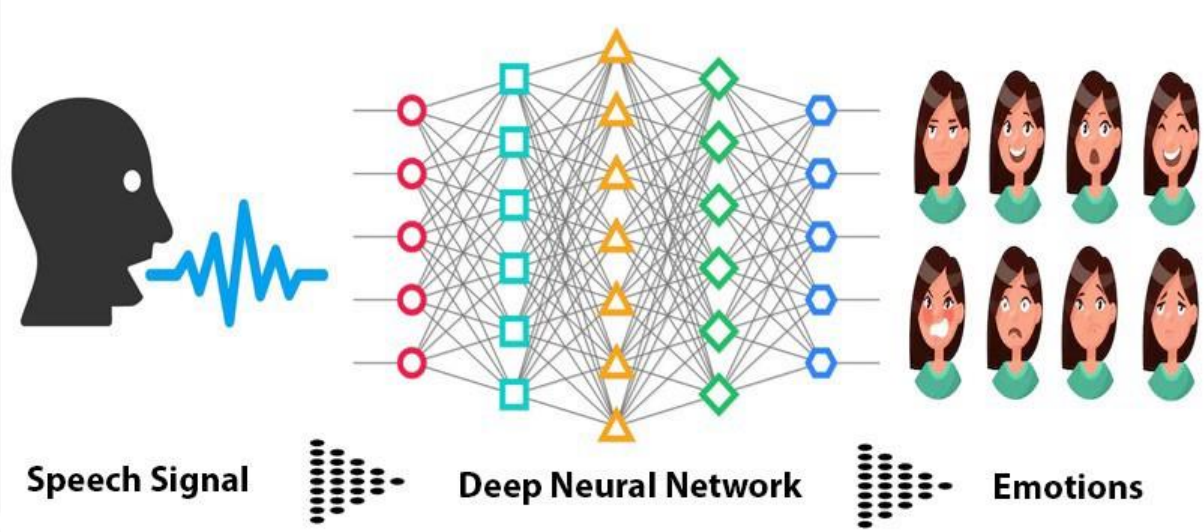
Machine Learning Methods for Speech Emotion Recognition: A Survey Study

Yenho Chen, Hyeongju Choi, Zaid Khan, Mohammad Nikbakht

Motivation

The pandemic social restrictions led to a lack of interactions and psychological distress which affected the emotional and mental health of individuals impacted by the pandemic. Thus, a need for remote emotion monitoring is felt. Speech emotion recognition (SER), an algorithmic detection of emotional state from speech signals can help address this issue.

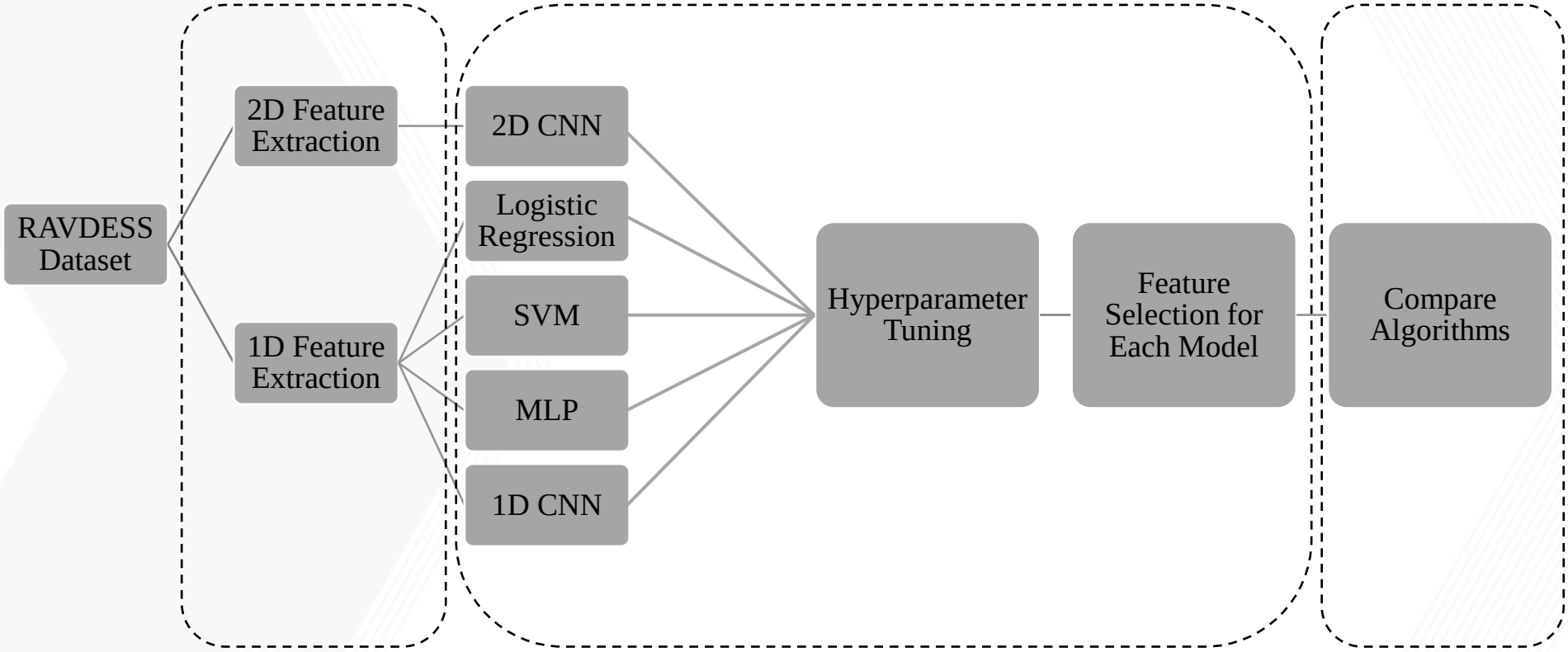


Contributions

- **Our contribution**
  - Design a pipeline that compares performance of SER across different machine learning models and feature extraction methods to gain deeper knowledge into the task
  - For each model, we discover the optimal feature subset in a principled manner via a feature selection algorithm to identify the features that carry the most emotional information
- **Broader impacts**
  - Improves human computer interaction
    - Applications in healthcare, military, and many other fields.
    - Enables remote emotion assessment from speech in parallel with facial image methods
  - With the rise of immersive virtuality reality technologies emotion recognition can become an important feature

Georgia Tech
CREATING THE NEXT

Technical Approach



Feature Extraction

Model/feature selection and Hyperparameter Tuning

Compare on Test Set

RAVDESS Dataset

- **Dataset Summary**
  - Used audio-only samples for this project
  - Consist of 1440 recording samples each ~4/5 seconds long
  - 24 professional actors (12 male, 12 female)
  - 8 emotions: calm, happy, sad, angry, fear, surprise, disgust, neutral

- **Dataset Split**
  - Split data samples into three sets: train, validation, test
  - Actors 1-21 for training and validation & 21-24 for test to prevent data snooping
  - Account for imbalanced class by oversampling to avoid biased predictions
    - Only neutral emotion required oversampling

Feature Extraction

Humans can sense emotions from only sounds. If we convert the raw audio signals into a perceptually meaningful space, we can represent the data more closely to how humans perceive and improve audio classification. Some of these features are:

- **Mel-frequency cepstral coefficients (MFCCs)**
- **Mel spectrogram**
- **Spectral contrast**
- **Chromagram**

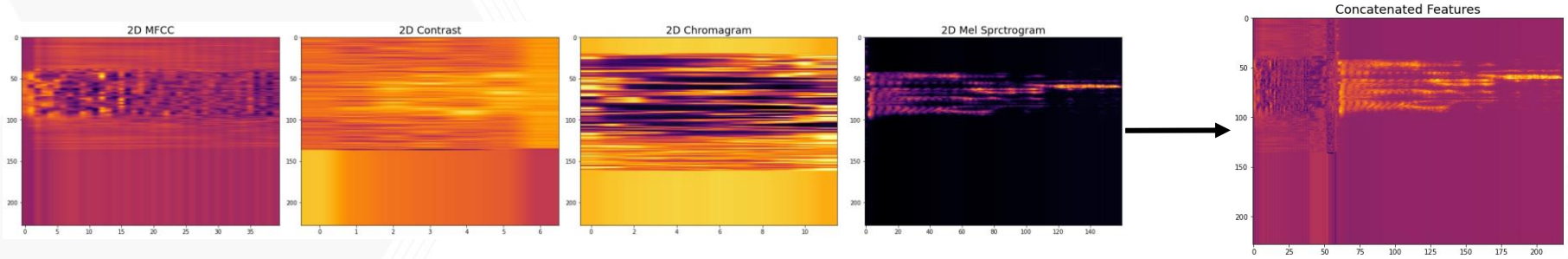
Feature Extraction

- **2D feature extraction:**

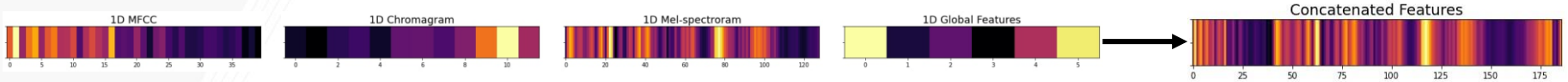
Windowing the signals by 50 ms windows 50% overlap

Compute Mel-scaled spectrogram, MFCCs, chromagram, and spectral contrast

Represent as a spectral image



- **1D features extraction:**
  - Average of 2D features over time to create 1D vectors
  - Append global features such as max, min, mean, std, range

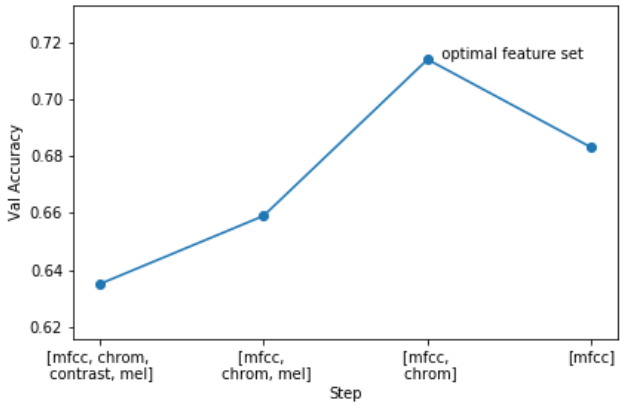


Feature Selection

- **Why?**
  - Reducing the number of features used for models allows faster learning, improved generalization, and enhanced interpretability
- **Backward Selection**
  - We utilized backward selection algorithm for feature selection
  - Greedy search to find the optimal set of features by generating a path through all possible subsets
  - We use validation accuracy as the metric

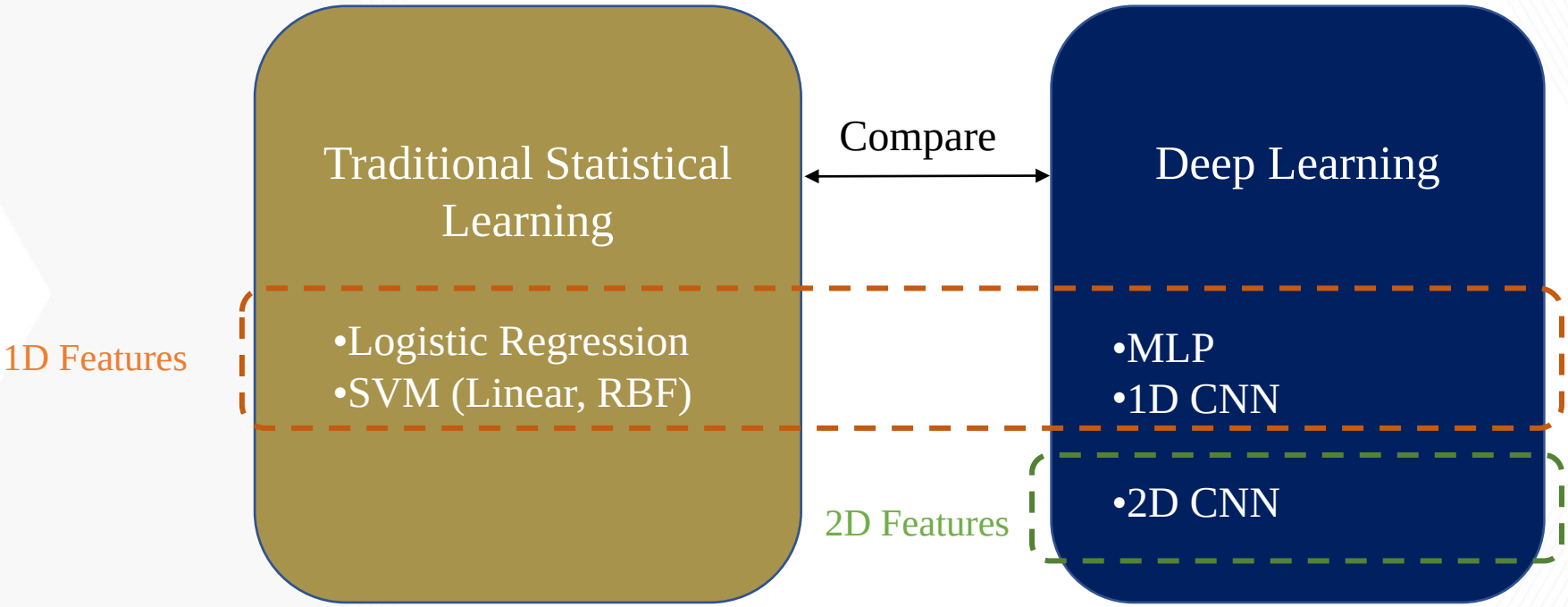
Algorithm 1 Backwards selection algorithm

Input: $X_n = [\{x_1, ..., x_n\}]_{i=1}^m$ feature set, $y = \{y_1, \ldots, y_m\}$ emotion labels, n total number of features, m number of data samples, $f(x;\theta)$ model, θ model parameters
Output: k Optimal feature subset
Initialize: $k \leftarrow \{1, \ldots, n\}$ all features, $j \leftarrow 1$ step number, $P_1 \leftarrow k$ backwards path, A step scores, a step scores for inner loop
$X_{k,\text{train}}, X_{k,\text{val}}, y_{\text{train}}, y_{\text{val}} \leftarrow \text{TrainTestSplit}(X_k, y)$
$\theta^* = \text{fit}(f(x;\theta), X_{k,\text{train}}, y_{\text{train}})$ ▷ fit full model
$A_1 \leftarrow \frac{1}{n_{\text{val}}} \sum_{i=1}^m \mathbb{I}[f(X_{k,\text{val}}; \theta^*) = y_{\text{val}}]$ ▷ evaluate
while $n > 0$ **do**
  for $i \in \{1 \ldots, n\}$ **do** ▷ eliminate one feature at a time
    $k \leftarrow \{1, \ldots, n\}/i$
    $\theta^* = \text{fit}(f(x;\theta), X_{k,\text{train}}, y_{\text{train}})$ ▷ fit reduced model
    $a_{i,n} \leftarrow \frac{1}{n_{\text{val}}} \sum_{i=1}^m \mathbb{I}[f(X_{k,\text{val}}; \theta^*) = y_{\text{val}}]$ ▷ evaluate
  end for
  $k \leftarrow \{1, \ldots, n\}/\text{argmax} a_n$ ▷ record subset with best score
  $A_j \leftarrow \max a_n$
  $P_j \leftarrow k$ ▷ record path
  $j \leftarrow j + 1$
  $n \leftarrow n - 1$
end while
$j^* \leftarrow \text{argmax} A$
$k \leftarrow P_{j^*}$ ▷ choose optimal subset



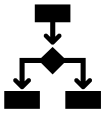
Outline of Models

Does more model complexity improve SER performance?



Model Selection

- **Steps taken for each model:**

1. Hyperparameter tuning using the train and validation set 
   - Regularization strength
   - Network architecture
2. Backward selection to pick the best features 
3. Retrain on the whole train + validation set 
4. Test on held-out test set and record evaluation metrics 
   - Accuracy, Precision, Recall, F1 Score

Model Training

- **Statistical Learning**
  - **Hyperparameter Search:** Grid search to determine regularization strength.

- **Deep Learning**
  - **Training program**
    - Cross-entropy loss
    - Adam optimizer with learning rate of 1e-4
    - ReLU activation function
    - He initialization
    - Mini-batch size of 8.
  - **Architecture Search**
    - Explore several configurations of models with different depths and layer size.

Feature Selection Results

Backwards selection complete steps with optimal feature set in bold

- MFCC are consistently included in the optimal feature subset indicating their importance for emotion recognition
- For all models, reducing the feature set improves performance

ML Models	Full Model	Backward Step step 1	step 2	step 3	step 4
LR	mfcc, mel, chrom, contrast, global	**mfcc, mel, chrom, contrast**	mfcc, mel, contrast	mfcc, mel	mfcc
SVM_{lin}	mfcc, mel, chrom, contrast, global	**mfcc, mel, chrom, global**	mfcc, mel, chrom	mfcc, mel	mfcc
SVM_{rbf}	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, global	**mfcc, mel, global**	mfcc, mel	mel
MLP	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, contrast	mfcc, mel, contrast	**mfcc, mel**	mfcc
1D CNN	mfcc, mel, chrom, contrast, global	mfcc, mel, chrom, contrast	mfcc, mel, chrom	**mfcc, chrom**	mfcc
2D CNN	mfcc, mel, chrom, contrast	mfcc, chrom, mel	**mfcc, chrom**	mfcc	N/A

Model Evaluation Results

1D Deep learning models easily overfit

- 1D CNN and MLP have near perfect training, suggesting overfitting
- LR outperforms both MLP and 1D CNN

2D CNN outperforms all models

- Suggests that time-series are crucial while global features are not

	Train				Validation				Test			
ML Model	accuracy	precision	f1 score	recall	accuracy	precision	f1 score	recall	accuracy	precision	f1 score	recall
LR	0.710	0.707	0.705	0.707	0.571	0.583	0.557	0.571	0.422	0.476	0.419	0.422
SVM_{lin}	0.798	0.800	0.795	0.798	0.595	0.612	0.589	0.595	0.378	0.431	0.365	0.378
SVM_{rbf}	0.979	0.979	0.979	0.979	0.611	0.611	0.605	0.611	0.400	0.428	0.385	0.400
MLP	0.997	0.997	0.997	0.997	0.738	0.748	0.736	0.738	0.417	0.425	0.384	0.384
1D-CNN	1	1	1	1	0.746	0.769	0.748	0.748	0.356	0.364	0.320	0.320
2D-CNN	0.951	0.952	0.951	0.951	0.714	0.762	0.712	0.714	**0.489**	**0.479**	**0.463**	**0.489**

Georgia Tech
CREATING THE NEXT

Future Directions

1. Design time-series features for 1D models.
2. Implement cross-subject cross validation procedure.
3. Explore more modern feature selection methods.
4. Simplify the classification problem.

End

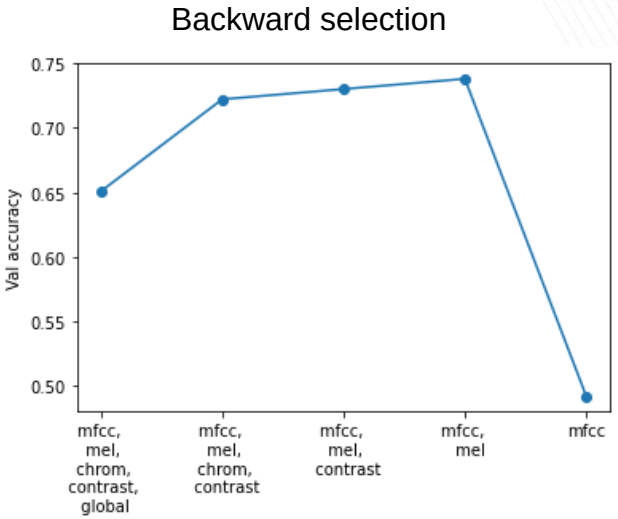
Georgia Tech
CREATING THE NEXT

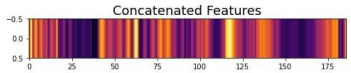
GitHub Repository

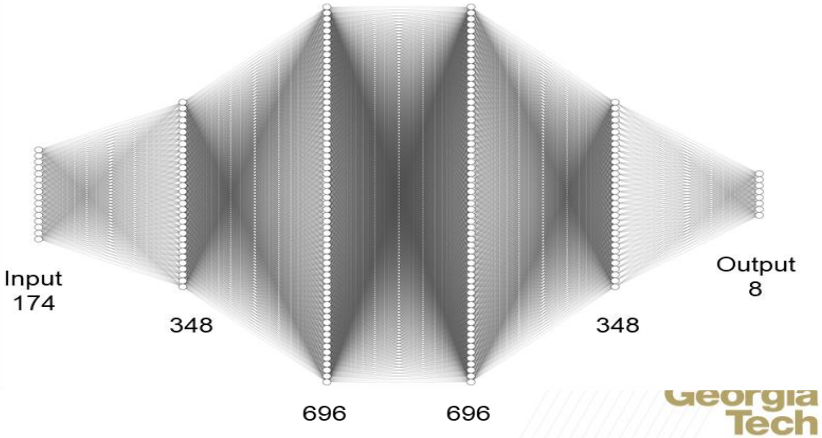
GitHub Repository for Source Codes

Multi-layer perceptron

- **Architecture:**
  - 5-layer fully connected
  - Batch normalization with ReLU activation
  - Dropout to reduce overfitting
  - Realized increasing the size of middle hidden layers improves performance
- **Training:**
  - Cross-entropy loss
  - Adam optimizer
  - Learning rate of 1e-4
  - He initialization
  - Batch size of 8
  - 80 epochs







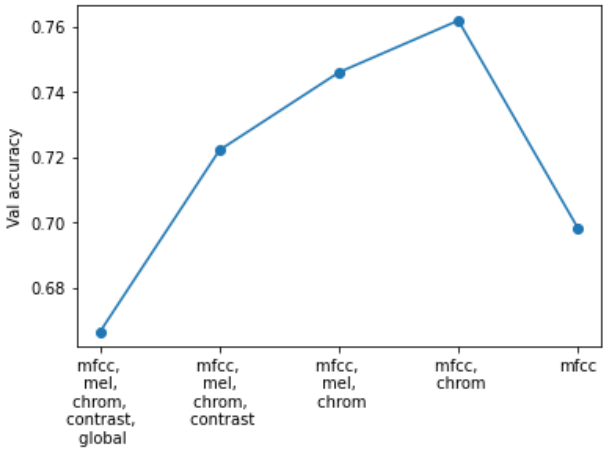
Georgia Tech
CREATING THE NEXT

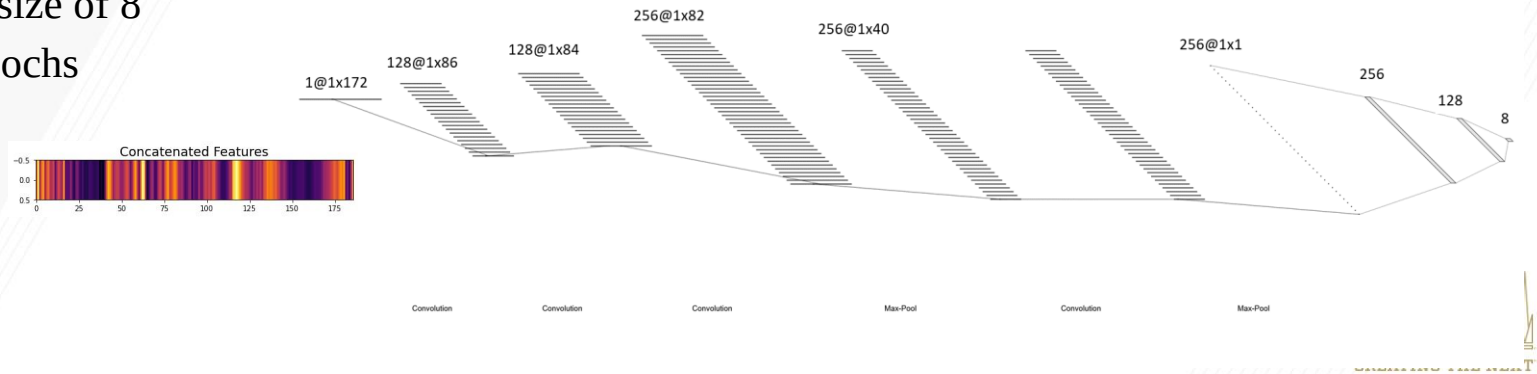
1D CNN

- **Architecture:**
  - 4 convolutional layers
  - Max pooling layers
  - ReLU activation
  - Dropout to reduce overfitting
  - Realized increasing the size of middle hidden layers improves performance

- **Training:**
  - Cross-entropy loss
  - Adam optimizer
  - Learning rate of 1e-4
  - He initialization
  - Batch size of 8
  - 100 epochs

Backward selection





2D CNN

- **Architecture:**
  - 3 convolutional layers
  - Average/max pooling layers
  - ReLU activation
  - Dropout to reduce overfitting
  - Used rectangular filters instead of square to cover more time windows for each frequency feature step
- **Training:**
  - Cross-entropy loss
  - Adam optimizer
  - Learning rate of 1e-4
  - He initialization
  - Batch size of 8
  - X epochs

Add backward selection plot