Lyrics Based Music Genre Classification

Final Presentation

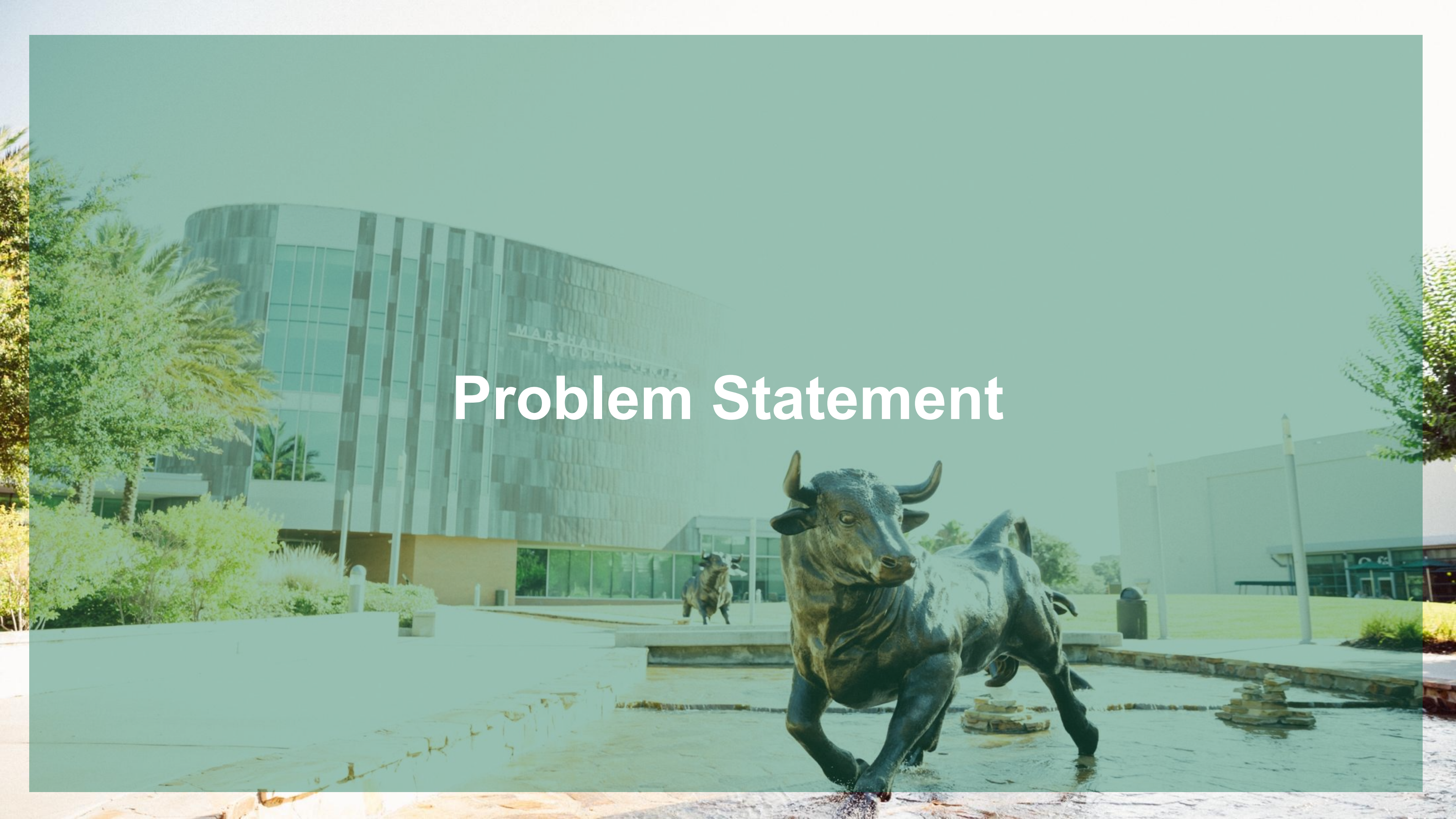
Fida Rahman
Ricardo Frumento
John Ha
Alvaro Montoya Ruiz

UNIVERSITY of SOUTH FLORIDA

Outline

- **Problem Statement**
  - Challenges
  - Proposed Solution
- **Dataset**
  - Preprocessing
  - Statistics
  - Bag of Words
  - Train, Test, Validation Splits
- **Models**
  - SimpleRNN
  - GRU
  - LSTM
  - CNN
- **Next Steps**
  - Accuracy
  - Embedding
  - Testing

Problem Statement



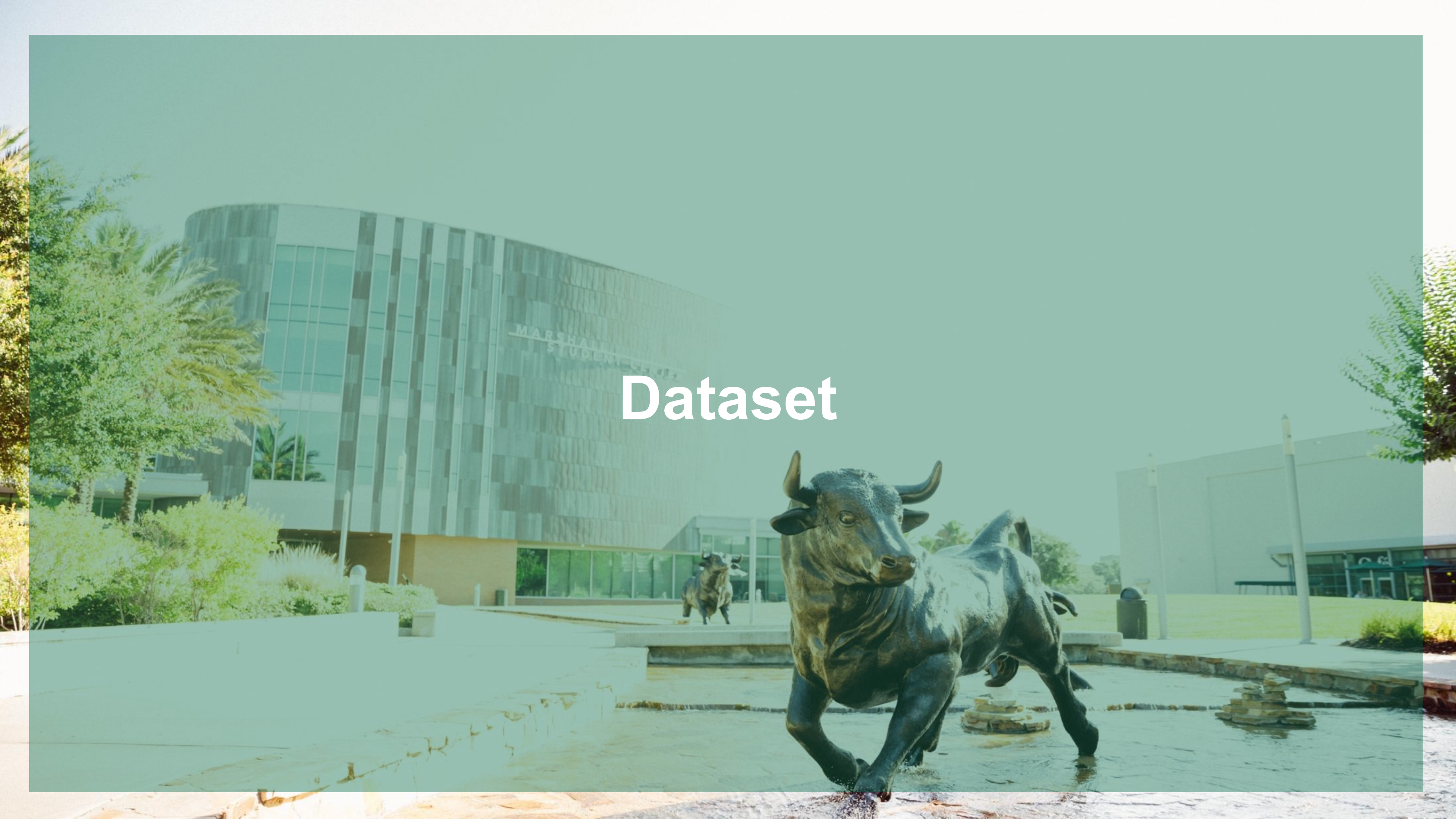
Challenge

- Genre classification is hard when melody, artist name, and other important features are removed from the picture
- Using only lyrics has posed to be a challenging task
- There is no reliable method to classify the music genre using only lyrics

Proposed Solution

- Use natural language processing (NLP) and machine learning (ML) to solve it
- Bag of Words
- Different Models
  - Simple RNN
  - GRU
  - LSTM
  - CNN

Dataset



Preprocessing

- Drop unnecessary columns
- Drop genres with less elements
- Balance the dataset (2500)

```
# Load the dataset
df = pd.read_csv('./tcc_ceds_music.csv', delimiter=',')
df.dataframeName = 'tcc_ceds_music.csv'

df.drop(columns=['Unnamed: 0', 'len', 'dating', 'violence', 'world/life', 'night/time',
        'shake the audience', 'family/gospel', 'romantic', 'communication',
        'obscene', 'music', 'movement/places', 'light/visual perceptions',
        'family/spiritual', 'like/girls', 'sadness', 'feelings', 'danceability',
        'loudness', 'acousticness', 'instrumentalness', 'valence', 'energy',
        'topic', 'age', 'artist_name', 'track_name', 'release_date'], inplace=True)

df.drop(df[df['genre'] == 'hip hop'].index, inplace = True)
df.drop(df[df['genre'] == 'blues'].index, inplace = True)
df.drop(df[df['genre'] == 'jazz'].index, inplace = True)
```

```
[37] N = 2500
     df = df.groupby('genre')\
         .apply(lambda x: x[:N][['genre', 'lyrics']])
```

Statistics

- Lyrics - 9998
- Genre distribution
  - Country - 2500
  - Pop - 2500
  - Rock - 2500
  - Reggae - 2498
- Average number of words per song - 69.68

Bag of Words

Largest frequency of words per genre

Genre: pop	Genre: reggae	Genre: rock	Genre: country
know : 2445	like : 3790	know : 2410	know : 2438
come : 2024	know : 3577	time : 2204	time : 2083
time : 1901	come : 2951	come : 1927	heart : 1862
heart : 1673	time : 2830	like : 1810	come : 1563
away : 1521	yeah : 2492	away : 1732	like : 1463
like : 1491	life : 2212	feel : 1497	away : 1328
baby : 1369	live : 1981	yeah : 1442	leave : 1113
life : 1219	cause : 1770	life : 1395	life : 1078
feel : 1186	feel : 1760	want : 1334	long : 1038
leave : 1115	want : 1655	live : 1299	night : 1005

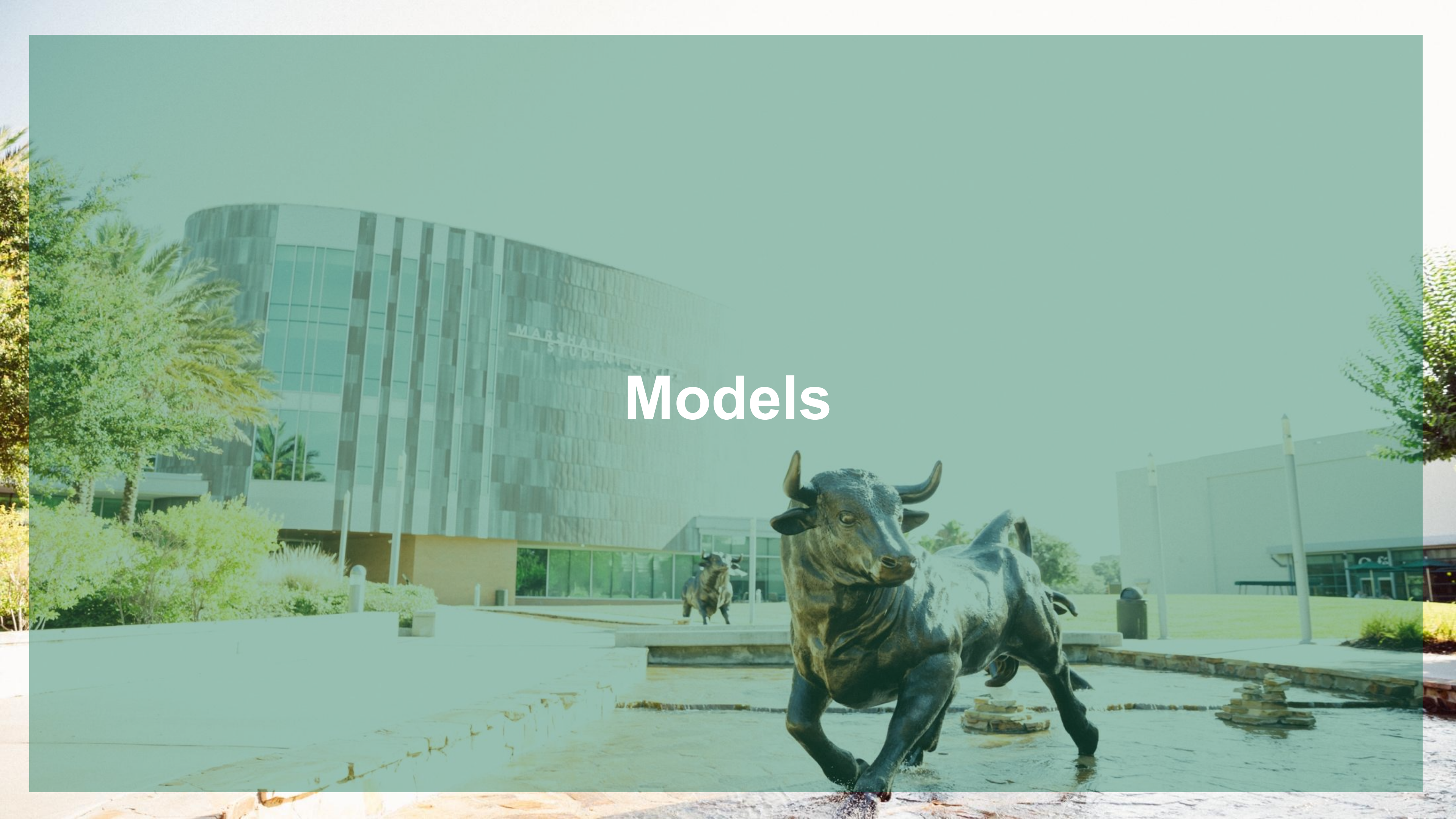
Splits

- First Split (Entire Dataset)
  - 90% - Train and Validation
  - 10% - Test

- Second Split (90% of Dataset)
  - 80% - Train
  - 20% Validation

```
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(df['lyrics'], df['genre'], test_size=0.1, random_state=42)

# Split the test data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(df['lyrics'], df['genre'], test_size=0.2, random_state=42)
```

Models



Parameters

- Regularizer - L1 1e-5 and L2 1e-4
- Dropout Rate - 50%
- Optimizer - RMSProp
- Loss - SparceCategoricalCrossEntropy

SimpleRNN implementation

Fully-connected RNN where the output is to be fed back to input.

Preliminary Result

- 35% Accuracy
- No improvement during training

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_11 (Embedding)     (None, 100, 128)          3094272

dropout_14 (Dropout)         (None, 100, 128)          0

simple_rnn_1 (SimpleRNN)     (None, 16)                2320

dropout_15 (Dropout)         (None, 16)                0

dense_11 (Dense)             (None, 4)                 68

=================================================================
Total params: 3,096,660
Trainable params: 3,096,660
Non-trainable params: 0
_________________________________________________________________
```

GRU implementation

Gated Recurrent Unit -

Cho et al. 2014

Preliminary Result

- 50% Accuracy
- Signs of overfitting

Layer (type)	Output Shape	Param #
embedding_10 (Embedding)	(None, 100, 128)	3094272
dropout_12 (Dropout)	(None, 100, 128)	0
gru_1 (GRU)	(None, 16)	7008
dropout_13 (Dropout)	(None, 16)	0
dense_10 (Dense)	(None, 4)	68

Total params: 3,101,348
Trainable params: 3,101,348
Non-trainable params: 0

LSTM implementation

Long Short-Term Memory layer - Hochreiter 1997

Preliminary Result

- 49% Accuracy
- Signs of overfitting

Layer (type)	Output Shape	Param #
embedding_9 (Embedding)	(None, 100, 128)	3094272
lstm_6 (LSTM)	(None, 16)	9280
dropout_11 (Dropout)	(None, 16)	0
dense_9 (Dense)	(None, 4)	68

Total params: 3,103,620
Trainable params: 3,103,620
Non-trainable params: 0

CNN implementation

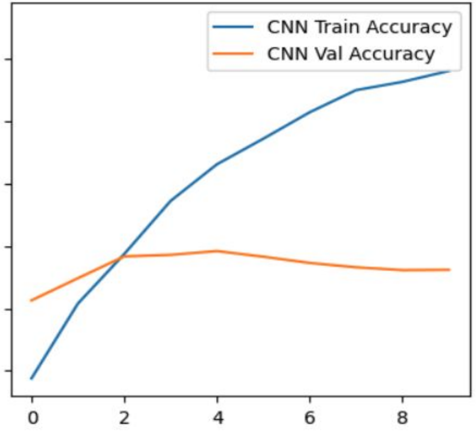
1D convolution layer
(e.g. temporal convolution)

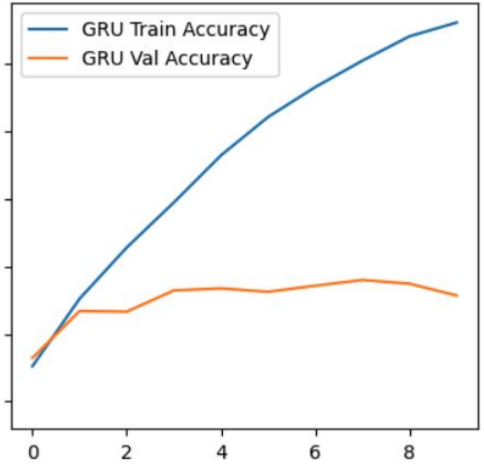
Preliminary Result

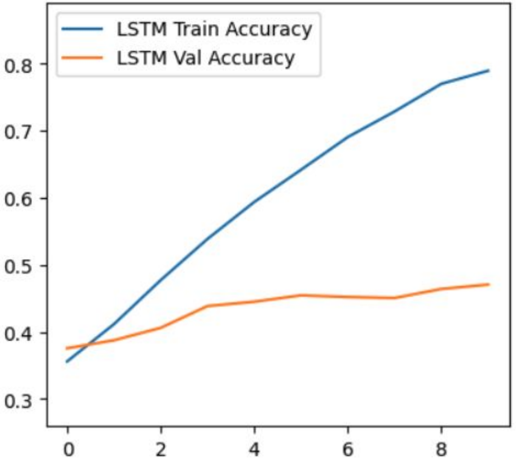
- 49% Accuracy
- Signs of overfitting

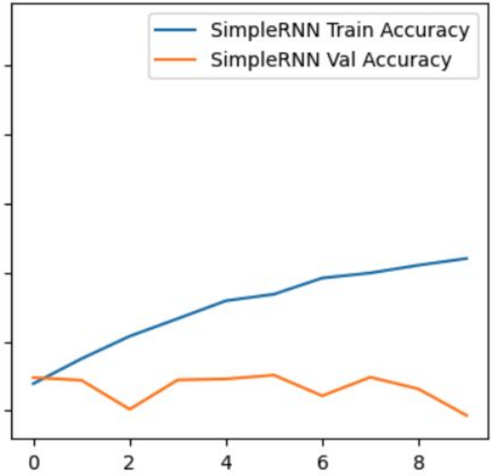
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 60, 128)           3094272

 conv1d (Conv1D)             (None, 58, 16)            6160

 global_max_pooling1d (Globa  (None, 16)               0
 lMaxPooling1D)

 flatten (Flatten)           (None, 16)                0

 dense_3 (Dense)             (None, 4)                 68

=================================================================
Total params: 3,100,500
Trainable params: 3,100,500
Non-trainable params: 0
_________________________________________________________________
```

Plots


CNN Train Accuracy
CNN Val Accuracy
0
2
4
6
8

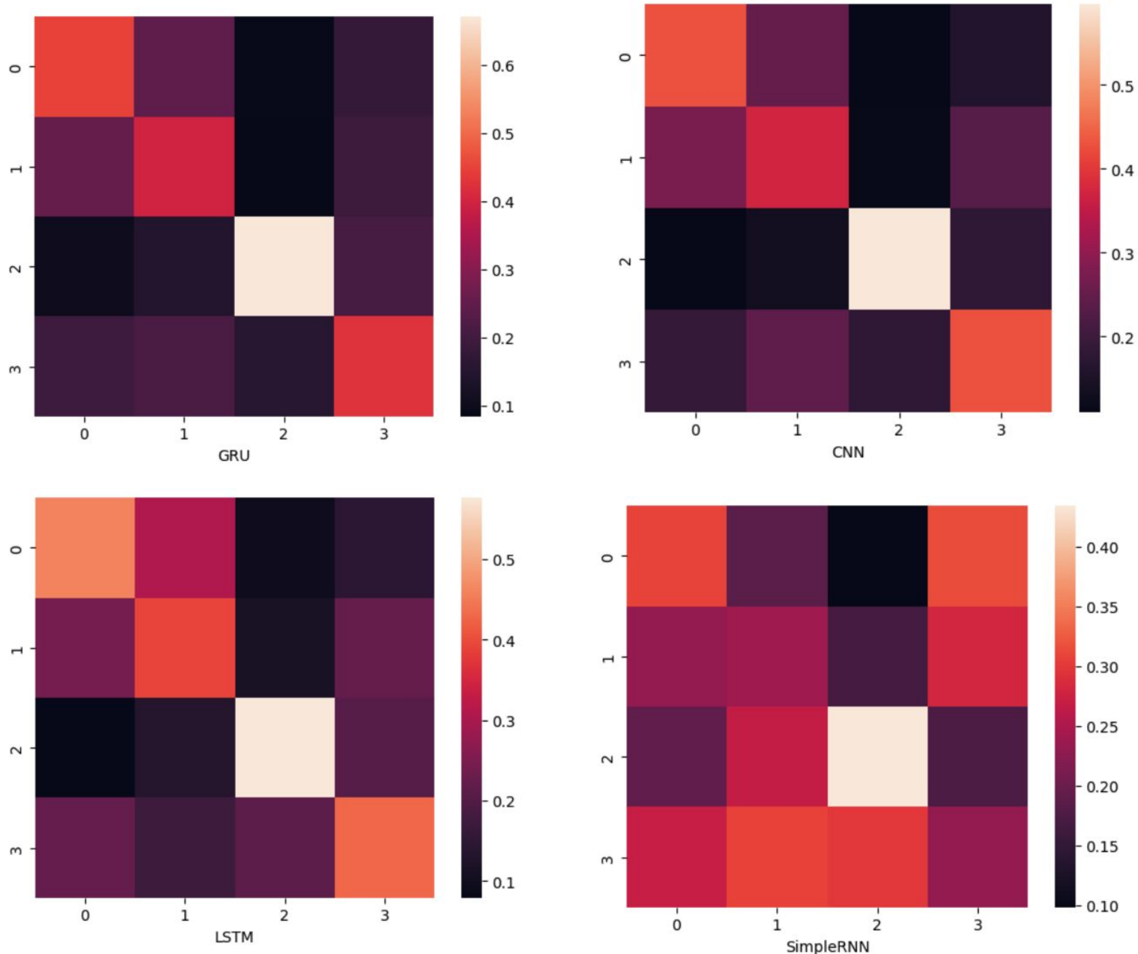

GRU Train Accuracy
GRU Val Accuracy
0
2
4
6
8


LSTM Train Accuracy
LSTM Val Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8

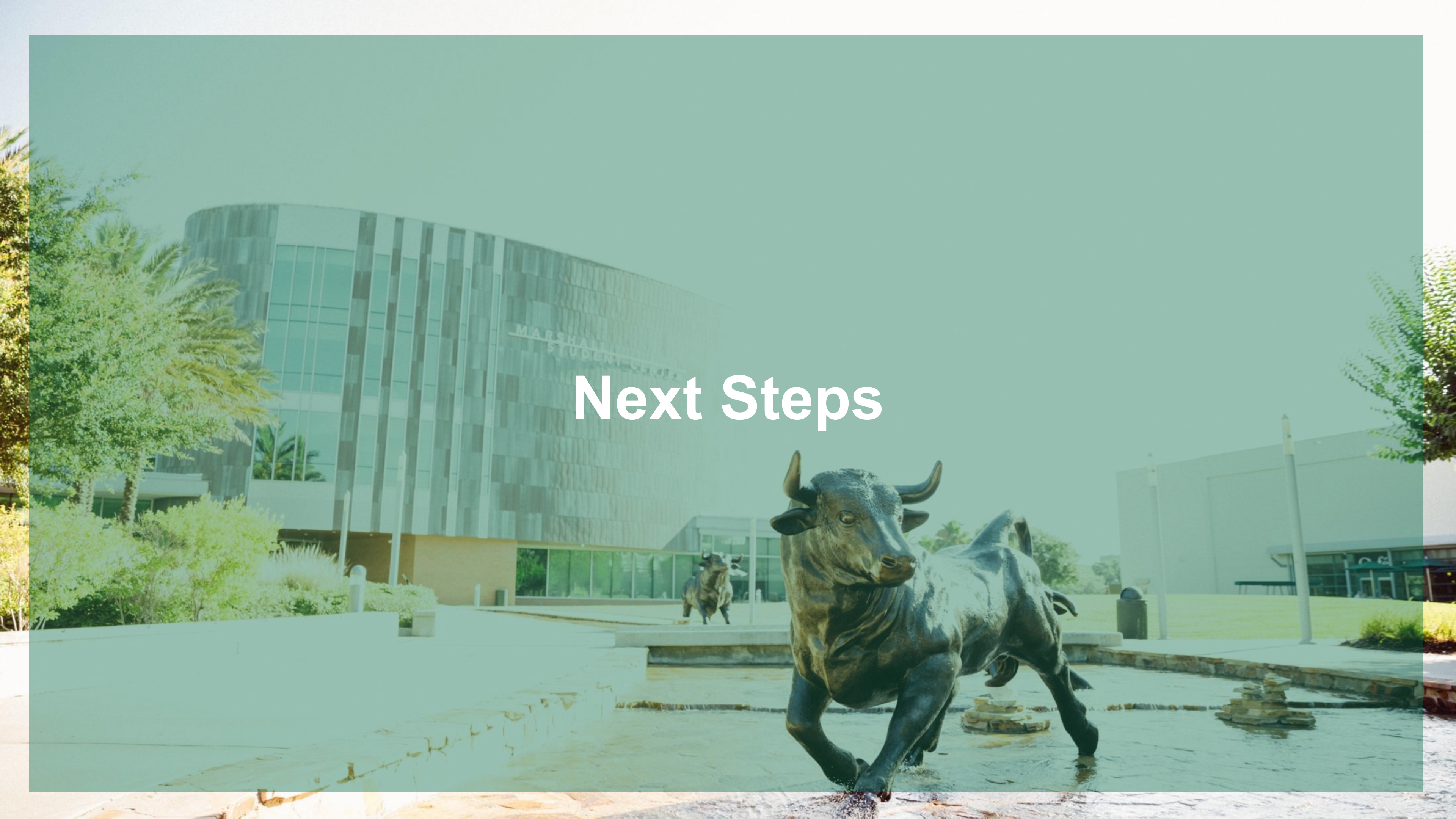

SimpleRNN Train Accuracy
SimpleRNN Val Accuracy
0
2
4
6
8

Confusion Matrices

- 0 : Pop
- 1 : Reggae
- 2 : Rock
- 3 : Country



Next Steps



Improve Accuracy

- GRU, LSTM, and CNN show indications of overfitting
  - Regularization was already implemented
  - Dropout as well
  - Hyperparameter tuning will be executed
- SimpleRNN does not improve during training
  - Different architecture is needed, will be tested
- Weighted Loss Function
  - Fight unbalanced dataset

Embedding

- Right now a simple embedding layer is being added to the model
- GloVe is being studied and might be used to improve performance(these txt files are made available under the Public Domain Dedication and License)
-
    - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB)
    - Common Crawl (840B tokens, 2,2M vocab, cased, 300d vectors, 2.03 GB)
    - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 300d vectors, 822 MB)
    - Twitter (2B tweets, 27B Tokens, 1.2M vocab, uncased, 200d vectors, 1.42 GB)

Testing

- Include random network
  - Frozen weights after random initialization as baseline

- After a model shows a clear advantage over the others
  - 5 runs
  - 20 epochs

- Then it will be compared with the literature review
  - Lyrics based models
  - Lyrics and audio based models
  - Audio based models

- Expected Accuracy
  - Between 50% and 60%

References

- Moura, Luan et al (2020), “Music Dataset: Lyrics and Metadata from 1950 to 2019”, Mendeley Data, V2, doi: 10.17632/3t9vbwxgr5.2
- Martín Abadi et al. (2015) “TensorFlow: Large-scale machine learning on heterogeneous systems”
- Jeffrey Pennington et al (2014), “GloVe: Global Vectors for Word Representation”

Questions