Computer Science MS Project

Poster and Presentation\(^1\)

\(^1\)Adapted from slides by Joe Geigel.
Overview

Guidelines for poster and presentation

- Important Dates and Resources
- Content and structure
Poster

What is an academic poster?

▶ A means to communicate your research to others at a conference.
▶ A point of reference for presenting and discussing.
▶ A visual abstract of your work.

Emphasis on *abstract*:

▶ a concise summary of paper’s topic, approach, and results
▶ *not* the details of the work itself
▶ a person should be able to read your poster in 5 minutes
Poster

Important dates:
- Oct. 23 (Fri): Outline (tentative)
- Nov. 13 (Fri): Draft (for peer review)
- Nov. 20 (Fri): Transcript Draft (for peer review)
- Dec. 2 (Wed): Final Poster and Audio Recording to CS Dept.
Poster

Discuss poster with your faculty advisor early and often:

- Ask for exemplar project posters supervised by your advisor
- Share outline and drafts and ask for feedback

Recall that poster is a significant component of course grade:

- 15% assigned by CS faculty feedback
Poster

What (does a good poster look like)

► Ask for exemplar posters supervised by your advisor
► Look at previous CS MS Projects, especially “Best Poster Award” recipients:
  ► Visual Tracking for Corobots; Orens Xhagolli (2181; Dr. Zack Butler)
  ► Correctly Rounded Floating-point Binary-to-Decimal and Decimal-to-Binary Conversion Routines for Standard ML; Richard Eugene Munson (2181; Dr. Matthew Fluet)
  ► Multiple camera array for dynamic real-time facial motion capture; Tanvi Milind Raut (2181; Dr. Joe Geigel)
  ► Colluded Application Attack And Its Detection Using Machine Learning; Arpit J Vora (2185; Dr. Leon Reznik)
  ► Integrating Kubernetes and RDMA; Coleman Link (2188; Dr. Minseok Kwon)
  ► A Mechanized Formalization of the WebAssembly Specification in Coq; Xuan Huang (2191; Dr. Matthew Fluet)
  ► Road Pothole Classification and Reporting with Data Quality Estimates; Sudhish Surendran Thazhakasseril (2191; Dr. Leon Reznik)
  ► Detecting Phishing using CT Logs and Website similarity; Bharath Suresh Modhipalli (2191; Dr. Taejoong Chung)
  ► Natural Language Voice Recognition System Integration; Sergey Goldobin (2198; Dr. Zack Butler)
  ► Deep Analysis of Conflict Speech Signals; Haoran Sheng (2198; Dr. Ifeoma Nwogu)
  ► Ramifications of Quantum Computing over Hybrid Cryptosystems; Viraj Vilas Chaudhari (2198; Prof. T.J. Borrelli)

Ask yourself why these posters are good?
Poster

Why (is preparing a good project poster difficult)

- Must think of presenting from the perspective of the audience
- You already know everything, your audience does not know anything
  - How did you get to know everything (about your project)?
  - What is the story that you are trying to tell?

Note: The story in the final report is different from the story in the poster. It is tempting to “compress” your final report into a poster, but they are serving different purposes.

*The purpose of a presentation is to introduce a research program and persuade the audience that the work is significant and interesting. Detail that is essential to a paper is often of little value in a presentation, so the principles of organization and presentation are quite different to those of a write-up. For any audience, there is no need for a talk to just be an overview of the paper; it is an introduction to the ideas and research results described in the paper, and many paths can lead to that same outcome of teaching the audience about your work.*

*J. Zobel, Writing for Computer Science, Ch. 16*
Scientific Method and Engineering Design

Scientific Method

1. Ask a Question
   - Do Background Research
   - Construct a Hypothesis
   - Test with an Experiment
   - Procedure Working?
     - No
       - Analyze Data and Draw Conclusions
       - Results Align with Hypothesis
       - Communicate Results
     - Yes
       - Experimental data (comes from background research for new/future project. Ask new questions, form new hypotheses, experiment again)

   - Results Align Partially or Not at All with Hypothesis
   - Communicate Results

Engineering Method

1. Define the Problem
   - Do Background Research
   - Specify Requirements
   - Brainstorm, Evaluate, and Choose Solution
   - Develop and Prototype Solution
   - Test Solution
   - Solution Meets Requirements
     - Communicate Results
   - Solution Meets Requirements Partially or Not at All
     - Communicate Results
     - Based on results and data, make design changes, prototype, test again, and review new data.
Always communicate results!
Academic Posters

Most posters follow a structure that (somewhat) aligns with scientific method and/or engineering design:

- Title
- Introduction
- Background
- Body (most variation)
  - Hypothesis / Requirements
  - Evidence / Description
  - Analyze Data / Evaluate Solution
- Conclusions
- References
- Acknowledgements
- Contact Information
Title

Prominently at top of poster with author’s name and affiliation

Most likely title is identical to that of the final report, but when presenting work at a conference without a companion paper, a catchy title can attract visitors

Logos

Some strong proponents and detractors

Good to include research group’s logo (if it has one)

CS. Dept. and RIT logos more important when presenting at a conference

Abstract?

Whole poster is an abstract of your project, so poster does not need an abstract panel.

However, a prominent, concise, one-sentence synopsis of conclusion can nicely complement the title.
Introduction / Background

Introduction

- Describe *what* the question / problem is
- Describe *why* the question / problem is important
- Describe (at a high-level) the *contribution*.

Background

- No question / problem or methodology is ever brand new
- Almost all work builds on existing work
- List few most important references
- Place your work in context.

Remember: will be saying much less than what is in final report
Body

Tell the story of your work.

Because every project is different, the body panels of each poster are a little different.

How you present your project is different from how you executed it. Remember: will be saying much less than what is in final report.

► Focus on “Key Ideas”:
  use high-level intuition to describe the “interesting” takeaway
Body / Results / Conclusions

Body
▶ e.g., Methods and Procedures
▶ e.g., System Overview
▶ e.g., Experiments
▶ Minimize text, using appropriate visuals

Results
▶ Highlights of your results
▶ Minimize text, using appropriate visuals

Conclusions
▶ How well did you solve your problem? What did you learn?
▶ List future work
References / Acknowledgements / Contact Info

References
▶ Proper citation of work is essential
▶ But, typically no more than four on a poster

Acknowledgements
▶ Thank the supporters of the work (including funding agencies)

Contact Information
▶ Let people know how to contact you later.
▶ Presenter e-mail, project web page, etc.
▶ More important when presenting at a conference.
Format and Layout

Many different styles and formats.
(Unlike for conference papers, there is rarely a prescribed poster template.)

Typically prepared with presentation software (e.g., PowerPoint, Keynote),
desktop publishing software (e.g., InDesign), or LaTeX.

▶ https://www.posterpresentations.com/free-poster-templates.html
▶ https://www.genigraphics.com/templates
▶ https://www.overleaf.com/gallery/tagged/poster
▶ ...Google “academic poster templates”

CS MS Project Posters printed by the CS Dept. are 24 inches × 36 inches;
choose a template with the proper aspect ratio.
(Although posters will not be printed for presentation,
best posters will be printed and displayed in the department hallways.)
Format and Layout

- Ensure that each section/panel is visually distinct
  - Don’t let one panel “bleed” into next
  - Use proper borders
- Ensure that the poster has natural flow
  - Does the reader know where to look next

Tempting to copy text from final report, but . . .

- Avoid too much text
  - Be brief and concise
  - Use bullets or sentences, rather than paragraph text blocks;
    if a paragraph is required, aim for no more than 7 lines
- Avoid all text
  - Use images, diagrams, illustrations, figures, tables, graphs, . . .
  - A picture is worth a thousand words
- Avoid small text
  - Titles and headings should be readable from 10ft away
  - Remaining text should be readable from 2ft away
  - Encourages keeping text concise

Aim for 1000 words or less (including figures and tables)
Format and Layout

https://www.youtube.com/watch?v=AwMFhyH7_5g

https://colinpurrington.com/tips/poster-design/
Visual Tracking For Corobots
Orens Xhagolli
Advisor: Dr. Zachary Butler
Department of Computer Science, Rochester Institute of Technology

Introduction
Tracking is a generally well researched area and there exist many benchmarks for different challenges which focus on particular features or applications of trackers in a specific setting. We are interested in finding algorithms that perform well in a general setting, and algorithms that are fast, that we can realistically use in an embedded systems setting.

We explore some issues with existing benchmarks and algorithms that perform well in specific benchmarks but do not generalize well. We develop a new benchmark and a dataset that corresponds to the 7 levels of difficulty that we propose are common in any tracking application. We focus on providing a small but specific dataset that tests the limits of each algorithm by providing these edge cases specifically.

Finally, we provide an implementation interface for working with ROS and specifically our corobots. We tested our robots with some of the analyzed algorithms and built robots that can follow humans in a crowded environment.

Methodology
- Created a dataset to identify the following 7 edge cases in Visual Tracking:
  - Level 1: Still life images
  - Level 2: Moving object frames
  - Level 3: Fast moving objects frames
  - Level 4: Scale invariant frames
  - Level 5: Image escape frames
  - Level 6: Crowded environment frames
  - Level 7: Crowded environment with 3D shift frames
- Table 1 shows the results of different OpenCV algorithms in the Hierarchy of Tracking:
- Table 2 shows the results of different OpenCV algorithms based on speed.
- ROS implementation available on github (oxhagolli/VisualTrackingCorobots)
- We provide qualitative analysis of the algorithms which can be found on the paper.

Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
<th>Level 5</th>
<th>Level 6</th>
<th>Level 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIL</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>KCF</td>
<td>Pass</td>
<td>Pass</td>
<td>Fail</td>
<td>Pass*</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>TLD</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass*</td>
<td>Pass*</td>
<td>Fail</td>
</tr>
<tr>
<td>MedianFlow</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass*</td>
<td>Pass</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>MOSSE</td>
<td>Pass</td>
<td>Pass</td>
<td>Pass*</td>
<td>Pass*</td>
<td>Pass*</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

* The algorithm did not immediately pass the test, or the solution was incomplete for the given scenario

Table 2: Speed Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIL</td>
<td>10.9</td>
</tr>
<tr>
<td>KCF</td>
<td>88.8</td>
</tr>
<tr>
<td>TLD</td>
<td>10.2</td>
</tr>
<tr>
<td>MedianFlow</td>
<td>63.7</td>
</tr>
<tr>
<td>MOSSE</td>
<td>173.9</td>
</tr>
<tr>
<td>CSRT</td>
<td>19.3</td>
</tr>
<tr>
<td>GOTURN</td>
<td>11.7</td>
</tr>
</tbody>
</table>

Conclusion
- Introduced a general framework of evaluating tracking algorithms.
- Created a benchmark dataset that highlights the issues identified by the framework.
- Ran benchmarking tests on existing OpenCV tracking algorithms.
- Provided an OpenCV implementation that works well with ROS and is parametrized so it can be used for further testing.

Acknowledgements & References
Thanks to Dr. Satya Mallick from learnopencv.com for implementation tutorials and code examples.

Thanks to Omar Kakade, Christian Brady, Aniruddha Shukla, Akash Sharma and Gokul Chandraku.

INTRODUCTION

- Smartphones, a computer like small multipurpose device have become man’s best companion.
- Statistical research shows that over 1.56 Billion smartphones were sold worldwide in the first quarter of 2018 and the number increases tremendously throughout the year of which more than 86% of the smartphones sold worldwide were phones having Android operating system.
- Applications developed with malicious intent that actively collaborate with each other to escalate their privileges which they do not possess in the first place and tries to leak the sensitive data leads to a Colluded Application Attack.
- The real-time data is a time-series telemetry data comprising of system statistics which is processed by an “AI pipeline” and then fed it to a dense deep learning architecture to detect the Colluded Application Attacks.

DATA ANALYSIS

- Based on the results of Exploratory Data Analysis (EDA), it was found that some data chunks had manual error introduced, there was lot of noise and it is highly imbalanced data.
- Principal Component Analysis (PCA) was performed to prove the hypothesis that “User-CPU usage” and “Memory Usage” are not only attributes that contributes towards an attack.
- Implementing a simple but innovative way to merge all small chunks of the time-series telemetry data without tampering the sequential pattern and preserving the history.

Deep Learning Architecture

Different deep learning architecture were trained by balancing the “Class Weight” to overcome the imbalance problem and tuned for a large set of hyper-parameters to detect an “Attack”.
- Feed Forward Neural Network: A light-weight neural network with no feedback loop is and tuned. This architecture was not able to detect the attack.
- Recurrent Neural Network: A simple recurrent neural network with a feedback gate was trained and tuned. It did not detect any attack but had less “False Alarms”.
- Long Short-Term Memory: a special type of Recurrent Neural Network having memory cells. It was able to detect an “Attack” but not always.

RESULTS

<table>
<thead>
<tr>
<th>Confidence</th>
<th>TPR</th>
<th>FPR</th>
<th>FPR w/ Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>93.33</td>
<td>11.69</td>
<td>10.49</td>
</tr>
<tr>
<td>95%</td>
<td>93.33</td>
<td>16.37</td>
<td>14.88</td>
</tr>
<tr>
<td>90%</td>
<td>93.33</td>
<td>20.62</td>
<td>17.98</td>
</tr>
<tr>
<td>85%</td>
<td>100</td>
<td>22.38</td>
<td>18.99</td>
</tr>
<tr>
<td>80%</td>
<td>100</td>
<td>24.18</td>
<td>19.85</td>
</tr>
</tbody>
</table>

Table 1. BI-LSTM architecture performance

CONCLUSION & FUTURE WORK

- Successfully able to detect “Attack” with high true positive rate and low false positive rate.
- Requires large data with more attributes for better correlation and to build a generic deep learning architecture.
- Experimenting with “Stateful” deep learning architecture and comparing the results with “Stateless” architecture.

REFERENCES

Integrating Kubernetes and RDMA

Background and Objectives
Kubernetes is a framework for managing containers across a cluster of computers. Containers managed by Kubernetes typically communicate with one another using traditional TCP/IP based networking. However, many containerized applications can benefit from using higher-bandwidth remote direct memory access (RDMA) interfaces for communication.

This project aims to add support for managing RDMA resources within a cluster to Kubernetes, such that the following objectives are satisfied:
- RDMA configuration should be automated. All necessary configuration can be performed from the master node in the cluster.
- Containers running on the same node can share that node’s hardware RDMA interfaces.
- The amount of bandwidth used by a specific pod (group of one or more containers) can be limited.
- Bandwidth on an RDMA hardware interface can be reserved for a pod.
- The existing level of isolation between containers is not reduced.

Solution
In order to extend the functionality of Kubernetes to support containers that use RDMA, several components were developed:
- **Scheduler Extension**: This component ensures that pods deployed within the Kubernetes cluster are deployed to a node that has enough available RDMA interfaces and free RDMA bandwidth to satisfy the pods requirements.
- **RDMA Hardware DaemonSet**: This component runs on each of the nodes in the cluster, configuring RDMA hardware and reporting resource usage and availability information to the other components.
- **CNI Plugin**: The container network interface plugin. This component runs on each RDMA-enabled node in the cluster when a pod that requires RDMA resources is being deployed. It allocates the necessary interfaces to that pod and configures bandwidth limitations.

Each component was developed to fit within Kubernetes’ extension framework, meaning that the solution can be deployed on a cluster without replacing any core Kubernetes components.

Deploying an RDMA-Enabled Pod in Kubernetes

Legend:
- Node within Kubernetes cluster
- Existing Kubernetes component
- Component developed as part of project
- Pod containing end-user RDMA application(s)

Steps:
1. Kubernetes API server receives request to deploy an RDMA-enabled pod on the cluster.
2. Kubernetes core scheduler sends a list of all potential worker nodes the new pod could be scheduled on to the scheduler extension.
3. The scheduler extension gathers information about available RDMA resources the daemonset running on worker node. It uses this information to filter the list received from Kubernetes, then returns the list.
4. The Kubernetes core scheduler chooses one of the nodes from the filtered list to deploy the pod on, then contacts the kubelet process running on that node.
5. The kubelet process creates the containers associated with the new pod, then runs the CNI plugin on that pod. The CNI plugin attaches the necessary RDMA interfaces to the pod.
A Mechanized Formalization of the WebAssembly Specification in Coq

Xuan Huang (xh5403@rit.edu)        Advisor: Dr. Matthew Fluet

Introduction

We developed a mechanized formalization of WebAssembly 1.0 specification in the Coq proof assistant, and (almost) completed a mechanized proof that the language is sound, implying both type safety and memory safety.

WebAssembly has been designed with formal semantics. In contrast to prior work, our mechanized formalization closely follows the W3C specification PDF. In addition to its scale and non-trivial soundness property, WebAssembly has unique features that complicate the language. Here is an example of a recursive factorial program written in C compiled to WebAssembly.

Methodology

WebAssembly is formalized as inductive tree structures (abstract syntax) with judgements and inference rules (operational semantics, static typing rules). We faithfully mechanized those mathematical definitions into Coq code with eyeball-closeness.

We then developed machine-checked proofs of preservation and progress lemmas, by induction on syntax or derivations, and the soundness theorem, as a corollary. The theorem statement and our mechanization are given on the right.

Background

Programming language theory: the formal study of PL, characterized by mathematically-precise definitions of PLs and programs.

Verification: proving the correctness of abstract systems or programs with respect to a specification or property, using formal methods.

WebAssembly (abrv. Wasm): a new low-level bytecode format and stack-based virtual machine designed as a compilation target for the Web.

Coq: a dependently-typed programming language enabling mechanized interactive theorem proving.

Challenges

1. The recursive definition of evaluation contexts introduces non-determinism and requires manual decomposition to discover structure.
2. Branching in structured control flow requires labels indexed by depth. Block types are preserved by stack unwinding and stack polymorphism.
3. Runtime structures (e.g., frames, instances, stores) not present in the source have complex invariants.

References

1. W3C, WebAssembly Core Specifications. https://www.w3.org/TR/wasm-core-1
System Architecture

Monitoring Certificate Transparency logs to detect phishing
Bharath Suresh Modhipalli | bm5937@rit.edu
Taejoong Chung | tjc@cs.rit.edu

Introduction

- Let's Encrypt started in 2014 and started providing SSL certificates for free and also automated renewal of certificates.
- Phishing websites naturally started getting SSL certificates to convince the users that they are a legitimate website.
- In 2015, Google Chrome started requiring Certificate Transparency for each new SSL certificate issued.
- Can we detect phishing by monitoring CT logs and checking for similarity with popular websites?

Background

- **Phishing** is an attempt to steal security credentials from users by disguising a fake website to look like the original website.
- **Certificate Transparency** (CT) logs are public logs that contain an entry for every new SSL certificate issued.
- Roberts et al. \(^{[1]}\), built a method that can identify domain names that could potentially be used for phishing.

System Design

- **Phishing Detector** - Compare a website fetched from CT log against popular websites from Alexa top 100k domains.
  - For each website the source HTML, screenshot of the website are collected.
    - **Histogram**: Generate histogram of each screenshot and calculate the distance between the histograms to identify similarity.
    - **Difflib**: Generate a list of HTML tags from the source HTML and compare them using difflib which outputs a similarity score.

System Architecture

CT Logs → Fetcher → Phishing Detector → Backend → User

Types of web pages

- Parking page: been.online
- Error page: cdab.gov.my
- Website: apple.com-login.ru

Results

- **Histogram**: method works well for exact matches between images.
- **Difflib**: works well for both exact matches and websites which are not exact matches.
- **Chrome extension:**

Conclusion

- Successfully built a pipeline that can detect phishing websites by monitoring CT logs and comparing against popular websites.
  - **Future work**:
    - Implement functionality to remove error and parking pages.
    - Implement more efficient phishing detection algorithms.
    - Change chrome extension to receive a list every X hours.
The objective of this project was to develop a flexible and robust framework for integrating voice controls into arbitrary applications. The framework utilizes cloud services and Natural Language Processing techniques to provide programmers with tools to define arbitrary voice commands for their applications and easily connect the commands to executable code. The framework provides a simple hook for the programmer to transfer control to the framework while abstracting away all the complexity.

The resulting system allows for easy and scalable integration of voice controls – an essential accessibility feature – into applications of any size, paving the way for improved human-computer interaction across a variety of software domains.

While state-of-the-art voice control technology is sophisticated, it is often proprietary and domain-specific, such as digital voice assistants. This project aims to make voice control accessible to projects by non-tech companies, small teams, and indie developers. Easy, cheap, and scalable voice control integration provided by this framework will allow for more software to be accessible to individuals who most benefit from it, such as those with visual impairments or learning disabilities.

The framework consists of four independent modules (Figure 1) connected into a command processing pipeline. The design is modular and open-source, allowing each component to be used independently if necessary.

The objective was to develop a flexible framework that allows developers to provide programs with voice-based controls while keeping integration as easy and seamless as possible. A programmer needs only to complete the following simple steps:

1. Import the main pipeline module into their code
2. Define a library of templates for desired commands
3. Write appropriate DMs for each template
4. Create a config file pointing at those resources
5. Install the framework like any other Python module

From that point, the full functionality of the framework can be leveraged through a simple function call at the points where receiving user input is required.

A detailed tutorial on framework integration is available at the project’s GitHub page available via the QR code to the right.

Future Work & References

Future work

• Focus on enhancing the expressive power of LF templates by allowing wildcard matching and grouping of primitive components.
• Make the framework more scalable by allowing nested folders within template/DM libraries. Improve expand compile-time framework integrity validation.
• Improve performance by allowing narrow-scope template matching.

References


Figure 1: Pipeline Diagram

Figure 2: LF for “Download pictures of puppies”

Logical Form

(SPEECHACT “REQUEST”)

content

Agent

“AUGMENT”

INFOM-person

(INFOM-SET:* Imaginary Person)

type

“*YOU”

INFOM-SET:* Imaginary Person)

Name (role)

Unique identifier used for DM matching

Name (command)

Reference to construction

Reference to construction

id

Component name

map_param_name

Word binding

group

Recursive word binding

fuzzy

Wildcard recursive match

Figure 4: Command Template Example

Command Dispatch

```python

Command Dispatch

( name: "GET_NAME",
  type: "GET",
  args: ["arg1"],
  groups: ["group1", "group2"];

class: "my_class"
method: "my_method"
)

Figure 3.1: Get Example

Figure 3.2: Invoke Example

DMs serve to connect implementation-independent templates to implementation-specific code. Get command yields a dictionary with bindings for specified parameters or groups. Invoke command yields a call to the function specified by module, name, and (optionally) class. Bound parameters are passed to the function as arguments.

Integration

The framework was designed to make integration as easy and seamless as possible. A programmer needs only to complete the following simple steps:

1. Install the framework like any other Python module
2. Define a library of templates for desired commands
3. Write appropriate DMs for each template
4. Create a config file pointing at those resources
5. Import the main pipeline module into their code

From that point, the full functionality of the framework can be leveraged through a simple function call at the points where receiving user input is required.
Deep Analysis of Conflict Speech Signals

Haoran Sheng (hs1911@g.rit.edu)  Advisor: Ifeoma Nwogu (ion@cs.rit.edu)

Introduction

- **Motivation**: Extend and improve the existing researches on using SER (Speech Emotion Recognition) model for emotion analysis.
- **Goal**: Develop a good SER that could be almost as good as the state of the art.

I. Jointly use more features to make the SER model achieve higher accuracy.
II. Use LSTM in recurrent neural network to build SER model.

Intimate Conversation Dataset (ICD)\(^1\)

- Conversations of 35 couples discussing some problems in their relationships.

Approach

1. Data processing and feature extraction:

<table>
<thead>
<tr>
<th>Features</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MfccMean</td>
<td>50</td>
</tr>
<tr>
<td>MfccStd</td>
<td>50</td>
</tr>
<tr>
<td>MfccMax</td>
<td>50</td>
</tr>
<tr>
<td>Chroma</td>
<td>12</td>
</tr>
<tr>
<td>Mel</td>
<td>128</td>
</tr>
<tr>
<td>Spectral Contrast</td>
<td>7</td>
</tr>
<tr>
<td>Others (Pitch, Magnitude, rms etc.)</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 1. Features

The audio waveforms of different emotions extracted using Librosa


3. Data Analysis: Apply SER model to ICD for analyzing.

Model

- **Features**: Processed 312-dimensional features.
- **Model**: LSTM (Long short-term memory).
- **Output**: Softmax of 7 emotions.
- **Accuracy**: 79% (Average) and 83% (Best) on RAVDESS dataset.

![LSTM Architecture](image1)

![LSTM Accuracy](image2)

Comparison of different models

In order to better demonstrate the superiority of LSTM, I also adopted some traditional machine learning methods. At the same time, LSTM was compared with some models that performed well on the RAVDESS dataset.

Model optimization process

Because we chose to use recurrent neural network modeling this time, I tested both LSTM and GRU recurrent neural networks. I also tried to combine it with a convolutional neural network, but because the dimensionality of my input is not large, the idea of CNN was abandoned.

Then, I return to optimize LSTM and GRU to achieve a higher accuracy. After constant parameter adjustment and experimentation, I found that after dividing the RAVDESS dataset into training set and test set accor-ding to the ratio of 17:3, the accuracy of the two models are the highest, and LSTM performs better. Therefore, I finally chose the LSTM for furth-er analysis.

![Model Accuracy](image3)

Future Work

- The accuracy of the SER model still needs to be improved. By increasing the accuracy of SER, the results will be more accurate.
- This project only used 35 of the 152 conversations in the Intimate Conversation Dataset. A comprehensive analysis needs to extend to all 152 conversations.
- The current SER model is based on the RAVDESS dataset. Although this dataset is good, its scale is still limited. If there is a more complete dataset, the trained model will be more reliable.
- The model we trained can only classify 7 emotions, the types of emotions that the model can recognize should be improved.

Reference

Presentation

Typically, would hold an in-person poster session during finals week; CS Faculty would stop at your poster to discuss your project.

Poster is a visual to support describing your work

▶ Know your audience (most are not experts in your area)
▶ Talk to your audience, not to the poster
▶ Be concise (like the poster itself)
▶ Use the poster as an aid; don’t read the poster verbatim
▶ Give audience an opportunity to ask questions; expand on details as the conversations allow
▶ Thank your audience for listening
Presentation

Typically, would hold an in-person poster session during finals week; CS Faculty would stop at your poster to discuss your project.

But, this is year is not typical....

Instead, submit a 5 minute audio of you presenting the poster; CS Faculty will view your poster and listen to the audio to give feedback.

Poster is a visual to support describing your work

- Know your audience (most are not experts in your area)
- Be concise (like the poster itself)
- Use the poster as an aid; don’t read the poster verbatim
- Thank your audience for listening

Note: a 5 minute presentation is roughly 750 words. Audience will be reading poster while listening to presentation; emphasize and add to what is on the poster, don’t repeat.
https://www.youtube.com/watch?v=0ozwCEeaVWE
Poster Grading

CS Faculty evaluate posters.

Criteria

▶ Content
▶ Format and Layout
▶ Presentation
Schedule

- Friday (no class)
- Friday (Oct. 23 @ 8:00pm): Poster - Outline due
- Monday (no class)
- Monday (Oct. 26 @ 8:00pm): Final Report - Additional Section Draft due
- Wed/Fri (Week 11): Peer Review - Final Report
- Wed/Fri (Week 12) and Mon/Wed (Week 13): Milestone 3 Presentations
  - Wednesday (Nov. 4 @ 8:00AM): Milestone 3 Presentation due
- Friday (Nov. 6): Milestone 3 Deliverables due to Faculty Advisor
Poster - Outline

Due Friday (Oct. 23) @ 8:00PM

- PDF format required; file should be named poster_outline.pdf
- A trivial “Problem, Example, Methodology, Evaluation, References” outline will receive 0 credit; it should be possible to identify the project from the outline.
- Be as detailed as possible, with planned content in each panel of the poster.
- Poster format is not important at this time; a bullet-list of panel titles and content suffices.
- Similarly, a hand-drawn poster outline suffices; use a document scanner or a mobile scanning app like Microsoft Lens or Adobe Scan to convert to PDF.
- Illustrations, figures, and tables are especially important in posters; if actual images are not readily available at this time, then include a textual description of the intended image.
Final Report - Additional Section

Due Monday (Oct. 26) @ 8:00PM

- PDF format required; file should be named report_section.pdf
- Minimum 1500 words (approx. 3 full columns of the CS MS Project template)
- Write as complete a draft as possible for a chosen section of the final report (other than the “Introduction”). At this stage of the semester, the “Background”, “Motivation”, “Related Work”, “Implementation”, or “Experimental Design” sections or some other project-specific section related to your solution might be good candidates.
- Low-level formatting (fonts, margins, etc.) is not a primary concern; focus on writing some text to get feedback.
Milestone 3 Presentation

Due Wednesday (Nov. 4) @ 8:00AM

- Milestone 3 Presentation
  - PDF format required; file should be named milestone3.pdf
  - 5 minutes (+ 3 minutes questions)
  - For a non-expert audience familiar with your previous Project Description and Milestone presentations.
  - Title: Project title, student, faculty advisor
  - Brief (1 slide, 30 seconds) summary / reminder of project
  - Summary of (Preliminary) Results
    - Accomplished Work: What was the proposed (ideal) final deliverable? What is the actual final deliverable?
    - Evaluation: What evidence to measure success was gathered?
    - Medium: Use an appropriate medium to show results (screenshots, pseudocode, figures, tables, etc.).
  - Conclusions: What are your conclusions with respect to your project? Has it been successful?
  - Future Work: How could your project be extended in the future?
  - Expectation is that you are presenting preliminary or final results.