Speckle-Based Reservoir Computing & Echo State Networks

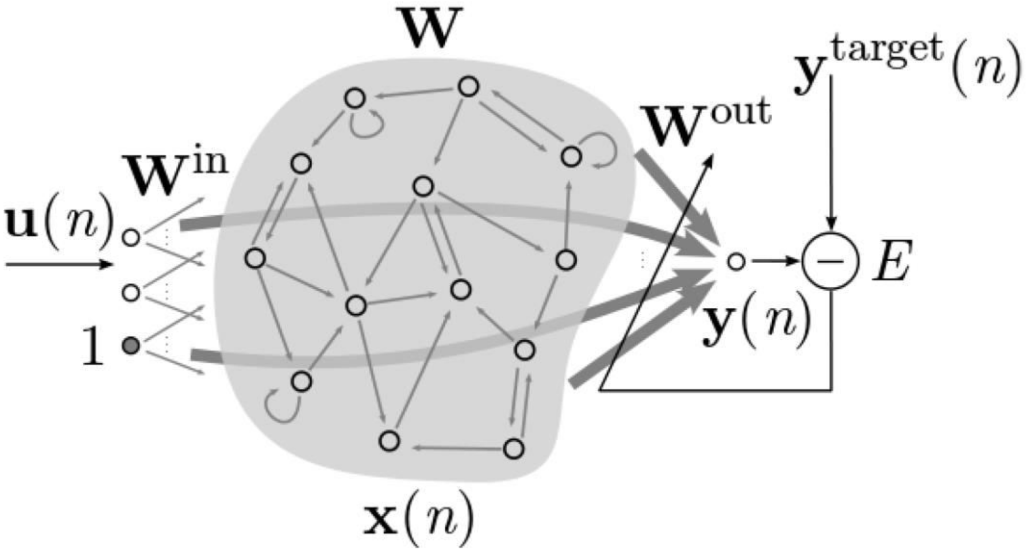
Jacob Pilawa, Uttam Paudel, Marta Luengo-Kovac, George Valley, Justin Shaw

08 August 2019

Overview

What's coming?

- Background of neural networks
- Echo state networks & Reservoir Computing
- "Running" an echo state network
- Optical implementation of an echo state network
- Day-to-day work
- Japanese Language Optical ESN Case Study
- Results and Analysis
- What else is there to do?



TAKEAWAY: *Story of my summer exploring echo state networks and optical computing.*

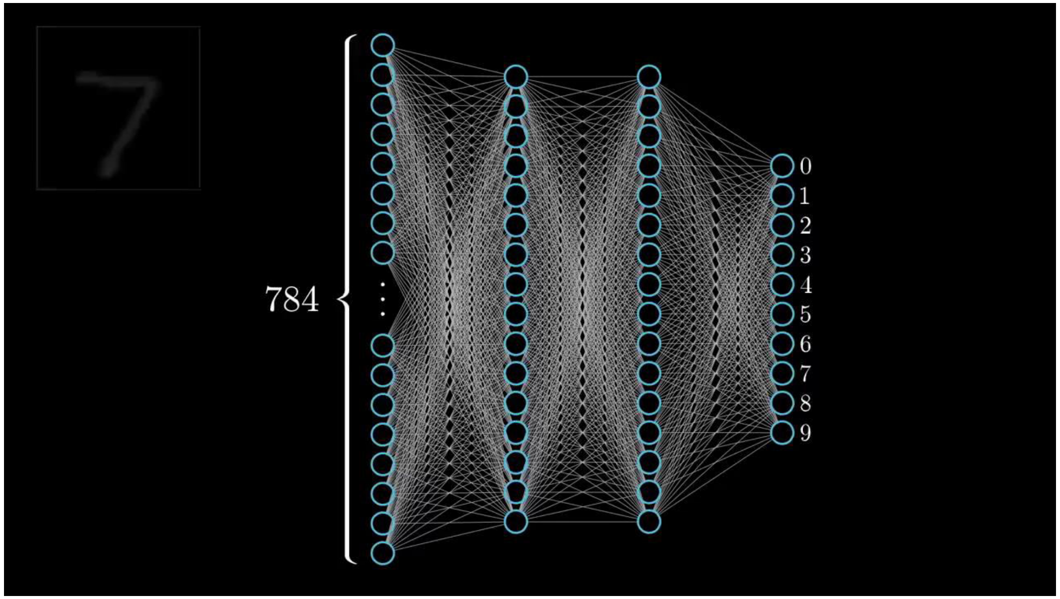
Figure: Melidis et al., 2019

Neural Networks

A bit of background…

- Mesh network of “connected” layers of “neurons” that try to find abstract features in datasets
- Neuron : a thing that holds a number and pathways to other neurons
- Network : neurons from layer to layer are connected and can influence each other

- **Goal** : Train the neural network (i.e., figure out the best set of connections) for a given task so that it can make predictions in the future



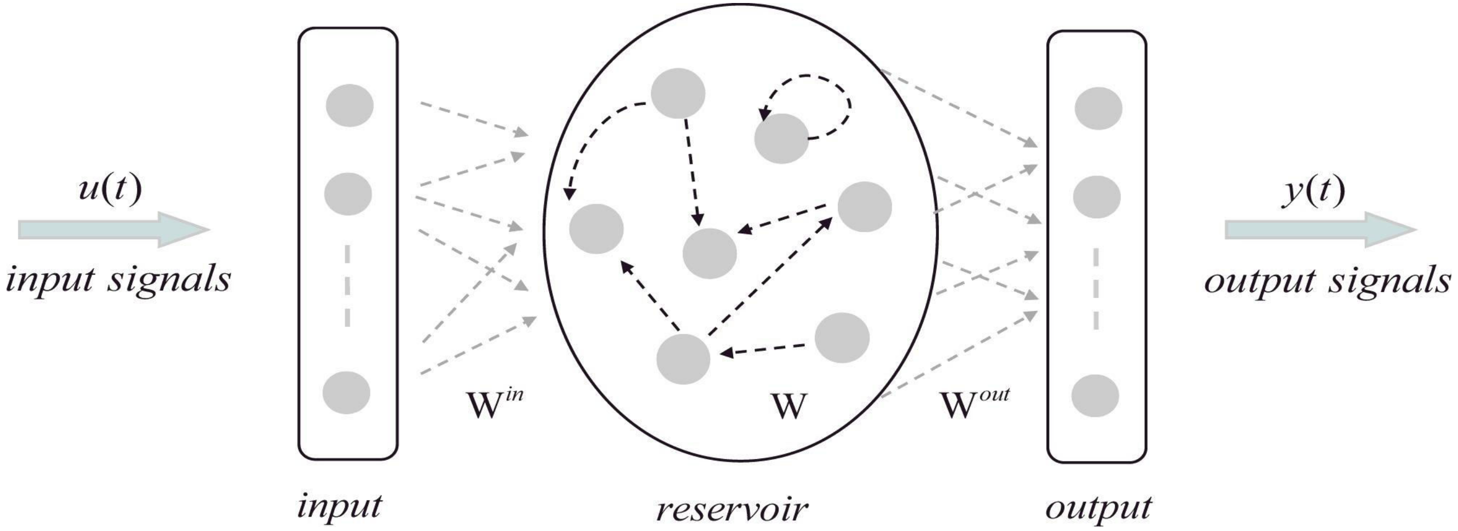
Animation: 3Blue1Brown, YouTube

TAKEAWAY: *A neural network, plain and simply, is a function that takes inputs and maps them to a desired output after it has been “trained” to do so.*

Reservoir Computing / Echo State Networks

Training the output, not the method

- Echo State Networks (ESNs) are a type of reservoir computer (RC), which itself is an extension of neural networks
- **Key Aspects**
  - Connections between neurons are random and fixed whereas traditional neural networks train all connections
  - Huge reservoir size + non-linear transfer function (connections) + Feedback connections = Captures non-linear, temporal dynamics of “real” systems
  - **Training is performed on a single layer of neurons using simple regression techniques (and thus computationally efficient)**

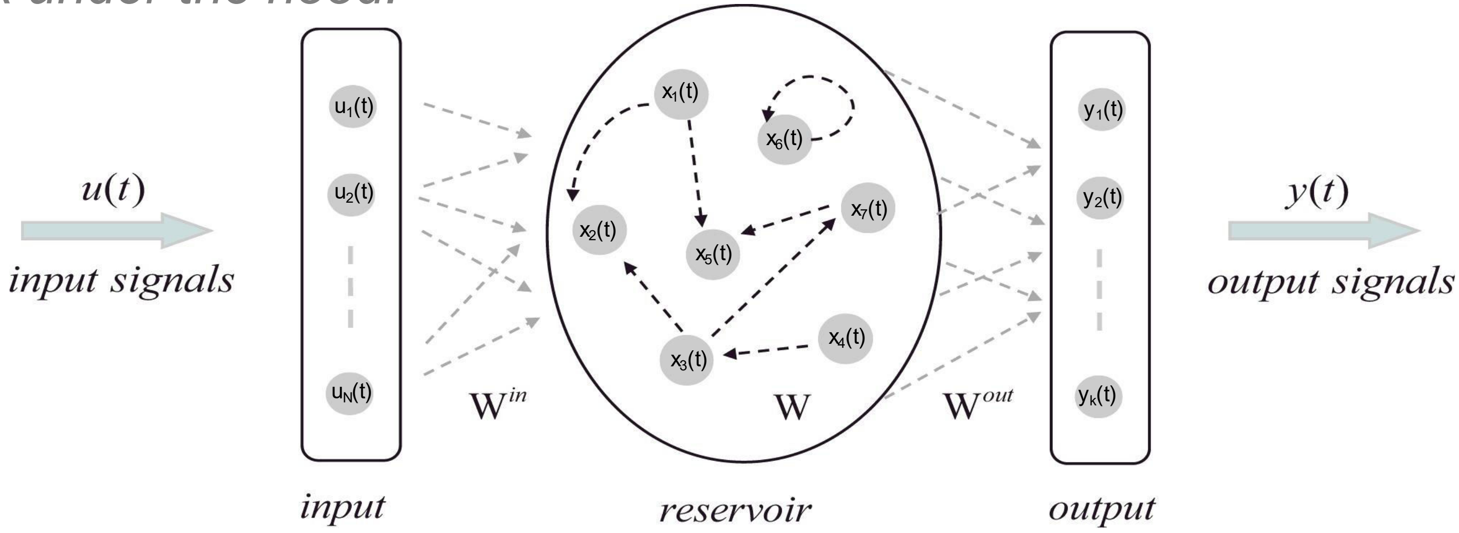


APPLICATIONS: *Speech recognition, time-series predictions, signal classification, denoising, demodulation, on-the-fly signals intelligence, etc.*

Figure: Ma et al., 2016

An Echo State Network in Operation

A look under the hood.



$$\mathbf{x}(t+1) = \alpha \tanh(\mathbf{W}\mathbf{x}(t) + \mathbf{W_{in}}\mathbf{u}(t)) + (1-\alpha)\mathbf{x}(t)$$

$$y(t) = \mathbf{W_{out}}\mathbf{X}(t)$$

1. Generate a large, random reservoir ($\mathbf{W}^{in}$, $\mathbf{W}$).
2. Run every training input u(t) through the reservoir, collecting how the reservoir responds to each input x(t).
3. Compute $\mathbf{W}^{out}$ from these activation states x(t) using linear regression, minimizing the error between y(t) and $y^{target}(t)$.
4. Use the trained network on new input data $u^{test}(t)$, computing $y^{predict}(t)$ using the trained output weights $\mathbf{W}^{out}$.

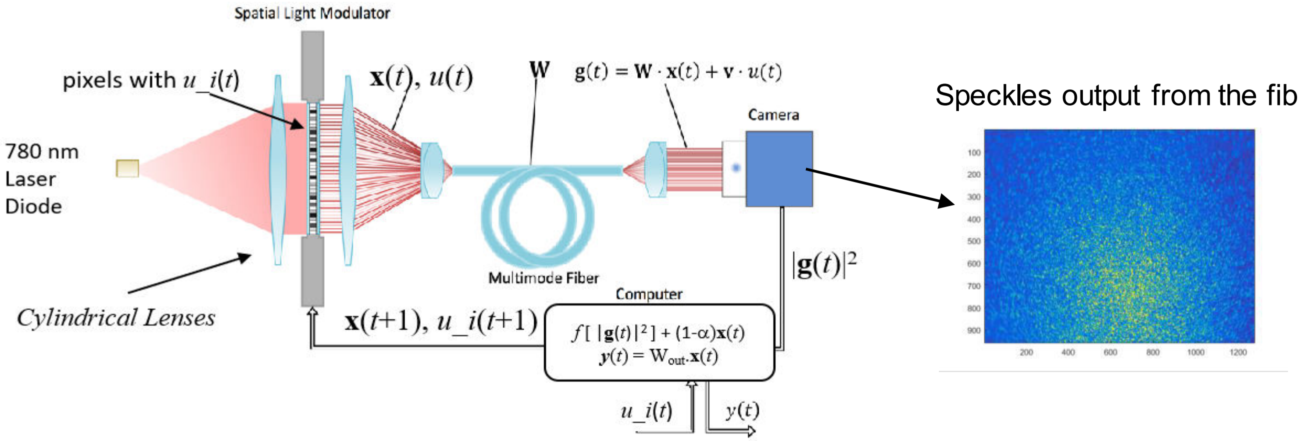
TAKEAWAY: Echo State Networks work by inputting a signal, seeing how the reservoir responds, and making predictions based on that response.

Figure: Ma et al., 2016

Optical Echo State Networks

Implementing neural networks in the lab

MOTIVATIONS + TAKEAWAYS: Using photonics, we can automatically perform the matrix multiplications "at the speed of light." This system implements an echo state network by using a speckle pattern as a substitute for electronic computation.


Spatial Light Modulator
pixels with $u\_i(t)$
$\mathbf{x}(t)$, $u(t)$
$\mathbf{W}$
$\mathbf{g}(t) = \mathbf{W} \cdot \mathbf{x}(t) + \mathbf{v} \cdot u(t)$
Camera
780 nm
Laser
Diode
Multimode Fiber
$|\mathbf{g}(t)|^2$
Computer
Cylindrical Lenses
$\mathbf{x}(t+1)$, $u\_i(t+1)$
$f[\,|\mathbf{g}(t)|^2\,] + (1\text{-}\alpha)\mathbf{x}(t)$
$y(t) = \mathrm{W}_{\mathrm{out}} \cdot \mathbf{x}(t)$
$u\_i(t)$
$y(t)$
Speckles output from the fiber
100
200
300
400
500
600
700
800
900
200
400
600
800
1000
1200

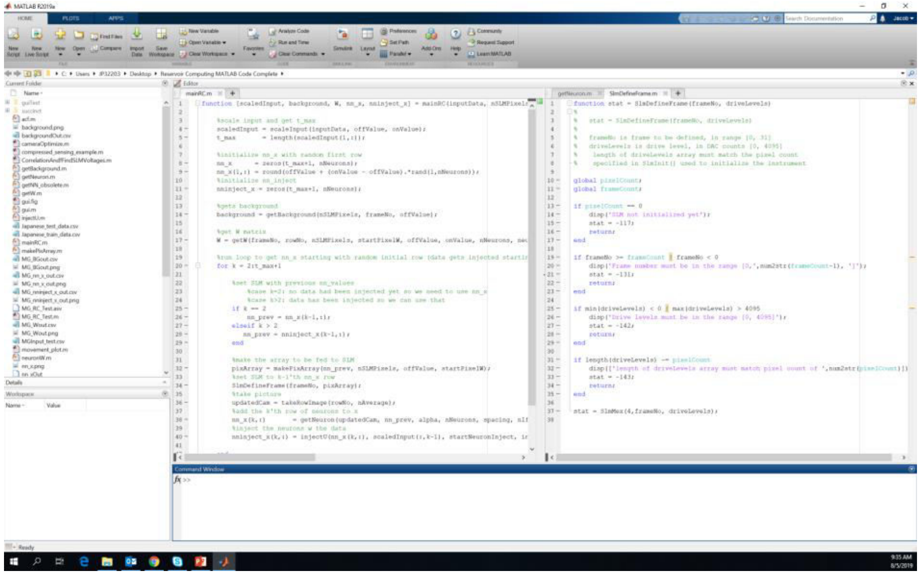
1. Encode the input as voltages into the pixels of the spatial light modulator (SLM).
2. Pass output of SLM into a multimode fiber to generate speckle pattern.
3. Apply non-linearity and feedback fraction of previous neurons to obtain new neuron values from speckle image.
4. Inject these neurons back into the SLM. Store this at each time step to build **X**(t).
5. Calculate **W**$^{\mathrm{out}}$ from **X**(t) using **Y**(t) = **W**$^{\mathrm{out}}$**X**(t).
6. Run the system on new data to obtain an **X**$^{\mathrm{test}}$(t). Recover input and classify by **Y**$^{\mathrm{test}}$(t) = **W**$^{\mathrm{out}}$**X**$^{\mathrm{test}}$(t).

What did I do?

A look into the day-to-day…

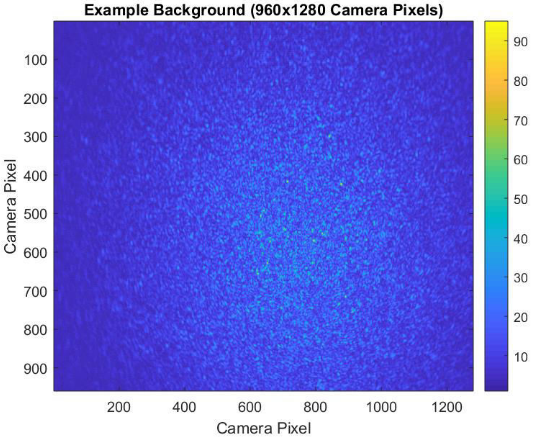
GOAL: Interface the SLM, the CCD camera, and the data acquisition into a flexible package of MATLAB .m files for easy data collection from the ESN.

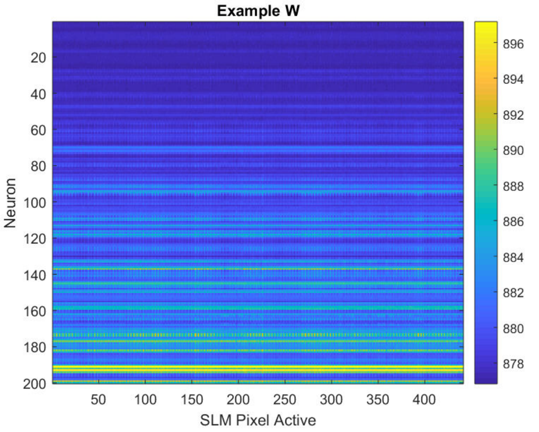
1. mainRC.m, getBackground.m
2. scaleInput.m, getW.m
3. getNeuron.m, injectU.m, neuronW.m
4. takeRowImage.m, makePixArray.m
5. SlmDefineFrame.m, SlmSelectFrame.m,

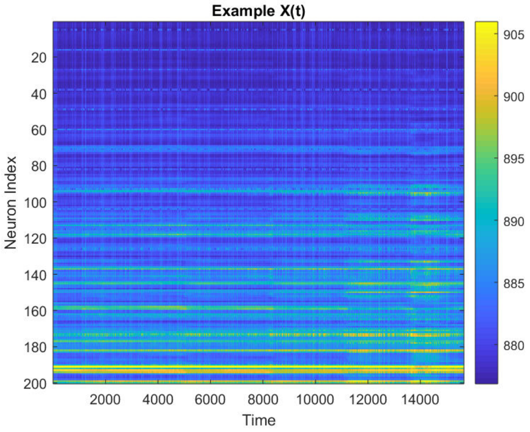


```
function [scaledInput, background, W, nn_x, nninject_x] = mainRC(inputData, nSLMPixels

    %scale input and get t_max
    scaledInput = scaleInput(inputData, offValue, onValue);
    t_max       = length(scaledInput(1,:));

    %initialize nn_x with random first row
    nn_x       = zeros(t_max+1, nNeurons);
    nn_x(1,:) = round(offValue + (onValue - offValue).*rand(1,nNeurons));
    %initialize nn_inject
    nninject_x = zeros(t_max+1, nNeurons);

    %gets background
    background = getBackground(nSLMPixels, frameNo, offValue);

    %get W matrix
    W = getW(frameNo, rowNo, nSLMPixels, startPixelW, offValue, onValue, nNeurons, neu

    %run loop to get nn_x starting with random initial row (data gets injected startin
    for k = 2:t_max+1

        %set SLM with previous nn_values
            %case k=2: no data had been injected yet so we need to use nn_x
            %case k>2: data has been injected so we can use that
        if k == 2
            nn_prev = nn_x(k-1,:);
        elseif k > 2
           nn_prev = nninject_x(k-1,:);
        end

        %make the array to be fed to SLM
        pixArray = makePixArray(nn_prev, nSLMPixels, offValue, startPixelW);
        %set SLM to k-1'th nn_x row
        SlmDefineFrame(frameNo, pixArray);
        %take picture
        updatedCam = takeRowImage(rowNo, nAverage);
        %add the k'th row of neurons to x
        nn_x(k,:)        = getNeuron(updatedCam, nn_prev, alpha, nNeurons, spacing, nII
        %inject the neurons w the data
        nninject_x(k,:) = injectU(nn_x(k,:), scaledInput(:,k-1), startNeuronInject, in
```

```
function stat = SlmDefineFrame(frameNo, driveLevels)
%
%   stat = SlmDefineFrame(frameNo, driveLevels)
%
%   frameNo is frame to be defined, in range [0, 31]
%   driveLevels is drive level, in DAC counts [0, 4095]
%    length of driveLevels array must match the pixel count
%    specified in SlmInit() used to initialize the instrument

global pixelCount;
global frameCount;

if pixelCount == 0
    disp('SLM not initialized yet');
    stat = -117;
    return;
end

if frameNo >= frameCount | frameNo < 0
    disp('Frame number must be in the range [0,',num2str(frameCount-1), ']');
    stat = -131;
    return;
end

if min(driveLevels) < 0 | max(driveLevels) > 4095
    disp('Drive levels must be in the range [0, 4095]');
    stat = -142;
    return;
end

if length(driveLevels) ~= pixelCount
    disp(['length of driveLevels array must match pixel count of ',num2str(pixelCount)])
    stat = -143;
    return;
end

stat = SlmMex(4,frameNo, driveLevels);
```







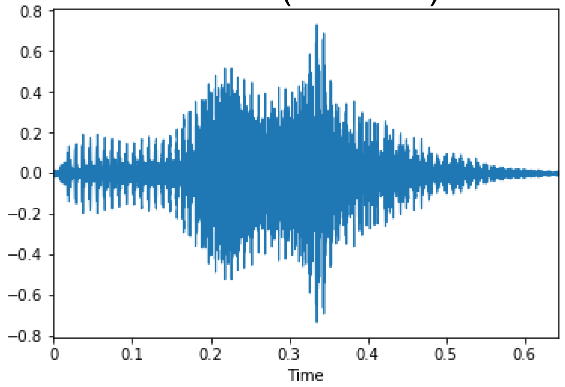
Optical ESN in Action

A Japanese language classification case study.

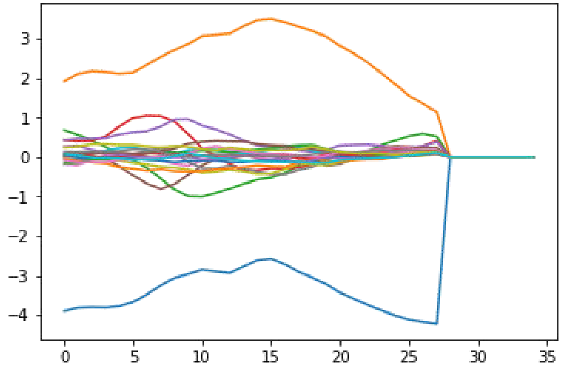
- Nine male speakers uttered two Japanese vowels /ae/ successively
- The audio files are pre-processed to generate Mel-frequency cepstrum coefficients (MFCC)
  - *Think of MFCCs being representations of the power spectrum of a sound*
- Each sequence consists of a time series of 12 MFCC coefficients

GOAL: *Given audio data, classify spoken Japanese vowels by speaker (1-9) using an optical reservoir computer.*

“One” (.wav file)



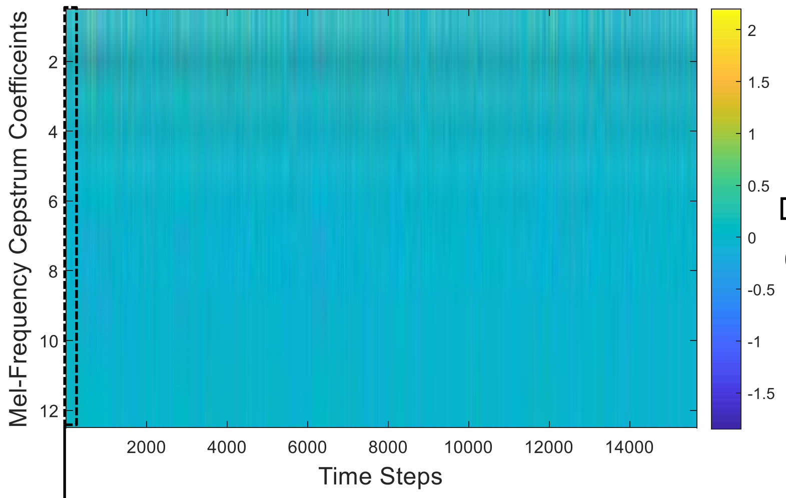
MFCCs



Plots: M. Luengo-Kovac

Experimental test of the optical reservoir computer

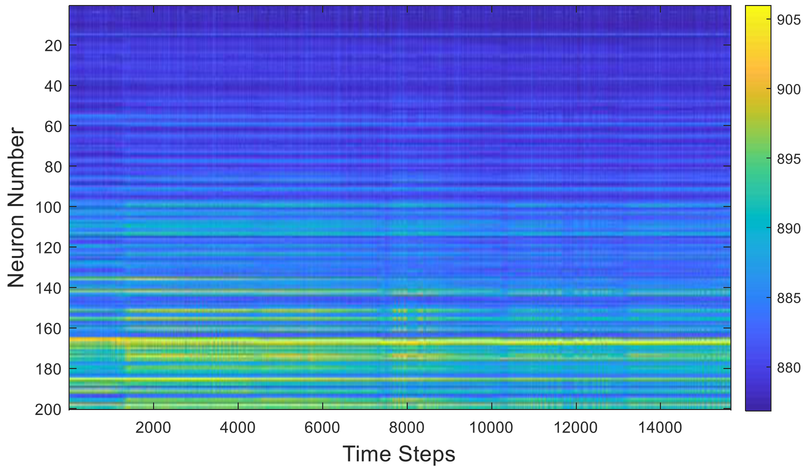
Training and Test Input

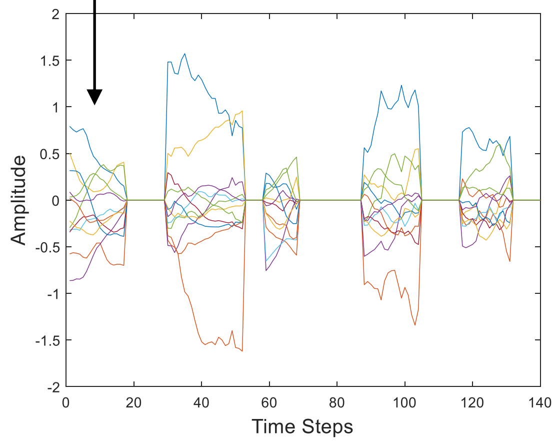


Reservoir

Computer

Neurons Output





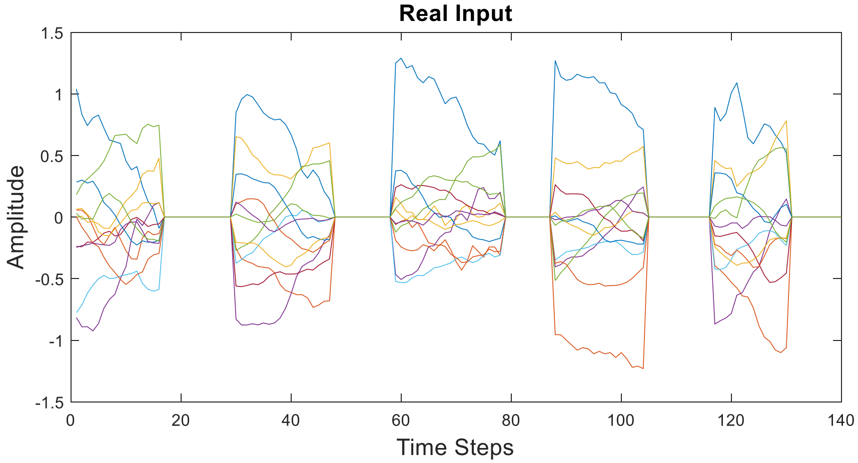
- Send input $u_i(t)$ to 12 pixels of SLM
- Take an image of the speckle pattern on the computer
  - Convert pixel values to neuron values
  - $\mathbf{x}(t+1) = \alpha f [\, \mathbf{W}\, \mathbf{x}(t) + \boldsymbol{v}\, u_i(t)] + (1 - \alpha)\, \mathbf{x}(t)$
- Inject feedback "neurons" $\mathbf{x}(t+1)$ to SLM
- Store vectors into $\mathbf{X}$ matrix of neurons for duration of training sequence
- Calculate output weight $\mathrm{W}_{\mathrm{out}}$ matrix from training data and neuron matrix
  - $\mathbf{Y}(\mathrm{t}) = \mathrm{W}_{\mathrm{out}}\mathbf{X}(\mathrm{t})$
- Continue running system to obtain test neuron matrix X
- Multiply test neuron X matrix by $\mathrm{W}_{\mathrm{out}}$ to recover waveform and classify

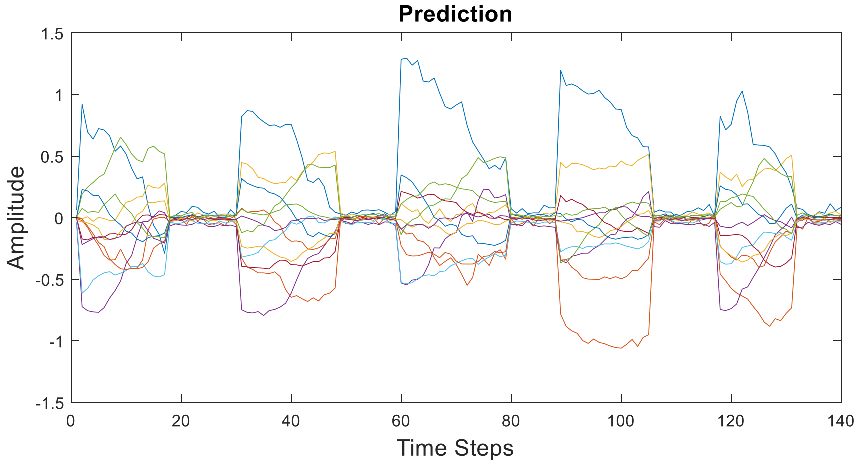
9 different speakers' MFCC coefficients ($Y_{predict} = W_{out}X_{test}$)

The features of the 12 inputs get mapped to 200 neurons. This enables important data features to be extracted automatically through the training process.

Input vs prediction of the Mel coefficients waveform

First 140 time steps of the prediction

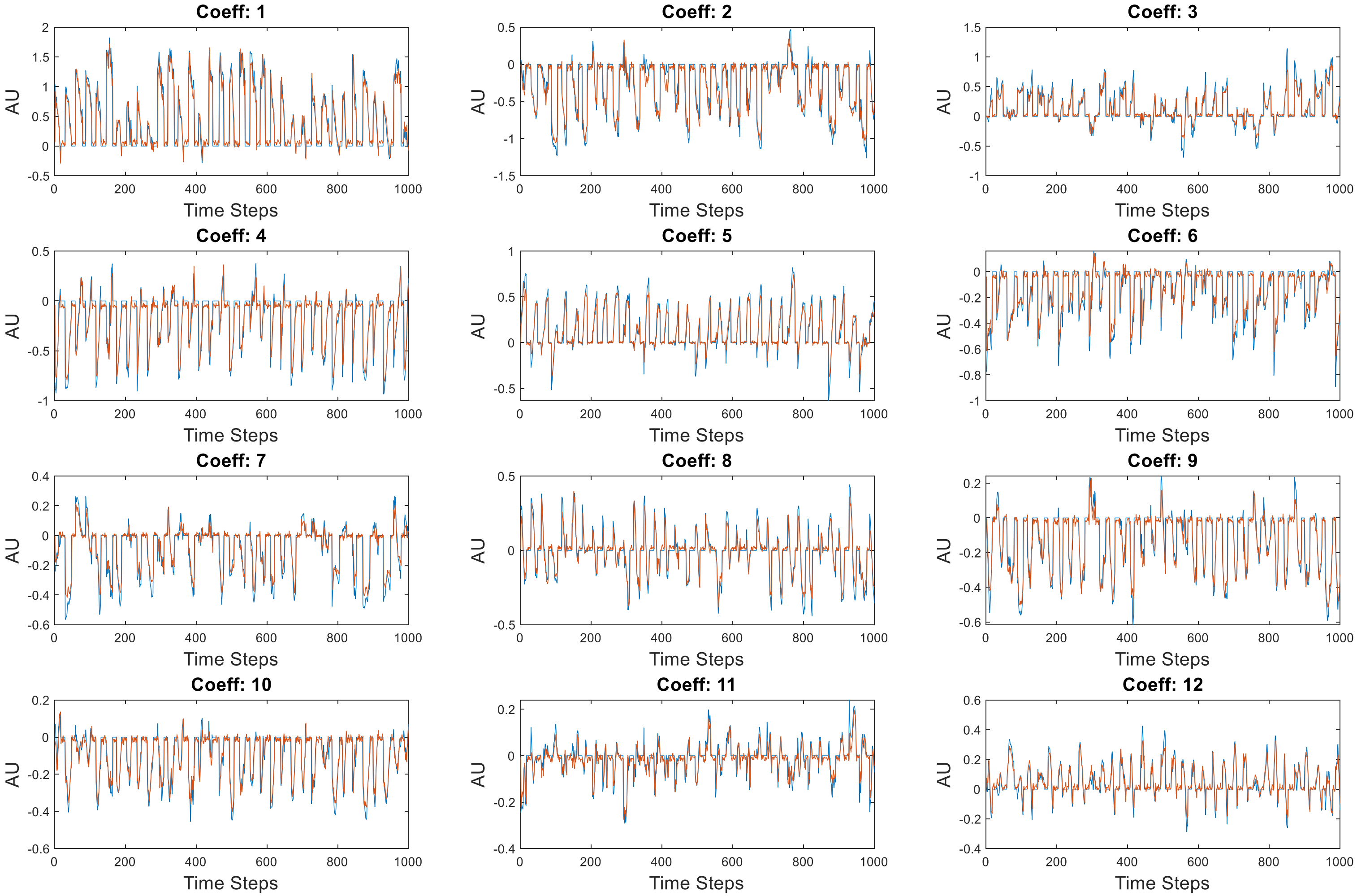




***TAKEAWAY:** All features of the Mel coefficients are well captured by the RC.*

Input vs prediction of the Mel coefficients waveform

First 1000 time steps of the prediction



Blue: Input waveform

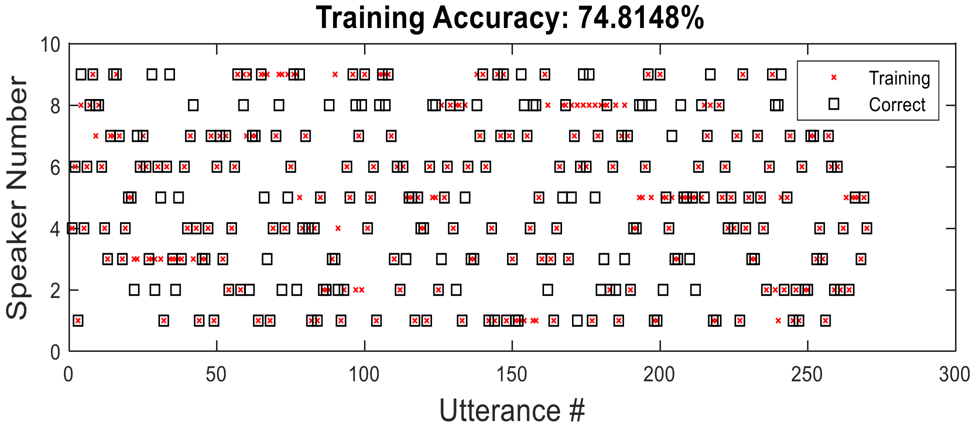
Orange: Reconstructed waveform

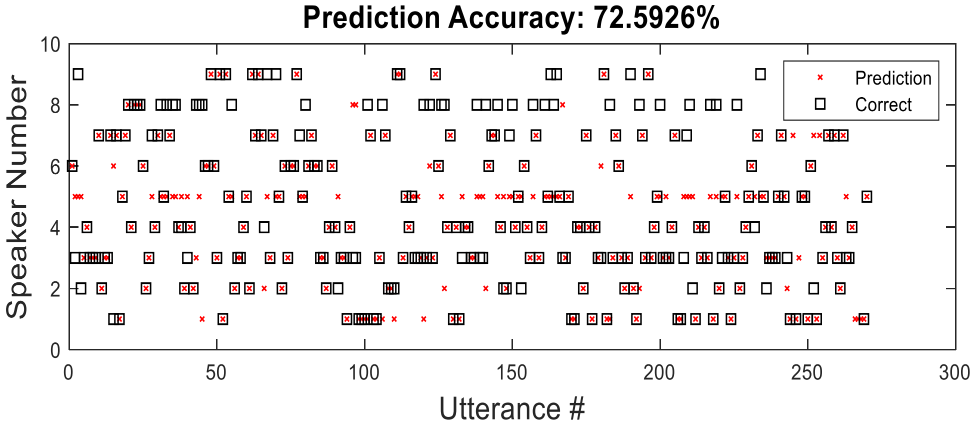
Waveform reconstruction is good but not perfect!

Classification accuracy

Can successfully predict the speaker with never seen training data

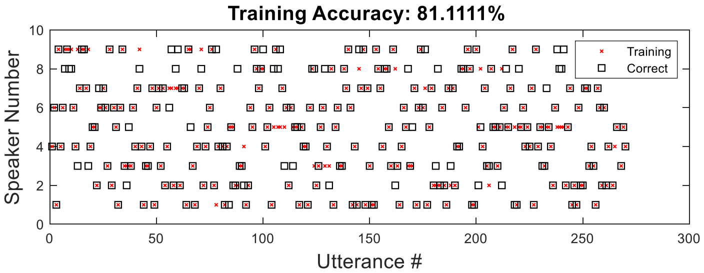
- First 270 audio inputs are used as training.
- Remaining 270 are used for test/prediction.
- Overlap of red cross and black square correspond to correct predictions.
- Obtain ~73% accuracy with regularized regression
- Random Guess Accuracy: ~11%

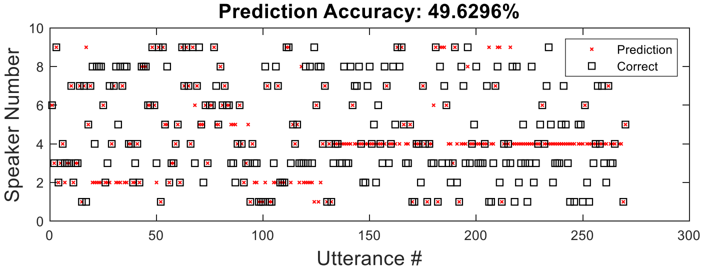


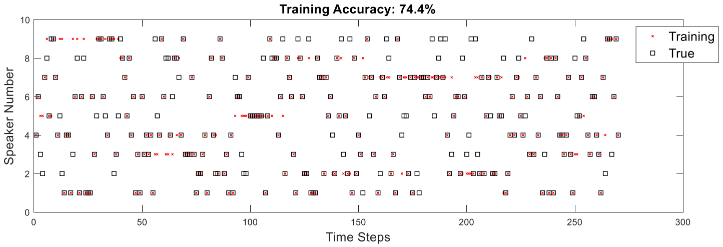


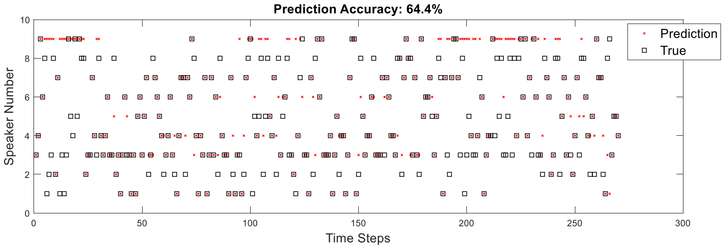
***TAKEAWAY:** Successfully classified audio files using optical RC.*

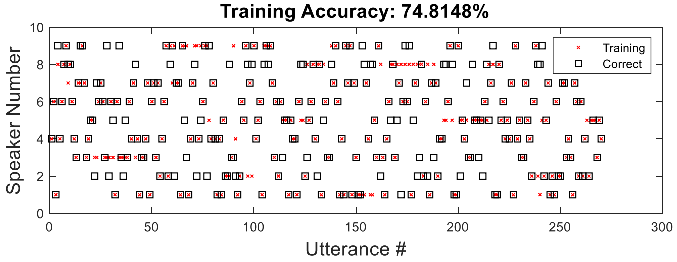
Classification Accuracy from Multiple Runs

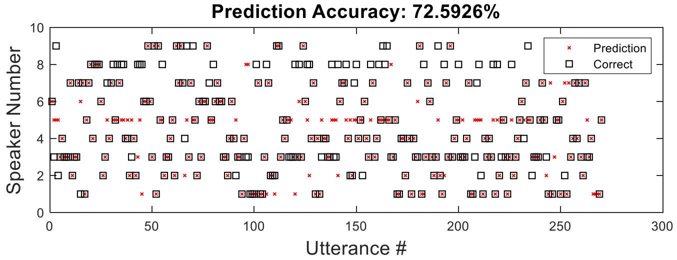


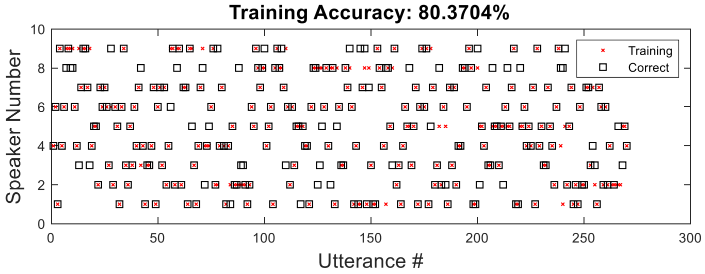


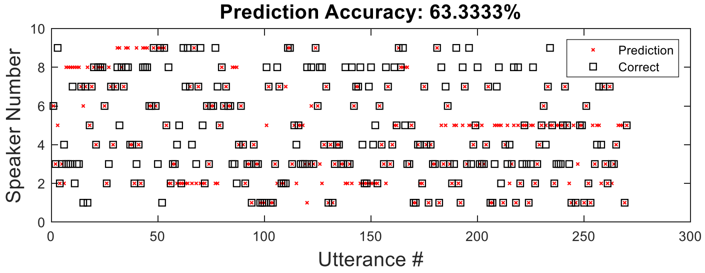












Successfully classified audio files using optical RC.

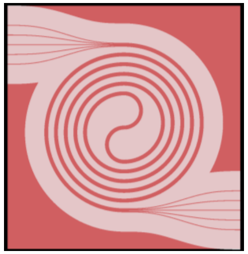
Toward the future…

What are the next steps?

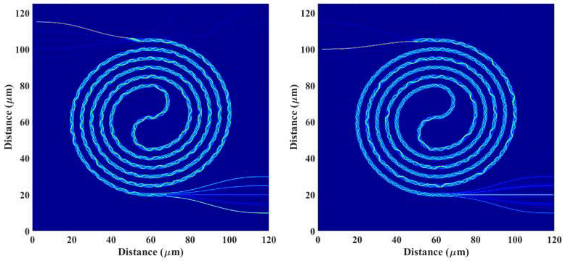
- Short Term
  - Improve the speed of data acquisition + processing
  - Stability of the optics (thermal stability, vibrational stability, etc.)
  - Optimize the kinds of problems and parameter space in which we operate echo state network
  - Understand the effect of noise and drift on the experimental setup
  - Increase number of nodes + compare simulation optimized parameters with experimental optimized parameters

- Long Term
  - Robust signal classification
  - Raw data processing (instead of intermediate MFCC-like steps)
  - Photonic integrated circuit for echo state networks

Example of planar waveguide on SOI platform



Simulated propagation for different input ports



CONCLUSION: *Much progress has been made, but as always, there is more to do.*

Thank you!

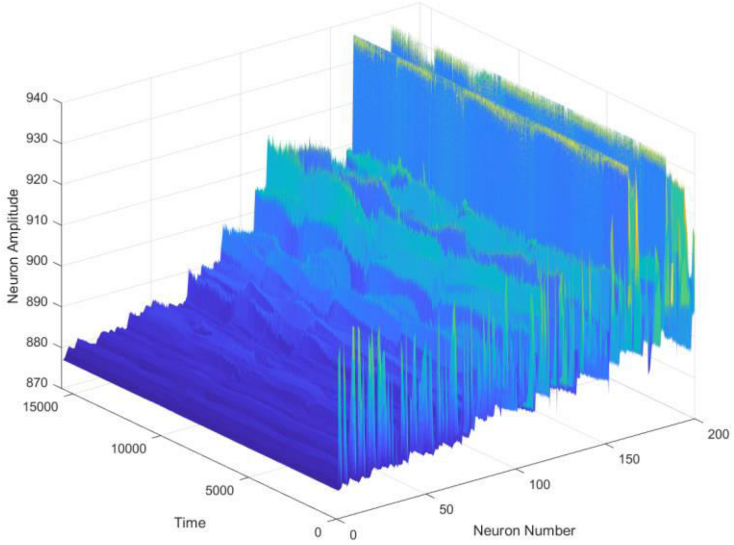
A special thanks to Uttam Paudel, Marta Luengo-Kovac, George Valley, Justin Shaw, and all the interns for the incredible experience and support.

Backup Slides/Extra Information

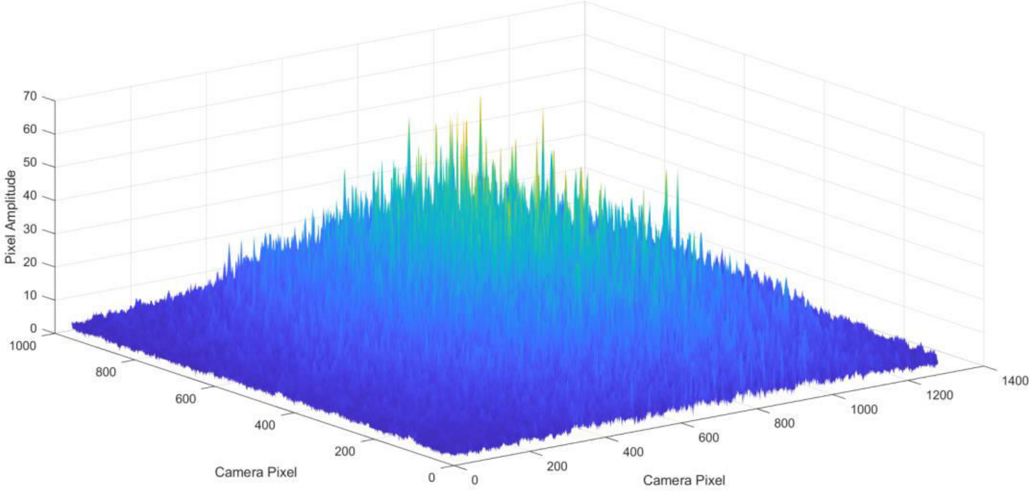
(Thanks to Uttam Paudel & Marta Luengo-Kovac)

Visualizations

Neuron Value vs. Time

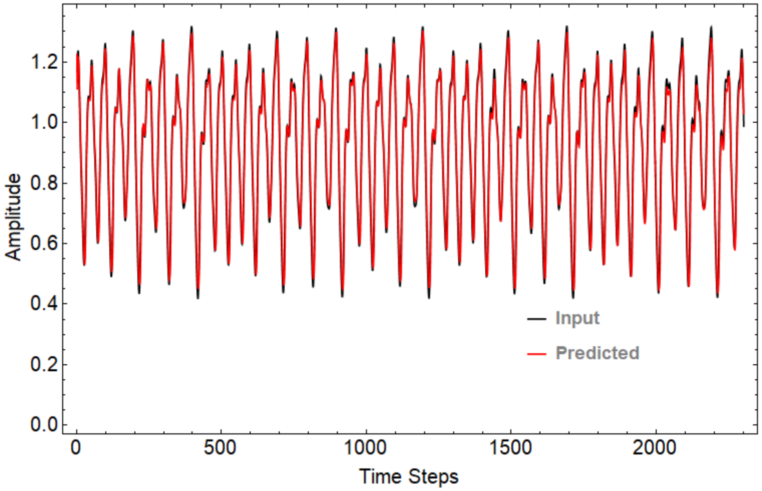


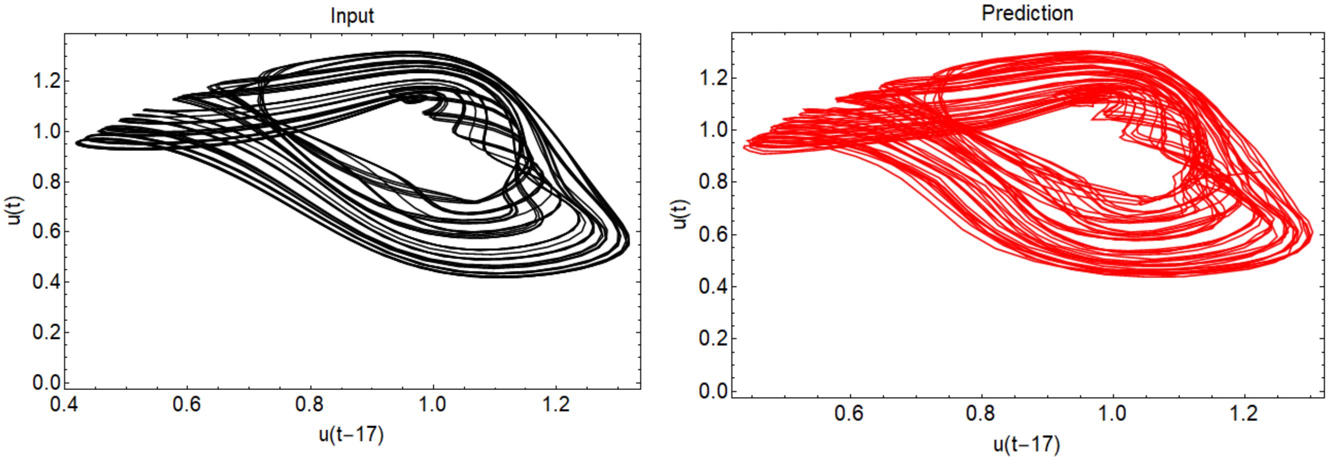
Background



Mackey-Glass non-linear time-series prediction

- Mackey-Glass function gives a complex, near-chaotic dynamics and is extremely difficult to predict.
- A Mackey-Glass time-series is mapped using the optical hardware. The system can map-out the series with high accuracy.



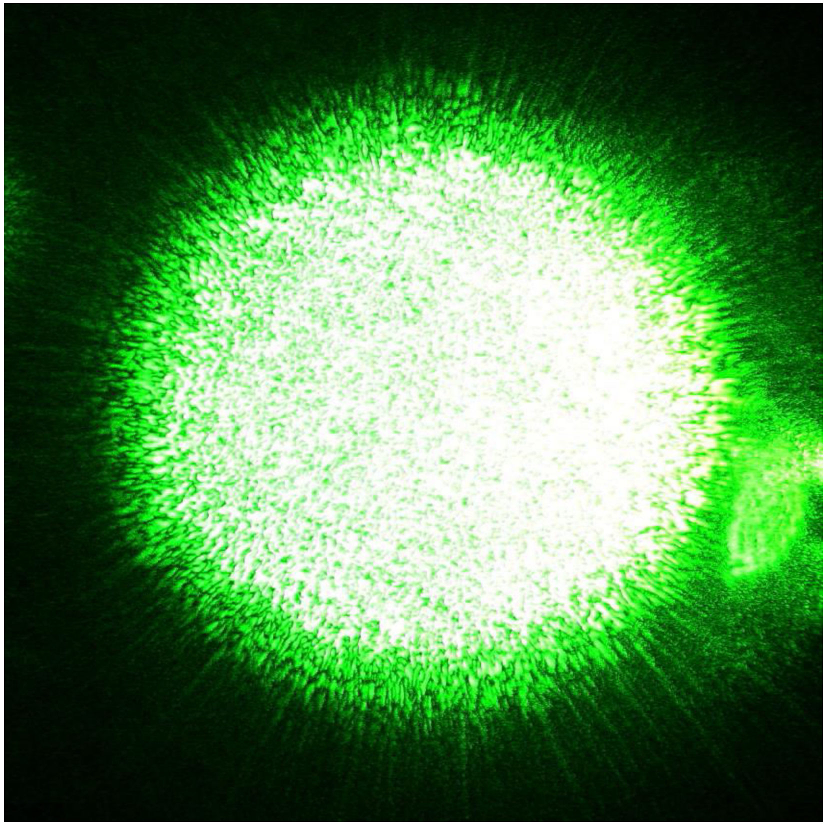


Chaotic attractor for the input and prediction generated from the time delayed MG input. The plots show that the system operates in a high-dimensional regime and our optical reservoir computer is fully able to map the dynamics.

Waveguide speckle-based optical reservoir computer can map near-chaotic dynamics

What is speckle?

- Pattern of light on a surface produced by mutual interference of wave fronts
- Typically occur in diffuse reflection of monochromatic light (like laser light)
- Results because different phases and amplitudes add to give a resultant, random pattern of light



Mathematical Formulation of Echo State Networks

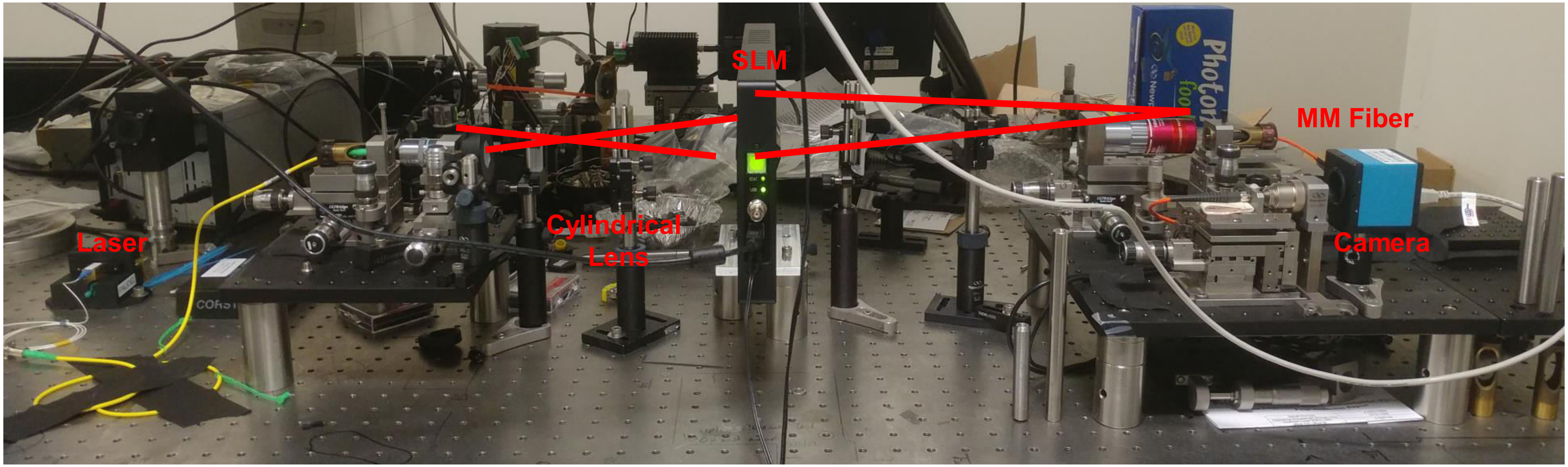
$$\mathbf{x}(t{+}1) = \alpha f\,[\,\mathbf{W}\,\mathbf{x}(t) + \boldsymbol{v}\,u_i(t)] + (1 - \alpha)\,\mathbf{x}(t)$$

$$y(t) = \mathbf{W}^{\mathrm{out}}\mathbf{x}(t)$$

- $\mathbf{x}(t)$ are the values of the neurons at time t
- α is the "leaking rate"
- f is the nonlinear sigmoidal function (e.g. tanh(x), sigmoid)
- $\mathbf{W}$ and $\boldsymbol{v}$ are the random weight matrices from speckle
- $u_i(t)$ are the input signals
- $y(t)$ is the output signal
- $\mathbf{W}_{\mathrm{out}}$ is the vector of output weights
- $\mathrm{X}_{\mathrm{training}} = \{\mathbf{x}(0), \mathbf{x}(1), \ldots, \mathbf{x}(n_{\mathrm{training}})\}$
- $\mathbf{W}_{\mathrm{out}} = (\mathrm{X}_{\mathrm{training}})^{\mathrm{PseudoInverse}}\,\mathrm{Y}_{\mathrm{training}}$
- $\mathrm{X}_{\mathrm{test}} = \{\mathbf{x}(n_{\mathrm{training}}{+}1), \ldots, \mathrm{x}(n_{\mathrm{training}} + n_{\mathrm{test}})\}$
- $\mathrm{Y}_{\mathrm{test}} = \mathbf{W}_{\mathrm{out}}\,\mathrm{X}_{\mathrm{test}}$

n_{training} = length of training data
n_{test} = length of test data

Experimental Layout



Japanese Language Classification

Slide from Marta Luengo-Kovac + Uttam Paudel

GOAL: *Classify spoken Japanese vowels into their corrects bins using an optical reservoir computer.*

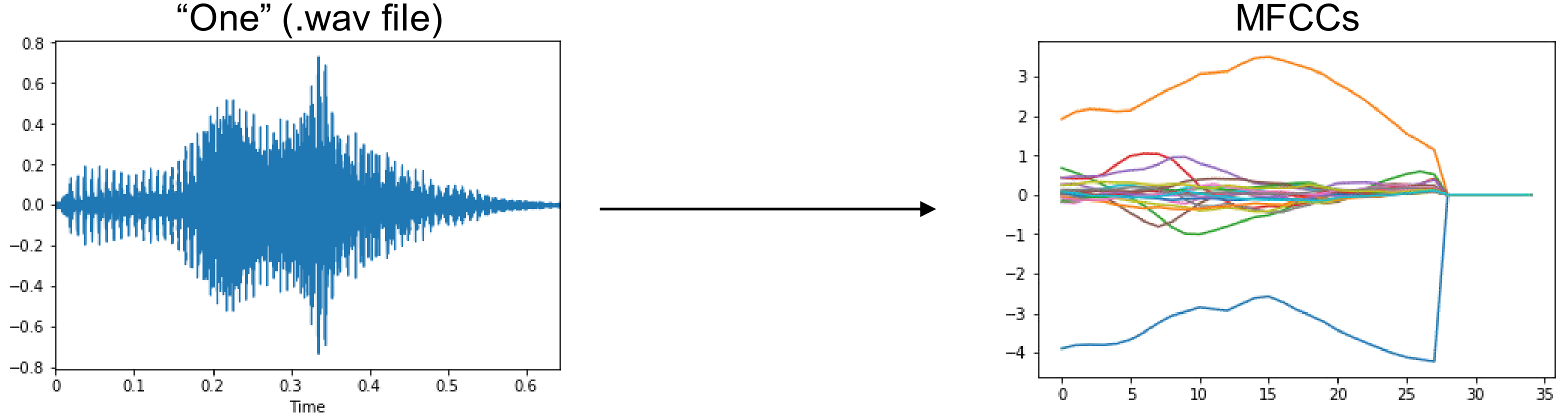
- Nine male speakers uttered two Japanese vowels /ae/ successively
- The audio files are pre-processed to generate Mel-frequency cepstrum coefficients (MFCC)
- Each sequence consists of a time series of 12 MFCC coefficients
- Training set: 270 sequences (30 utterances by 9 speakers)
- Testing set: 270 sequences (24-88 utterances by the same 9 speakers)
- Length of time series: 7-29 depending on the utterance
- <u>The task is to classify speakers using the optical RC</u>

Mel-frequency cepstrum coefficients

Widely-used method for speech classification

Slides from Marta Luengo-Kovac + Uttam Paudel

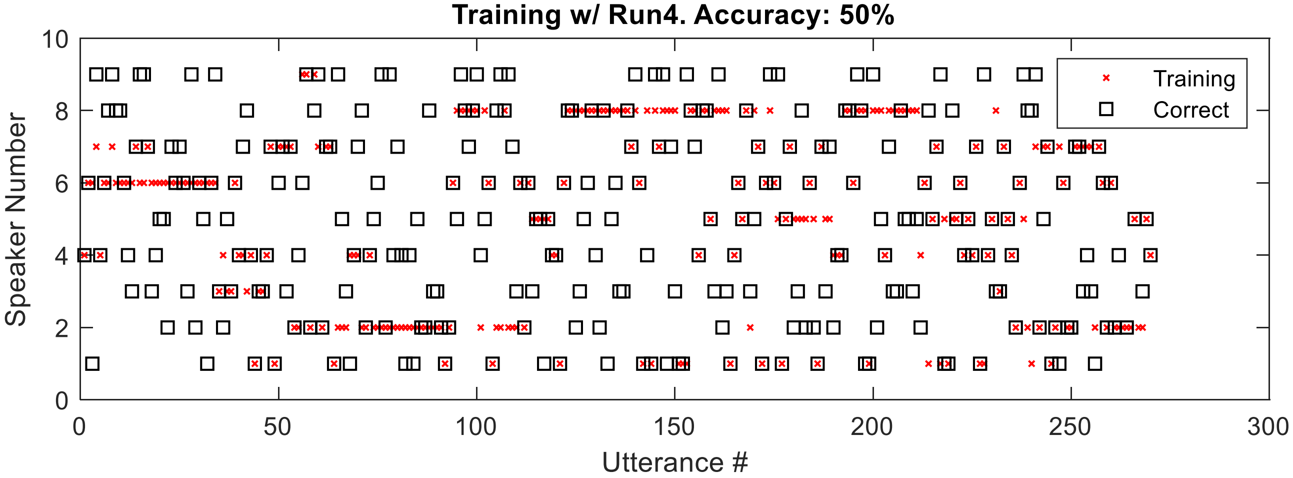
- The mel-frequency cepstrum is essentially the spectrum of the power spectrum of a sound
  - *Compute the power spectrum of a signal*
  - *Bin the power spectrum based on the mel scale (this scale more closely matches the human auditory system – e.g. less sensitive at higher frequencies)*
  - *Take the log of the power at each mel frequency (because human hearing follows a log scale)*
  - *Take the discrete cosine transform of the mel log powers to get the spectrum of the log of the power spectrum*
  - *The MFCCs are the amplitudes of the resulting spectrum*

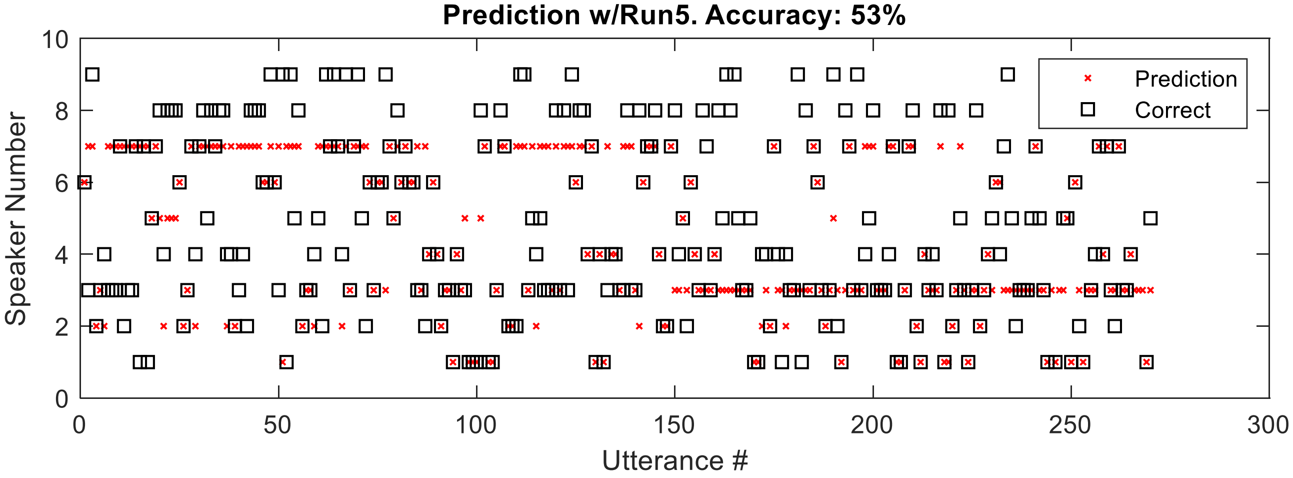

“One” (.wav file)

MFCCs

  - *Generally only coefficients 2-13 are used (the first coefficient is usually a large offset and the higher coefficients are mostly high frequency noise)*

System stability (Experiment)





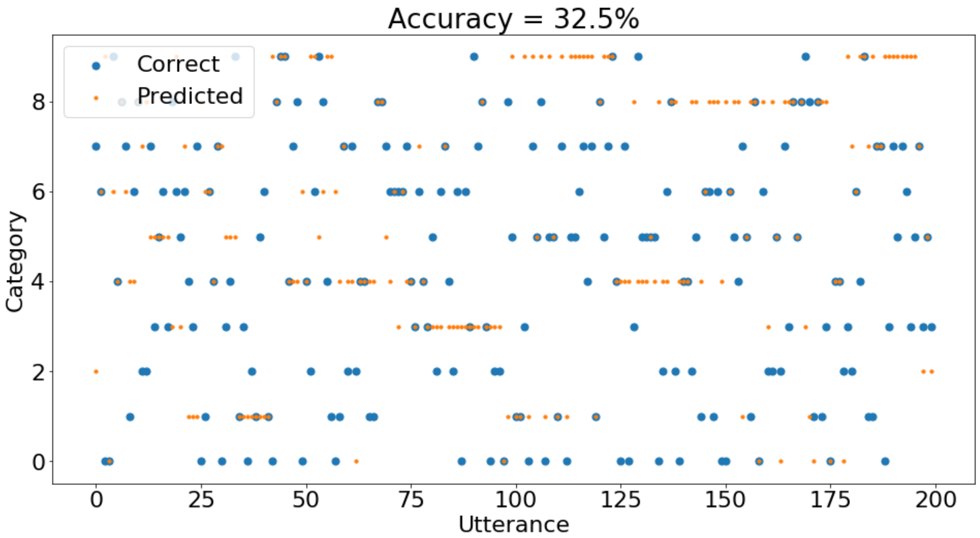
Training and test were taken over 6 hours time window.

Integrated photonics should allow >GHZ processing, training would take microsecond time.

English spoken digits classification

Four speakers with different accents

- The 10 English digits were spoken 10 times each by 4 different speakers (2 with American accents, one with a French accent, one with a German accent).
- Of those 10 times, half data were used for training and half were used for testing, resulting in 200 training utterances and 200 testing utterances.
- The raw data was the waveform of each utterance. This was converted into its mel-frequency cepstrum coefficients before being fed into the RC.
- The prediction dataset are not seen during the training period.



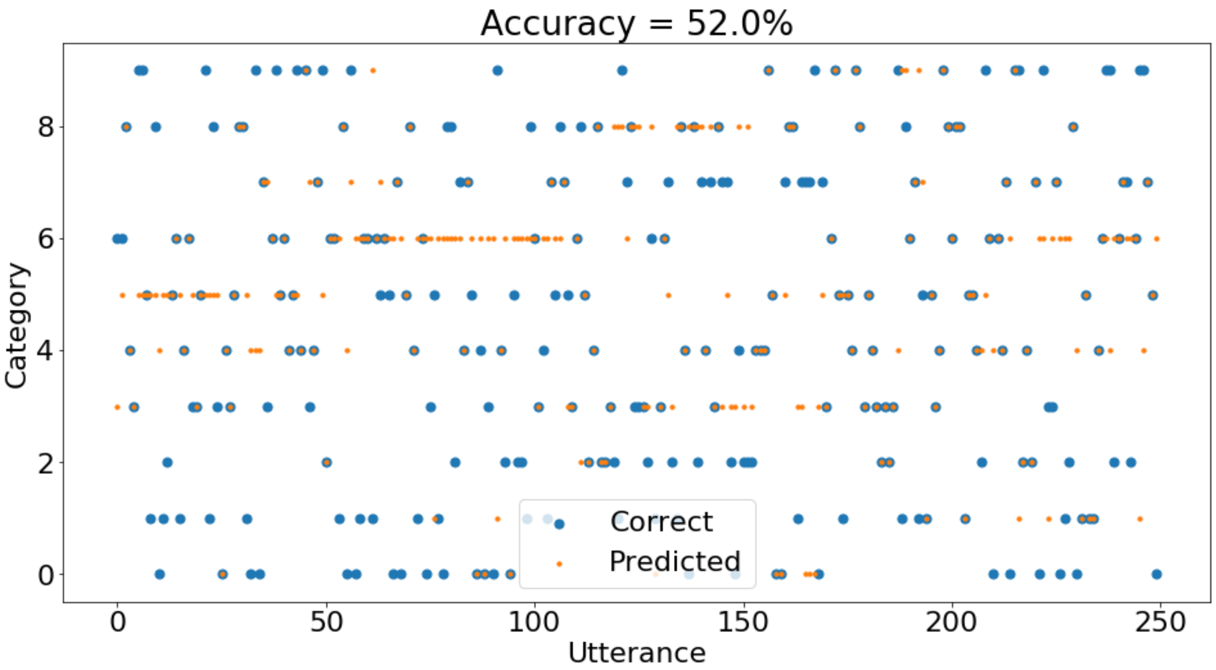
- Obtain 32.5% accuracy which is above random chance (10%).
- The lower accuracy might be due to multiple speakers with different accents.
- Longer training dataset should increase the accuracy.

Above random guess classification for English spoken digits problem.

English spoken digits classification

Single speaker

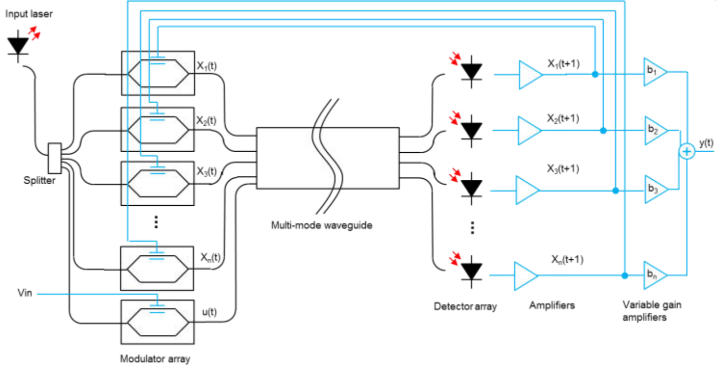
- The 10 English digits were spoken by a single speaker.
- Half data were used for training and half were used for testing, resulting in 200 training utterances and 200 testing utterances.
- The raw data was the waveform of each utterance. This was converted into its mel-frequency cepstrum coefficients before being fed into the RC.
- The prediction dataset are not seen during the training period.



52% classification accuracy for English spoken digits by a single speaker.

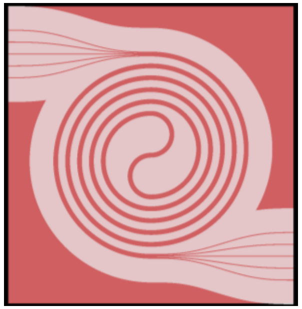
Chip-scale integrated photonics reservoir computer

- Lasers can be fabricated (InP) or chip-mounted (Si)
- Modulators array could be build using electro-absorption devices (InP) or compact ring modulators (Si).
- Planar multimode waveguide in silicon is verified to generate the speckles as predicted.
- Germanium detectors can be fabricated on silicon platform.
- All components can be readily fabricated on an integrated platform using a commercial foundry.

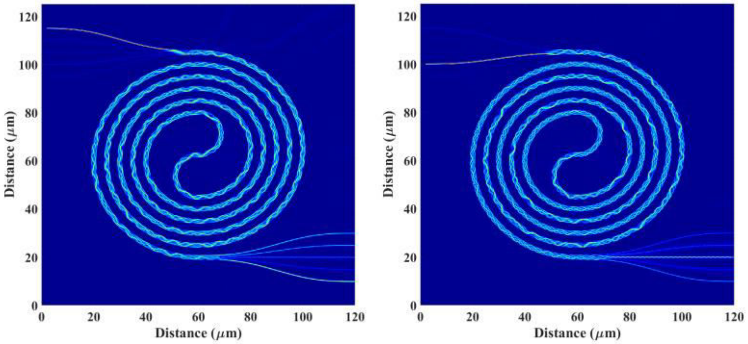


A schematic for an optical reservoir computer implemented on photonic integrated circuit.

Top-view



Example of planar waveguide on SOI platform



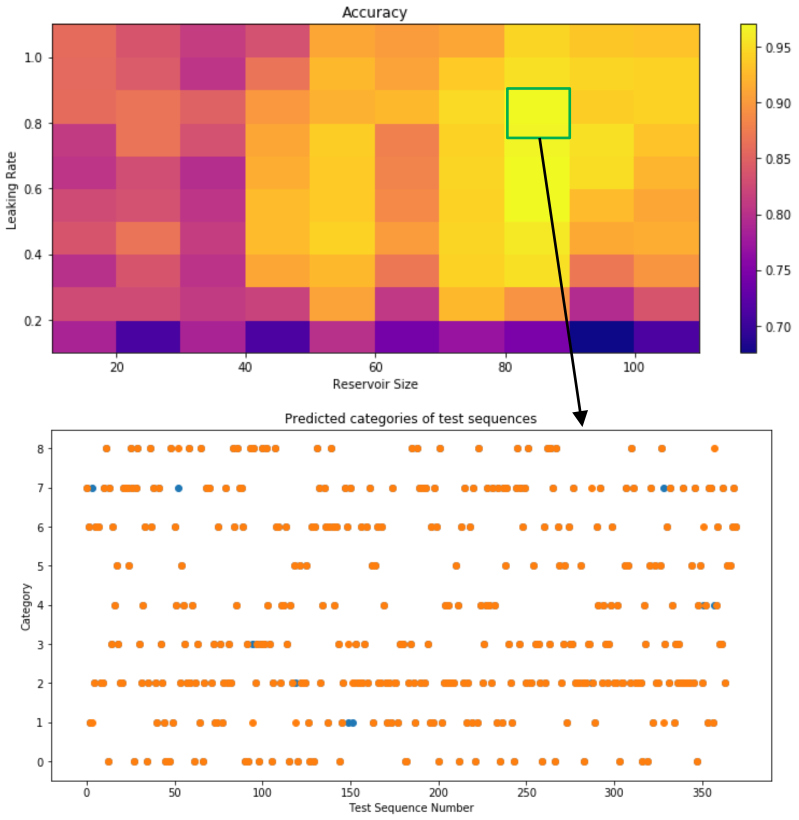
Simulated propagation for different input ports

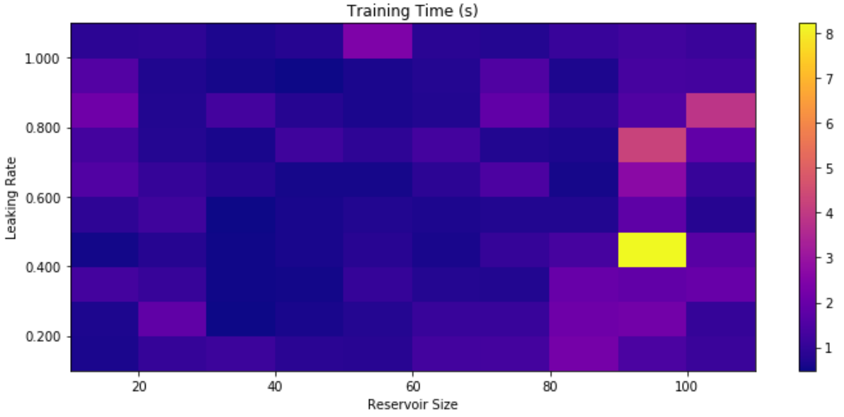
Clear path-forward for building integrated photonics circuit. Can achieve>GHZ processing with a PIC, training would take ~microsecond time.

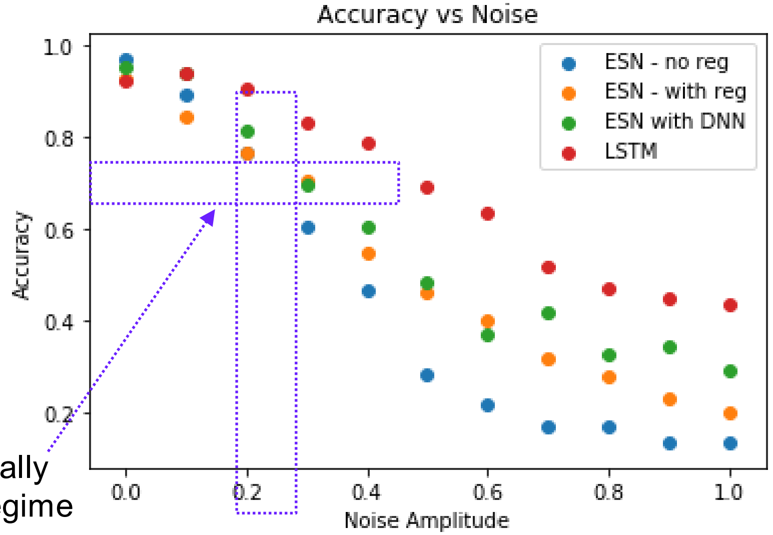
Reservoir computer on a personal laptop

Alpha, neuron size, and noise analysis

- The training sequences were sent in order (i.e. all of Speaker 1's sequences, then all of Speaker 2's sequences, etc.)
- The test sequences were sent in a random order







- Incorrect predictions indicated by blue dots.
- **Reservoir size = 80**, Leak rate = 0.8
- **Accuracy = 97.3%, Training time = 0.6 s**

Experimentally achieved regime

80 high-quality neurons are sufficient for classification, doable number for high-speed PIC