking, because it contains videos in varying resolutions and some of the target objects (e.g. a football) are very small relative to the image size, and some objects undergo significant appearance shifts. The dataset contains 60 video sequences with more than 200 frames per sequence on average. To deal with the limited number of videos, we use 10 fold-cross validation and randomly distribute 60 sequences into 10 data fold. Each fold contains 54 videos in the training set and the other 6 videos in the testing set. Testing videos often contains objects not seen in the training set. For all models training is only done on the training set and we report the average accuracy of our result on the testing set.

The minibatch of sequences are prepared by the following steps:

1. Manually select a frame from a sequence randomly as the initial frame. Initial frame does not contain artifacts including occlusion, motion blur etc.

2. Crop this and all subsequent frames to generate ground truth ROI images. The width of the square ROI is twice the longer length of the object along the horizontal and vertical directions. The ROI width is further truncated to within the range of [56, 672]. In order to train the model to deal with the potential error of the global model, the location is set according to the object mask at frame $t - 5$. Resize all ROI images to $224 \times 224$, equal to the input image size for the VGG network.

3. Perturb the resulting ROIs in both positions and size randomly. Random scaling is set in the range of [0.9, 1.1] and spatial shift $[-10, 10]$ pixels. We denote the resulting sequences of ROI images for all training videos (each
video contains only one object) as $x_{0:T}$, and the sequences of ground truth segmentation masks within the ROI as $z_{0:T}$.

4. For each training video $i$, replace the ROI image at a randomly chosen time $t_i$ with the ROI image for another randomly chosen video $j$ at another time $t_j$. The ground truth segmentation maps for such ROI images are set to all zero. Motivation for this step is explained in Sec. 5.3.4.

After these steps, each training sample is a pair of video clips (the ROI image sequence and the ground truth ROI mask sequence for a training video), we then solve the following optimization problem in Eq. 5.1, where $L(\hat{z}_{1:T}, z_{1:T})$ and $V(\hat{z}_i)$ are element-wise cross entropy loss and image total variation loss respectively. $\Phi$ defines the local segmentation network and $\theta$ is the parameters belonging to $\Phi$. We use the image total variation loss $V$ to discourage the resulting segmentation map to contain multiple small isolated components. We intentionally avoid applying more complicated post-processing on the segmentation map using approaches like markov random field (MRF) to both reduce the computation complexity at the inference time and to enable end-to-end training. $\beta$ is a thresholding term that stabilizes the training procedure especially at the beginning stage. $\beta = 1000$ and $\lambda = 1e-4$ was found to achieve the best performance.

$$\min_{\theta} L(\hat{z}_{1:T}, z_{1:T}) + \lambda \min(\beta, \frac{1}{T} \sum_{i=1}^{T} V(\hat{z}_i))$$

$$\hat{z}_{1:t} = \Phi_\theta(z_0, x_{0:t})$$

$$L(p, y) = \sum_i -(1 - y_i) \log(1 - p_i) - y_i \log(p_i)$$

$$V(y) = \sum_{i,j} |y_{i+1,j} - y_{i,j}| + |y_{i,j+1} - y_{i,j}|$$
5.3.4 Comparison and analysis

We found step 4 in the data preparation is crucial for the success of the local segmentation network. Without step 4, the convolution LSTM merely learns a frame by frame saliency detection. In Fig. 5.3, we compare the memory state evolution for two networks with and without step 4 on unseen sequences. The module learnt with step 4 is much more stable especially when there are multiple salient objects in the same ROI.

We further conducted another experiment to demonstrate the benefit of using convLSTM. In this experiment, we fine tune a pre-trained segmentation network using fully convolutional neural network [44] structure for the local segmentation task. The FCN is pretrained on COCO dataset [43]. The feature extraction part of our local model use the same model up to pool4. When using the FCN segmentation network on a testing video, we fine tune it on the first frame of testing video clips with small learning rate and few iterations, and apply the refined model to subsequent frames. We compare the segmentation accuracy for the following 32 frames in all testing video clips. For convLSTMs trained with and without step 4, we don’t fine tune based on the first frame of the testing video. We report the ROC curve and framewise IOU curve for 1200 randomly sampled video clips in the testing set in Fig. 5.4. True positive rate and false positive rate is defined at the pixel level. Framewise IOU is defined as in Eq. 5.2. Convolution LSTM trained under both strategies get higher AUC for the ROC curves and the FCN network with refinement during testing stage could not adapt to appearance shift as demonstrated with Fig. 5.4.
Figure 5.4: Comparison between ConvLSTM and framewise segmentation. **Left:** ConvLSTM 1 and 2 represents training strategies with randomly replaced frame and without randomly replaced frame respectively. **Right:** IOU comparison per frame for convolutional LSTM training with randomly place frames vs segmentation.

The better performance using ConvLSTM for local model comes with a price, as analyzed in subsection 5.3.1. ConvLSTM is not shift equivariant, a large spatial drift between consecutive frames could cause loss of tracking. In our observation, spatial shift larger than 30 pixels in the ROI could cause instability in our tracking system. To circumvent this problem, we predict the ROI using a global attention network.

### 5.4 Global Attention Network

To predict where the ROI should be located in the current frame based on predicted segmentation map in the history, one naive way is to use weighted av-
verage of the past predicted location directly to decide where the ROI should be cropped. However during the test time, the local predictor might make prediction mistakes caused by light condition, drastic appearance change, motion blur etc. Such mistakes could then cause the global model to locate a wrong ROI for the next frame. Overtimes, the ROI could drift away from the correct object location. Therefore, we need to develop a rather robust global model that can handle such problems. The ROI is specified by a spatial transform acting on a fixed anchor (a square region) $Z^A$. We apply a LSTM on the past global segmentation maps to generate features that are then fed to a spatial transformer network to determine the transform parameter. Our spatial transform network is a special form of [28], but the transformation is not applied on the feature map, but on a fixed anchor $Z_A$. The training framework of global attention network is shown in Fig. 5.6.

During training stage, at each time $t$ a fixed size segmentation map $Z_{t-1}$ is feed into the global attention model $\tau$. The network generates a special form of affine transform parameters $\theta$. The spatial transform $T(\theta)$ is applied on $Z^A$, so that the transformed anchor $\hat{Z}^A_t$ maximally overlaps with the ground truth segmentation map in frame $t$, $Z_t$. We want the transformed anchor to enclose as much foreground pixel as possible, and we use a weighted $l2$ loss between $\hat{Z}^A_t$ and $Z_t$. We further add a $l2$ loss term between $\theta_t$ and $\theta_{t-1}$ so that the transformer is temporally smooth. Parameter $\theta$ constrains the transform to only allow spatial shift and resizing. The resizing operation takes consideration of image aspect ratio, so that when cropping the image at the image domain the aspect ratio is not distorted (the ROI on the real image is always a square but with varying sizes). The overall loss function is defined as:
\[
\min_{\phi} \sum_{t=i}^{T} (L(\hat{Z}_t^A, Z_t) + \lambda \|\theta_t - \theta_{t-1}\|_2)
\]

\[
\hat{Z}_t^A = T(\theta_t)(Z_A)
\]

\[
\theta_t = \tau_\phi(Z_{0:t-1})
\]  

(5.3)

The detailed number of parameters of our global model is shown in Tab. 5.1. During training, we observe that recurrent model needs burn-in time to accurately predict the spatial transform. Otherwise it would not utilize the full history of the observations. So we only compute the loss after \(i\)th frame. In our experiment, we find setting \(i = 5\) works best for a total sequence length of 32. To let our model converge faster, in practice we apply a dilation kernel on our input sequence \(Z_{1:t}\) and gradually shrink the size of the dilation kernel until convergence.

However during inference stage, since the model could only utilize the predicted masks by the local model, there is a distribution difference between testing sequences and training sequences. Fig. 5.5 demonstrates the training set and testing set difference. To handle the distribution gap we iteratively adapt our global model and local model. We describe the way to update our model in Sec. 5.5.

5.5 Experiment

5.5.1 Iterative optimization

In addition to preparing local samples as described in section 5.3.3, to handle the observation difference mentioned in Sec.5.4, on each of the data fold we perform our training as following:
Figure 5.5: Demonstration of observation difference between testing set observation and ground truth. Each row shows a sequence temporally downsampled by 4. From top to bottom: input to global in testing sequence, ground truth mask and predicted bounding box location.

Figure 5.6: Training framework for global attention model
1. Evenly separate video sequences of each training set into two subsets. On each subset use the ground truth bounding boxes to prepare a training set for the local model (see section 5.3.3). Train one local model on each subset with early termination with loss function Eq. (5.1).

2. Train the initial global model using sequences of ground truth segmentation maps with loss function Eq. (5.3). To increase convergence speed, we apply dilation operation on $Z_t$ and shrink dilation kernel size every ten thousand iterations until convergence.

3. Use the trained local model 1 from step 1 and global model from step 2 to generate predicted segmentation maps $\hat{Z}_{1:T}$, and ROI images $x_{1:T}$ and ROI segmentation maps $z_{1:T}$ for training data in subset 2. Procedure is described at algorithm 2. Use local model 2 to do the same on subset 1.

4. Update the global model with modified input sequence $\bar{Z}_{0:T} : \{Z_0, \hat{Z}_1, \cdots, \hat{Z}_T\}$ generated by step 3 using Eq. (5.3).

5. Train local model using the ROI image sequence $x_{1:t}$ and segmentation map sequence $z_{1:t}$, which are generated by the updated global model for the entire training set with Eq. (5.1).

### 5.5.2 Time complexity

We evaluate our tracking algorithm on VOT2016 [39] segmentation dataset. We implement our algorithm with tensorflow and test it on a single NVIDIA Tesla K80 with 24G RAM. The inference speeds using the local and global models are
Algorithm 2 Two stage tracking algorithm

Input: Raw image $X_0$, segmentation map $Z_0$, global model $M_1$, local model $M_2$

Output: Predicted segmentation map $\hat{Z}_{1:t}$, ROI image and segmentation map sequences $x_{1:t}$, $z_{1:t}$

1: Crop template $x_0$, $z_0$ with spatial parameter $\theta_0$
2: Initialize the memory of local model $M_2$ using $x_0$, $z_0$
3: for $t = 1, T$ do
4:   Estimate $\theta_t$ using $M_1(\hat{Z}_{t-1})$
5:   Update $\theta_{est}$ as $\theta_{est} = \beta \theta_t + (1 - \beta)\theta_{old}$
6:   Get ROI images $x_t$, $z_t$ from frame $X_t, Z_t$ use $\theta_{est}$
7:   Estimate $\hat{z}_t$ use $M_2(x_t)$
8:   Use $\theta_t$ to fill in $\hat{Z}_t$ with $\hat{z}$

24 ms and 6 ms per frame respectively. With memory initialization and ROI interpolation included, the entire framework still runs more than 20 fps.

5.5.3 Quantitative analysis

First, we compare our local segmentation network using LSTM with FCN segmentation network. Both models are provided with the same ROI sequences. The only difference is FCN network is further fine tuned on the first frame for each of the sequence. We follow the same procedure in Sec. 5.3.3 using the ground truth label to prepare sequences. The only exception here is that we use a exponential weighting on the ROI location center by $L_{new} = (1 - \beta)L_{old} + \beta L_{t-5}$. The decay rate $\beta$ is set at 0.8. Here, we use the object ground truth location at frame $t - 5$ to crop the local ROI so that object is reasonably far apart from the center but still in the ROI. We admit that the test could be still favoring our convolutional LSTM as the location of the object is registered to be close to the center of the template. On the other hand this test demonstrates the upper bound of the proposed object tracking approach, achievable when the global model can accurately locate the
Figure 5.7: Quality results for 8 videos. The prediction are overlaid on top of the original image. The results are based on 2-stage models. The failure case at sequence 2 and 8 are mostly due to a large camera motions.

ROI. The evaluation using the FCN segmentation network, on the other hand, is meant to evaluate the achievable tracking performance by a CNN-based segmentation network, when equipped with a near perfect global tracker. The result is shown in Tab. 5.2.

Next, we evaluate our global model and local model jointly. The inference for the two stage model (denoted by 2-stage ConvLSTM) follows algorithm 2. We also evaluate a benchmark 1-stage model, which replaces the global model with a simple predictor for the ROI center, described in Eq. (5.4). Here $p_i$ is the estimated probability of pixel $i$ belonging to the foreground by the local model for the previous frame. The ROI size is fixed. FCN based tracking uses the same approach for determining the ROI with its own segmentation map. Table 5.3 compares the performance of our 2-stage model, the benchmark 1-stage and the FCN segmentation network.
Table 5.2: Average IOU at different frame length with template cropped close to
ground truth location. For sequences shorter than preset length, we upsample the
testing sequences to the fixed length.

<table>
<thead>
<tr>
<th>Test sequence length threshold set at 0.4</th>
<th>20</th>
<th>40</th>
<th>80</th>
<th>160</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConLSTM</td>
<td>0.4690</td>
<td>0.4405</td>
<td>0.3962</td>
<td>0.3342</td>
</tr>
<tr>
<td>Segmentation</td>
<td>0.3785</td>
<td>0.3582</td>
<td>0.3157</td>
<td>0.2304</td>
</tr>
</tbody>
</table>

Table 5.3: Average IOU at different frame length.

<table>
<thead>
<tr>
<th>Test sequence length threshold set at 0.4</th>
<th>20</th>
<th>40</th>
<th>80</th>
<th>160</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-stage ConLSTM</td>
<td>0.3992</td>
<td>0.3606</td>
<td>0.3201</td>
<td>0.2564</td>
</tr>
<tr>
<td>1-stage ConLSTM</td>
<td>0.3800</td>
<td>0.3400</td>
<td>0.2846</td>
<td>0.2419</td>
</tr>
<tr>
<td>Segmentation</td>
<td>0.2275</td>
<td>0.2058</td>
<td>0.1678</td>
<td>0.1437</td>
</tr>
<tr>
<td>2-stage ConLSTM</td>
<td>0.2485</td>
<td>0.2302</td>
<td>0.202</td>
<td>0.1854</td>
</tr>
<tr>
<td>1-stage ConLSTM</td>
<td>0.2080</td>
<td>0.1926</td>
<td>0.1679</td>
<td>0.1518</td>
</tr>
<tr>
<td>Segmentation</td>
<td>0.1030</td>
<td>0.093</td>
<td>0.0711</td>
<td>0.058</td>
</tr>
</tbody>
</table>

Results in Tables 5.2 and 5.3 demonstrate that the proposed 2-stage convLSTM
architecture is better than a CNN fine tuned on the first frame. Even when pro-
vided with nearly correct location of the ROI in each new frame, the CNN-based
segmentation network could not handle appearance shift as well as our 2-stage
ConvLSTM. Furthermore, our global model performs better than a naive location
predictor. More visual results are shown in Fig. 5.7.

\[
\theta_x = \frac{1}{\sum_i p_i} \sum_i p_i x_i \quad \theta_y = \frac{1}{\sum_i p_i} \sum_i p_i y_i
\]  

(5.4)
Chapter 6

Conclusion

In this thesis, we investigated different aspect of neural activity analysis and prediction and developed a novel approach for pixel-wise object tracking.

6.1 Summary of Main Contribution

This thesis has addressed a number of challenging issues associated with neural activity analysis and prediction and object tracking. Major contributions are summarized below.

First, we have combined advanced video analysis and machine learning algorithms to analyze challenging \(\mu\)ECoG datasets in novel ways. We have developed efficient methods for identifying and localizing the spatial and temporal extent of inter-ictal and ictal spikes through graph filtering and region growing. For those identified spikes we designed a pairwise metric that characterize the similarity between the movement patterns of different spikes. Using this similarity metric, we discovered the lower-dimension manifold features using the Isomap and furthermore discovered different spike patterns using the DPM clustering method on the
manifold features. The clustering result reveals cleaner patterns that have not
been seen before. Leveraging on the discovered patterns, we also addressed the
challenging problem of spike wavefront prediction and seizure detection and pre-
diction. We developed a novel wavefront prediction approach, which makes use of
the prior knowledge of the mean wavefront trajectory of each spike cluster. We
further developed seizure detection algorithms based on the temporal variation of
the spike patterns. The proposed two stage HMM+SVM classifier yielded accurate
seizure detection and possibly prediction, by detecting 49 out of 61 seizures before
the human-labeled seizure onset.

Second, we developed two novel approaches for neural activity prediction. Ob-
serving that there are multiple clusters of neural activities, the first approach used
an ensemble of predictive models. We extended the multiple choice learning from
CNN to LSTM models. The MCL solves the assignment problem jointly with
the loss minimization problem. The MCL has enabled a significant improvement
in video prediction accuracy compared to averaging the predictions by separately
trained LSTM models. Some of the models are found to indeed model different
motion patterns in the neural dataset. We find that using the reconstruction error
as a guideline to select the model to be used for prediction can yield predictions
close to that using an oracle selected model. To enhance long term prediction per-
formance, the second approached adopted a multi-resolution representation. Two
network structures were explored. The first option uses a novel LSTM structure
with multiresolution layers for long term video prediction. The network creates
a temporal highway in the upper-layer to capture the long-term dependencies be-
tween video frames. The second option uses two-scale multi-resolution LSTM.
We compare the performance of these two approaches against single resolution
benchmark model and demonstrate the advantage of using multi-resolution representation of LSTM. Both multi-resolution LSTM and LSTM with multi-resolution layers have better performance than single resolution representation when they all use adversarial training. The long term prediction accuracy using LSTM with multi-resolution layers are much higher than the benchmark models with similar number of parameters. We also demonstrate that all models benefit from energy-based adversarial training which is accomplished by using a 3D CNN based encoder-decoder structure.

Last but not least, we tackle object tracking problem at the pixel level. By providing the beginning frame and corresponding segmentation map, we model the appearance shift as a time series. We propose a novel two-stage model handling micro-scale appearance change and macro-scale object motion separately. The local segmentation model has far better performance compared to a CNN fine-tuned on the object appearance in the first frame. The global model can accurately predict the rough location and size of the object from frame to frame. We demonstrate our novel approach on a very challenging VOT dataset. Finally our model performs pixel-wise object tracking at a reasonable accuracy in real time.

6.2 Future Research

6.2.1 Model Compression for Real Time Application

In chapter 4, we discussed the work of neural activity prediction. In the first part we focused on training ensemble of LSTM models. The ensemble has M times more parameters than a single network. Work in [24] has shown how to transfer knowledge from an large model into a smaller, distilled model. Yet how
to distill knowledge in the specialists back into a single network has not been shown. Without model distillation, the computation resource required for neural activity prediction might be too large for clinical application. Another different approach is model compression. By using a three stage pipeline: pruning, trained quantization and Huffman coding, [22] successfully compressed a CNN by $50 \times$ with no significant loss of accuracy. [89] uses group convolution and channel shuffle to greatly reduce computation cost while maintaining accuracy. Yet, with the limited knowledge of the author, there has not been much work on model compression of RNN mainly because of its recurrent nature. It is interesting to explore whether there is a way to perform model compression for recurrent network structure.

6.2.2 Long Term Prediction with External Memory Network

On the second part of chapter 4, we have demonstrated the long term prediction accuracy could benefit from multi-resolution representation. It is also interesting to see whether the accuracy could be further boosted by using external memory network, which is known to have good performance for preserving information for a much longer time horizon. For neural activity prediction, this might not have much advantage against multi-resolution representation or attention model since the signal has low long term temporal correlation. But using external memory may be helpful. For seizure prediction, as the necessary signal to classify seizure could easily go beyond tens of thousand time step. To build an end-to-end model, external memory network could be a better choice for this application.
Bibliography


[22] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep


[85] D. Yeo, J. Son, B. Han, and J. Hee Han. Superpixel-based tracking-by-


List of my Publications


