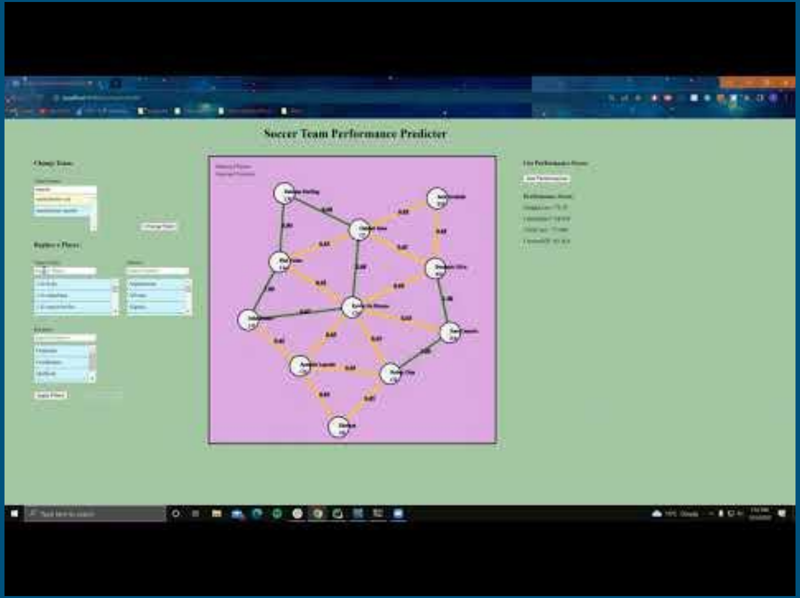
Analyzing Player and Team Chemistry in Soccer

Tusheet Goli, Tejas Pradeep, Aman Jain, Akshay Pramod, Jeffrey Chang

Demo and GitHub Repo

- Link to demo - **https://youtu.be/igk_jDorMGQ**
- Link to GitHub repository - **https://github.gatech.edu/tgoli3/gt-bds-f22-team5**

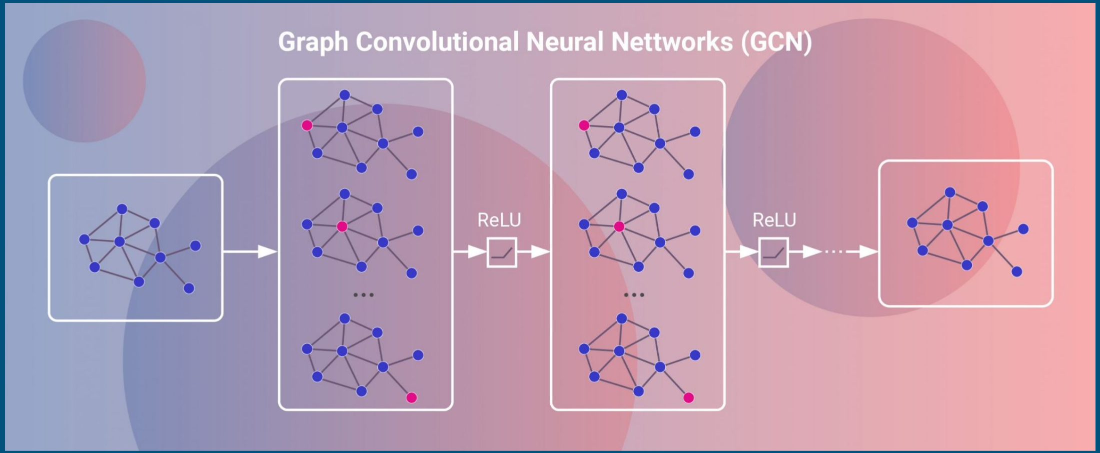


Introduction

- Create a soccer team chemistry and performance evaluator
- Current approach is heuristic based (nationality and league), not data based
- Build a framework that can assess a team’s projected performance based on various player and team attributes
    - Player attributes: pass completion percentage, goals per game, mentality
    - Team attributes: coach, formation, playing style, history

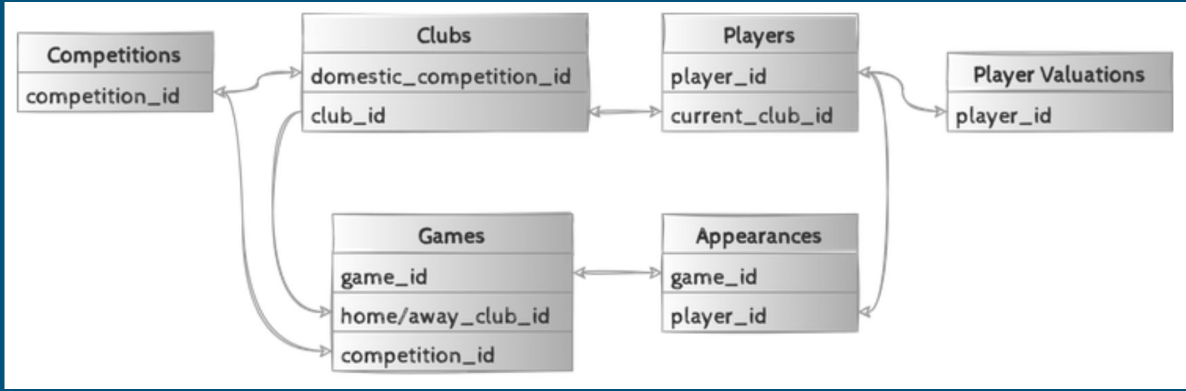
Applications

- Not many team chemistry based, data driven approaches
- New data and new technology (like GNNs) allows us to find additional insights
- Useful for soccer team management, gaming companies like FIFA and PES
- Enable dynamic team selection and scouting beyond individual statistics



Dataset

- Football Transfermarkt Dataset (https://data.world/dcereijo/player-scores)
- Up-to-date dataset obtained from the real-time FIFA website
- Has information for 20,000+ players, 350+ clubs, 55,000+ games, and 1,000,000+ player appearance records across the world
- Cleaned and integrated into a MySQL database with 5 related tables
    - players, clubs, games, competitions, appearances



Dataset

- Training:Test Ratio - 70:30

appearances clean data The `appearances` asset contains one records per player a...

View

	appearance_id	game_id	player_id	player_club_id	date	player_pretty_name
1	2483937_52453	2483937	52453	28095	2014-08-08	Haris Handzic
2	2479929_67064	2479929	67064	28095	2014-08-03	Felicio Brown Forbes
3	2483937_67064	2483937	67064	28095	2014-08-08	Felicio Brown Forbes
4	2484582_67064	2484582	67064	28095	2014-08-13	Felicio Brown Forbes
5	2485965_67064	2485965	67064	28095	2014-08-16	Felicio Brown Forbes

games clean data The `games` asset contains one row per game in the dataset. ...

View

	game_id	competition_id	competition_type	season	round	date
1	2219794	BESC	other	2011	Final	2012-07-22
2	2244388	SUC	other	2012	final 1st leg	2012-08-22
3	2211607	NLSC	other	2012	Final	2012-08-05
4	2252846	POSU	other	2012	Final	2012-08-11
5	2229332	DFL	other	2012	Final	2012-08-12

clubs clean data The `clubs` asset contains one row per club in the dataset. ...

View

	club_id	name	pretty_name	domestic_competition_id	total_market_value
1	1032	fc-reading	Fc Reading	GB1	33.
2	2323	orduspor	Orduspor	TR1	No dat
3	1387	acn-siena-1904	Acn Siena 1904	IT1	4.
4	1071	wigan-athletic	Wigan Athletic	GB1	12.
5	2703	spartak-vladikavkaz	Spartak Vladikavkaz	RU1	No dat

competitions clean data The `competitions` asset contains one row per competiti...

View

	competition_id	pretty_name	type	sub_type	country_id	count
1	L1	Bundesliga	domestic_league	first_tier	40	Germany
2	DFB	Dfb Pokal	domestic_cup	domestic_cup	40	Germany
3	DFL	Dfl Supercup	other	domestic_super_cup	40	Germany
4	NL1	Eredivisie	domestic_league	first_tier	122	Netherlands
5	NLP	Toto Knvb Beker	domestic_cup	domestic_cup	122	Netherlands

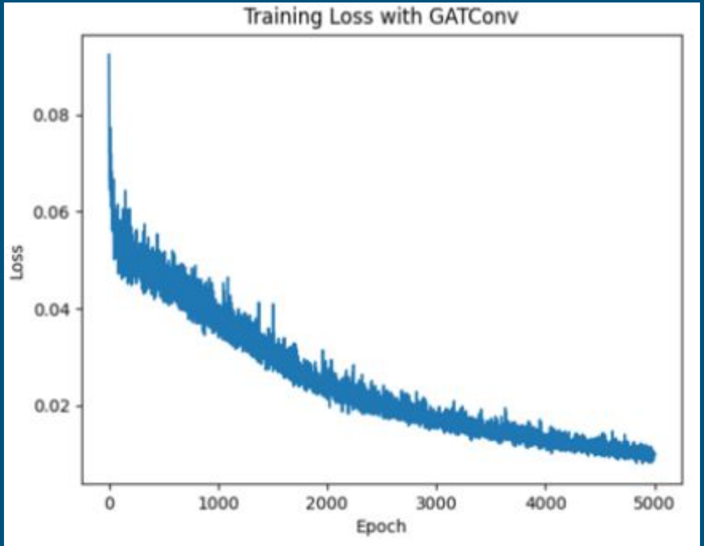
Previous Models

- We had three models from our previous implementation
  - GraphConv
  - GATConv
  - GCNConv
- Improved these models
  - Fine tuned them
  - Trained on newer and up-to-date datasets
- They serve as the baseline models

Reference to a previous study -
https://drive.google.com/file/d/1vSZWUgDxwR8u8hkbxKlK5brMbjWtbEzk/view?usp=sharing

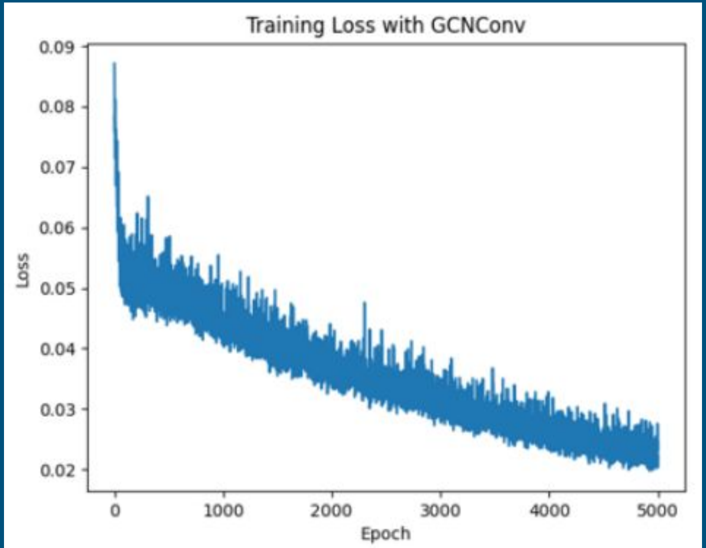
Model 1 - GATConv

- GATConv
  - GATConv incorporates a graph attention operator by utilizing the abilities of a attention network
  - Data Features Trained = ["position", "foot", "height_in_cm", "market_value_in_gbp"]
- Test MSE Error: 0.0185



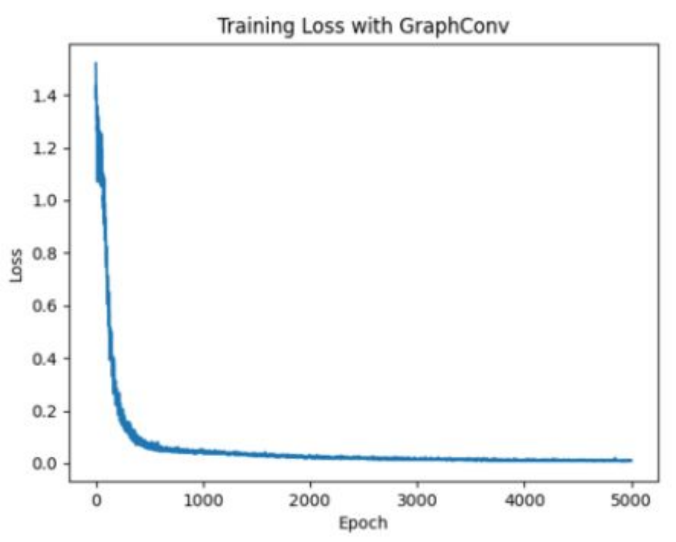
Model 2 - GCNConv

- GCNConv
  - GCNConv takes the weighted average of all it neighbouring nodes’ features and nodes with lower degrees get larger weights
  - Data Features Trained = ["position", "foot", "height_in_cm", "market_value_in_gbp"]
- Test MSE Error: 0.0136



Model 3 - GraphConv

- GraphConv
  - GraphConv is simple convolutional graphical network where you apply convolutions over a graph
  - Data Features Trained = ["position", "foot", "height_in_cm", "market_value_in_gbp"]
- Test MSE Error: 0.0251
- Best results of the 3 older models
- Used as baseline model



New Models

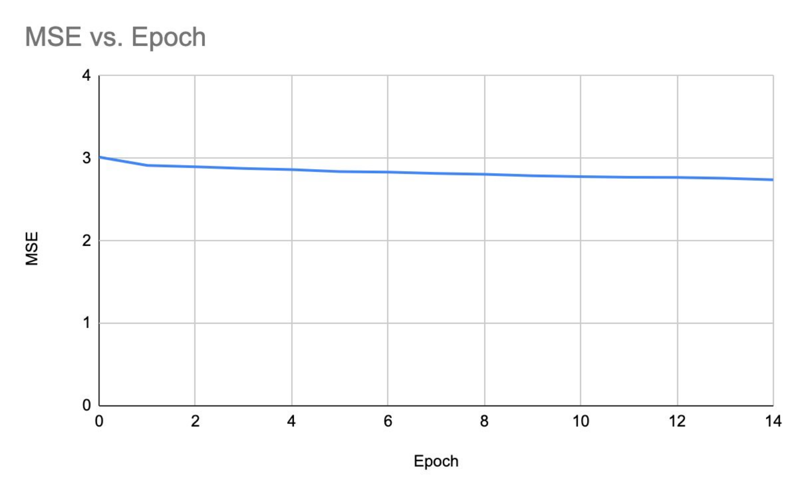
- We architected 3 new models to improve performance
  - Stat-Base Neural Network
  - TAGConv
  - ChebyShev
- Validated the models on our baseline
  - GraphConv

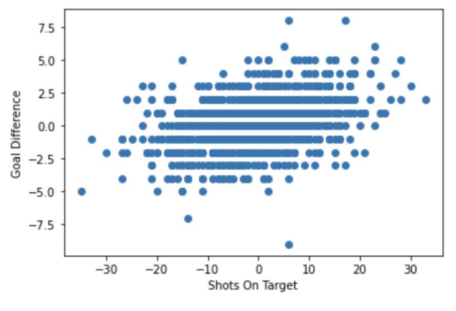
Model 1 - Stat-Based Neural Network

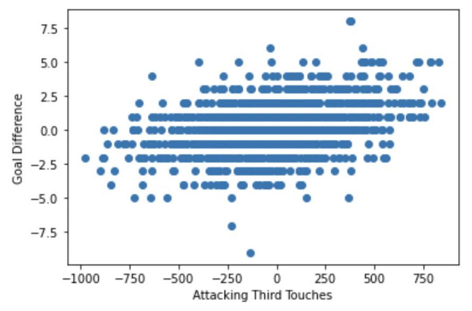
- **Overview:**
  - English Premier League ONLY (because of limited data set)
  - Use 40+ player & team statistics from games (i.e. assists, passes in the attacking third, tackles won, dribbles, etc.) to train a neural network
  - Network is trained on a dataset of ~1400 Premier league games over the last 6 years
    - Network is trained to predict goal difference between two teams
  - Inference is done on the aggregate statistics of the 11 players inputted
    - 11 player team provided is compared to the “statistically average” EPL team
    - Goal difference between team provided and Average Team is then normalized to a score between 0-100

Model 1 - MSE Convergence

- Converges rapidly, since the model is simple (3 layer linear neural network)
- MSE of ~ 2.8
  - This is in goal difference error - seems high because in practice, the model rarely predicts Goal Differences outside of the range [-1, 1], so outliers (which are not infrequent) increase MSE by a lot
  - Correlations are weak (graphs on right)
    - Shots on target (left)
    - Attacking third touches (right)







Model 1 - Model Evaluation

- Dataset size: 1301, taken from https://fbref.com/en/
  - Training:Test Ratio - 90:10
- Training time - 0.7 seconds for 15 epochs
- Training MSE: 2.68
- Testing MSE: 2.83
- Throughput: 195 inferences/second
- Model Architecture:
  - nn.Linear(41, 100)
  - nn.Dropout(p=0.5)
  - nn.Linear(100, 50)
  - nn.Dropout(p=0.3)
  - nn.Linear(50, 1)
  - Nn.functional.relu

Model 1 - Model Evaluation (Game Results)

- Same model architecture, compare it to Vegas oddsmakers in predicting game outcome (Win, Draw, or Loss):
    - Oddsmaker data taken from: https://www.football-data.co.uk/englandm.php
- Oddsmaker game prediction accuracy: 57.9%
- Our model prediction accuracy: 56.7%

- Model Architecture:
    - nn.Linear(41, 100)
    - nn.Dropout(p=0.5)
    - nn.Linear(100, 50)
    - nn.Dropout(p=0.3)
    - nn.Linear(50, 3)
    - nn.functional.relu
    - nn.functional.softmax

Model 1 - Simple Cases

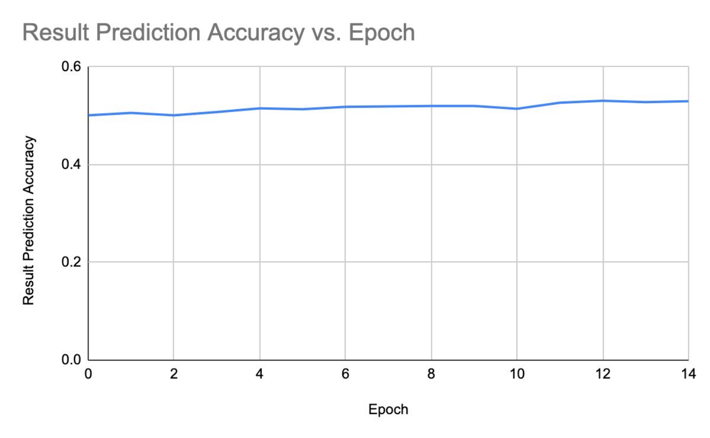
- Teams that already exist:
  - Predicts 2017 Manchester City is a really good team (1st in EPL by 19 points)
    - Score: 83.8/100
  - Predicts 2020 Arsenal is slightly above average (8th in EPL)
    - Score: 55.6/100
  - Predicts 2021 Norwich City is a bad team (Last in EPL)
    - Score: 44.8/100

Model 1 - Challenging Cases

- Players out of position
  - Manchester City
    - Subtract two attackers, add two defenders
    - In reality should be imbalanced, but score is still high: 77.0/100
- No Goalie
  - Manchester City
    - Score nearly identical: 88.5/100
- Entirely one position group
  - Manchester City
    - All attackers (even without a goalie or defenders)
    - Score: 76.8/100
    - Linear model strongly correlates some variables a lot higher than other, especially goals scored and shots on target

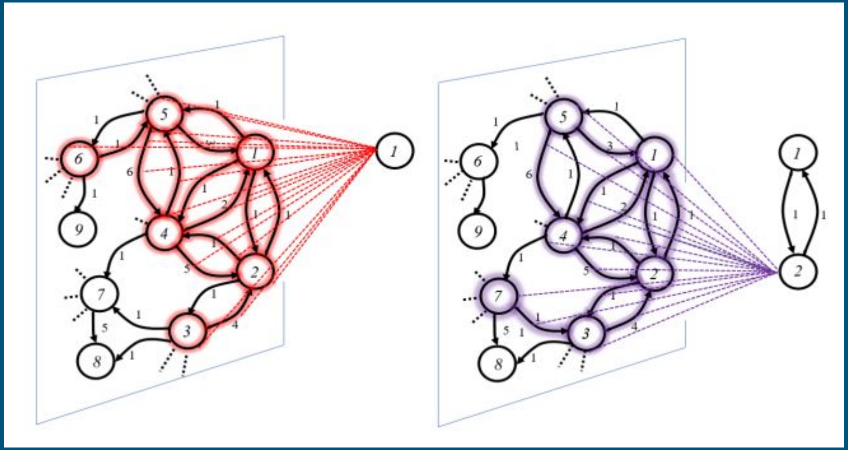
Model 1 - Best Use Cases

- Realistic team compositions
  - Useful for evaluating real trades that teams make
- Ran same model architecture with different evaluation criteria - **predicting game result** (Home Win, Home Loss, Draw)
  - Able to accurately predict 56.7% of games
  - The average Vegas predictions for 2021-2022 season were 57.9%
    - Average of 13 different oddsmakers



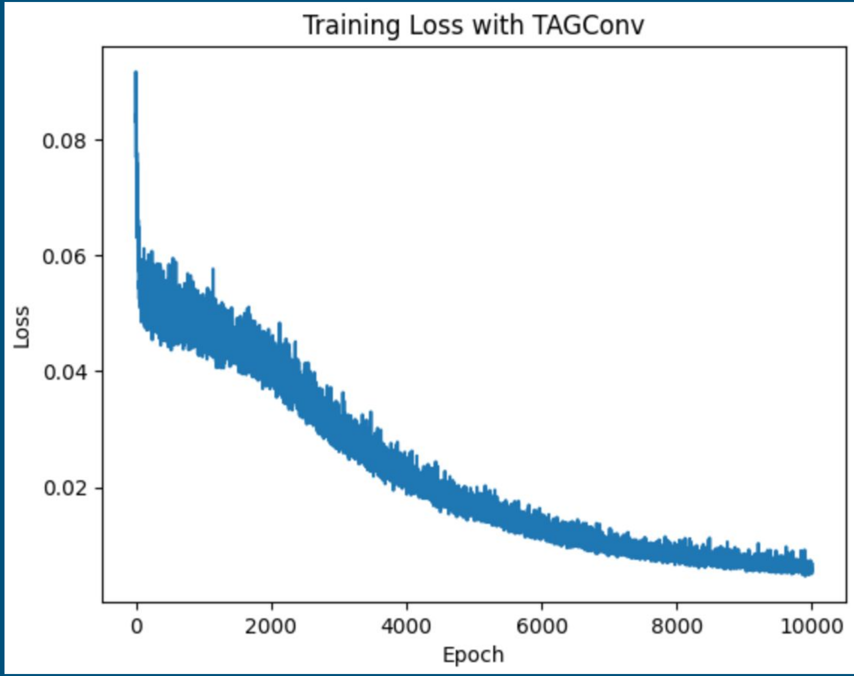
Model 2 - TAGConv

- The second model tested is the Topology Adaptive Graph Convolutional Network (TAGCN)
- Improvements on existing convolutional neural networks, and computationally simpler
- Two main types - spectral domain and vertex domain
- TAGCN creates polynomials of maximum degree 2 for each vertex using a similar convolution to traditional CNNs
- These polynomials can be accurately found using eigenvalue projections to avoid the loss of accuracy that normally occurs



Model 2 - TAGConv MSE

- TAGConv Model Details
  - 5 TAGConv layers with ReLU activation functions
  - Adam optimizer
  - Learning rate: 1e-5
  - Loss Criterion: Mean Squared Error Loss



Model 2 - TAGConv Evaluation

- Dataset size - 242 teams
    - 55000 games
    - Training:Test Ratio - 70:30
- Training time - 64 minutes for 10000 epochs
- Throughput - 32 inferences/second
- Training Loss: 0.0041
- Test Loss: 0.0131

Model Architecture

- TAGConv(k=3) + ReLU
- TAGConv(k=3) + ReLU
- TAGConv(k=3) + ReLU
- TAGConv(k=3) + ReLU
- TAGConv(k=3) + ReLU
- Mean Pool
- Dropout
- Linear
- Linear
- Sigmoid

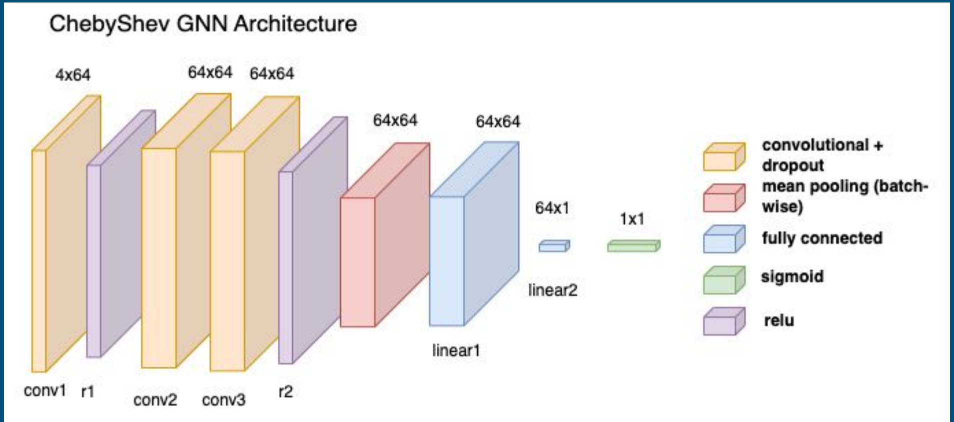
Model 2 - TAGConv Evaluation (Prediction Results)

- Evaluated model by using it to predict game results (real world data from the same dataset)
- Model was not trained to predict ties - picks whichever team has a higher score to only win
- Results:
  - 65.22% accuracy when ignoring tied games
  - 53.06% accuracy including all games
  - Consistent - almost every team was between 40% and 60% prediction accuracy, although there were some outliers

- Most popular teams also seemed to have high prediction accuracy
  - Barcelona 65.4%
  - Real Madrid 57.4%
  - PSG 67.5%

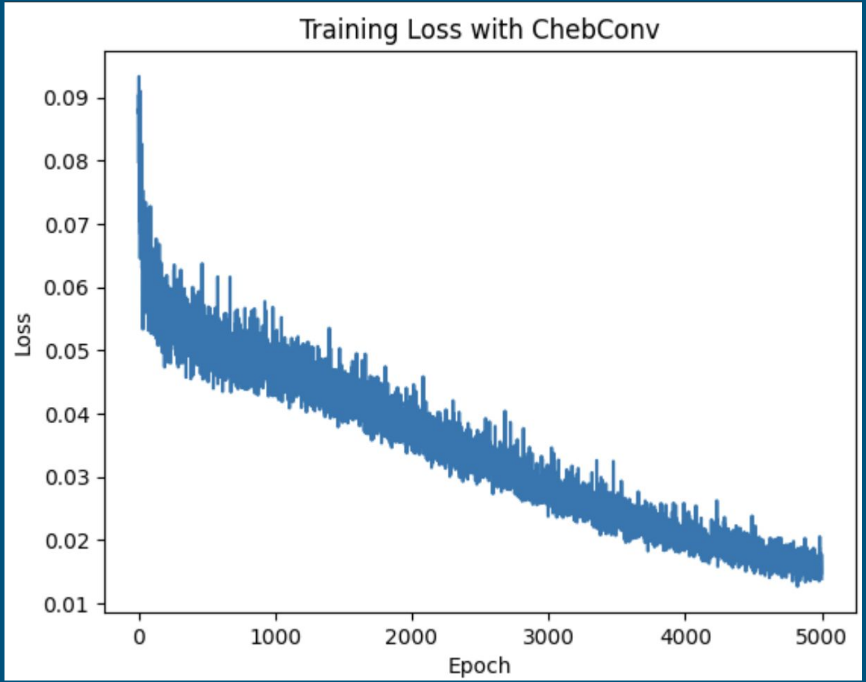
Model 3 - ChebyShev

- ChebyShev spectral graph convolution operator from “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”
  - Torch Geometric (GNN library)
- Model Features:
  - ChebyShev Filter Size (k) = 5
  - Data Features Trained = ["position", "height_in_cm", "market_value_in_gbp", “country”]



Model 3 - ChebyShev MSE

- Optimizer Parameters:
  - Adam Optimizer
  - Learning Rate = 1e-5
  - Loss Criterion: Mean Squared Error (MSE) Loss



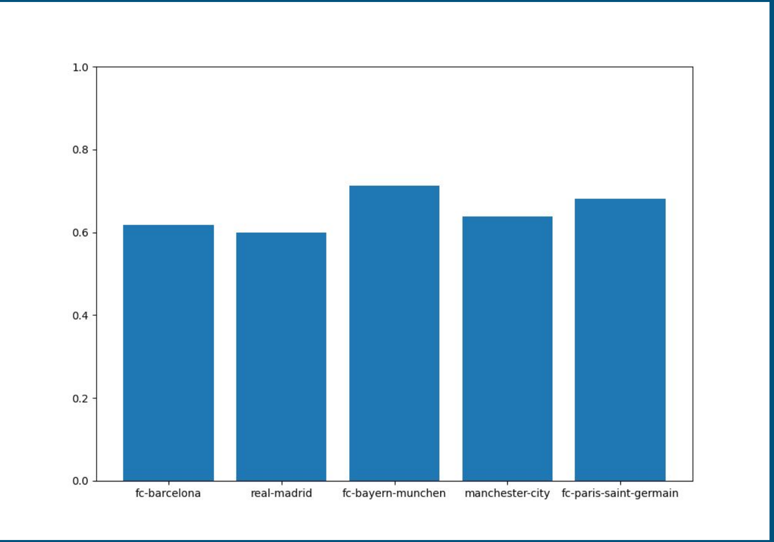
Model 3 - ChebyShev Evaluation

- Dataset Size: 242 teams, 242*11 = 2662 players
  - Utilizes Transfermarkt Dataset (see slide 5)
  - Training:Test Ratio - 70:30
- Training Time - 25 minutes across 5000 epochs
- Throughput: 78 inferences/s
- Training MSE: 0.003
- Test MSE: 0.014

- Model Architecture:
  - ChebConv(k=5) + ReLU
  - Dropout(p=0.5)
  - ChebConv(k=5)
  - Dropout(p=0.5)
  - ChebConv(k=5) + ReLU
  - Dropout(p=0.5)
  - Mean Pooling + ReLU
  - Linear_1
  - Linear_2
  - Sigmoid

Model 3 - ChebyShev Evaluation (Game Results)

- Similar to Model 1, we evaluated ChebyShev on Game Data
- Game Results Data comes from Transfermarkt SQL DB
  - Simplifying assumption: Team with greater strength score “should” win. Ignore Ties.
- Some Results:
  - Correctly able to predict 63% of games overall ignoring tied games.
  - Most Consistent Club: Union FC - Belgium (74%)
    - Win Games they’re supposed to win, lose games they’re supposed to lose
  - Most Inconsistent Club: Apollon Smyrnis - Greece (31%)



Game Result Accuracies for Popular Teams

Right to Left: FC Barcelona, Real Madrid, Bayern Munich, Manchester City, PSG

Easy Classification Tasks

- Case: Running the model on an existing well performing team
  - Test for if the model is performing correctly
- Example:
  - Consider the current Champions League champions, Real Madrid
  - Any model used to predict team performance and chemistry must have a high performance prediction for Real Madrid
  - Similar approach can also be used for other clubs and national teams that have a good performance history

Easy Classification Tasks

- **Case: Run on Real Madrid base/starting team**

MODEL	Original Model	Model 2 - TAGConv	Model 3 - ChebyShev
STRENGTH SCORE (out of 100)	84.162	74.617	93.598

Medium Classification Tasks

- Lewandowski as goalkeeper
  - Lewandowski is Barcelona’s current striker
  - Playing him out of position should lead to a worse team
  - Expected: Significantly worse results with the change
  - Tests the model’s ability to perceive position
- Griezmann to PSG
  - Griezmann is a world class French player
  - Consider a transfer to french side PSG to replace Neymar, a better player
  - Expected: Team becomes slightly worse with the change
  - Tests the model’s reliance on chemistry over performance
- Ronaldo to Manchester United
  - Ronaldo is a world class striker who has a history of playing with Manchester United
  - This recent transfer was expected to have good results
  - In reality the transfer lead to suboptimal results
  - Expected: Model predicts worse results for the team
  - Tests the model’s ability to see hidden attributes that can predict a negative result

Medium Classification Tasks

- **Case 1: Put Lewandowski as goalkeeper for Barcelona**
- **Case 2: Add Griezmann to PSG (replace an objectively stronger player in Neymar)**
- **Case 3: Add Ronaldo to Manchester United as a striker**

CASES	MODEL	Original Model	Model 2 - TAGConv	Model 3 - ChebyShev
CASE 1	SCORE BEFORE	77.631	80.301	88.696
	SCORE AFTER	86.974	22.345	39.638
CASE 2	SCORE BEFORE	80.351	78.943	94.73
	SCORE AFTER	80.511	78.682	94.57
CASE 3	SCORE BEFORE	80.344	78.943	94.73
	SCORE AFTER	65.976	82.828	94.68

Hard Classification Tasks

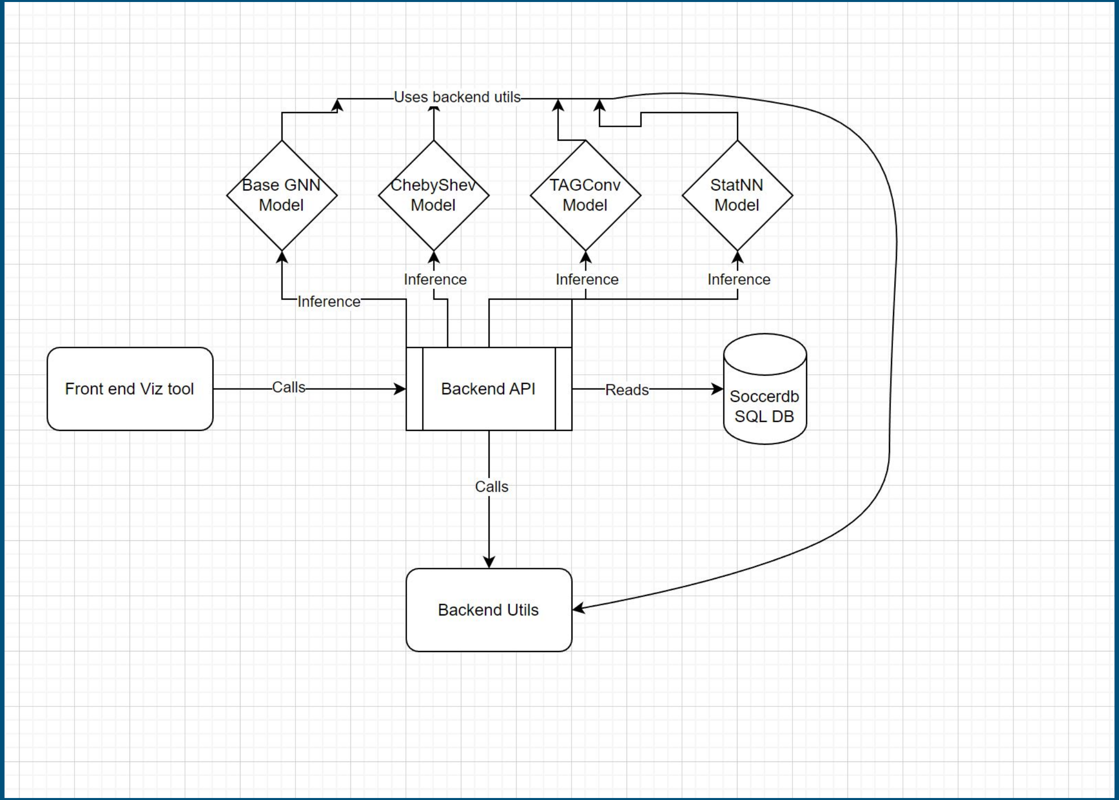
- Case: If there is little data and history between players when considering a new player for a team
  - A situation that is very difficult to predict due to the lack of data and uncertainty
- Example:
  - Additional of an established player to a good squad, but the player impacting the squad in an an unreliable way
  - Transfer of Lewandowski to Barcelona

Hard Classification Tasks

- **Case: Add Lewandowski to Barcelona as a striker**

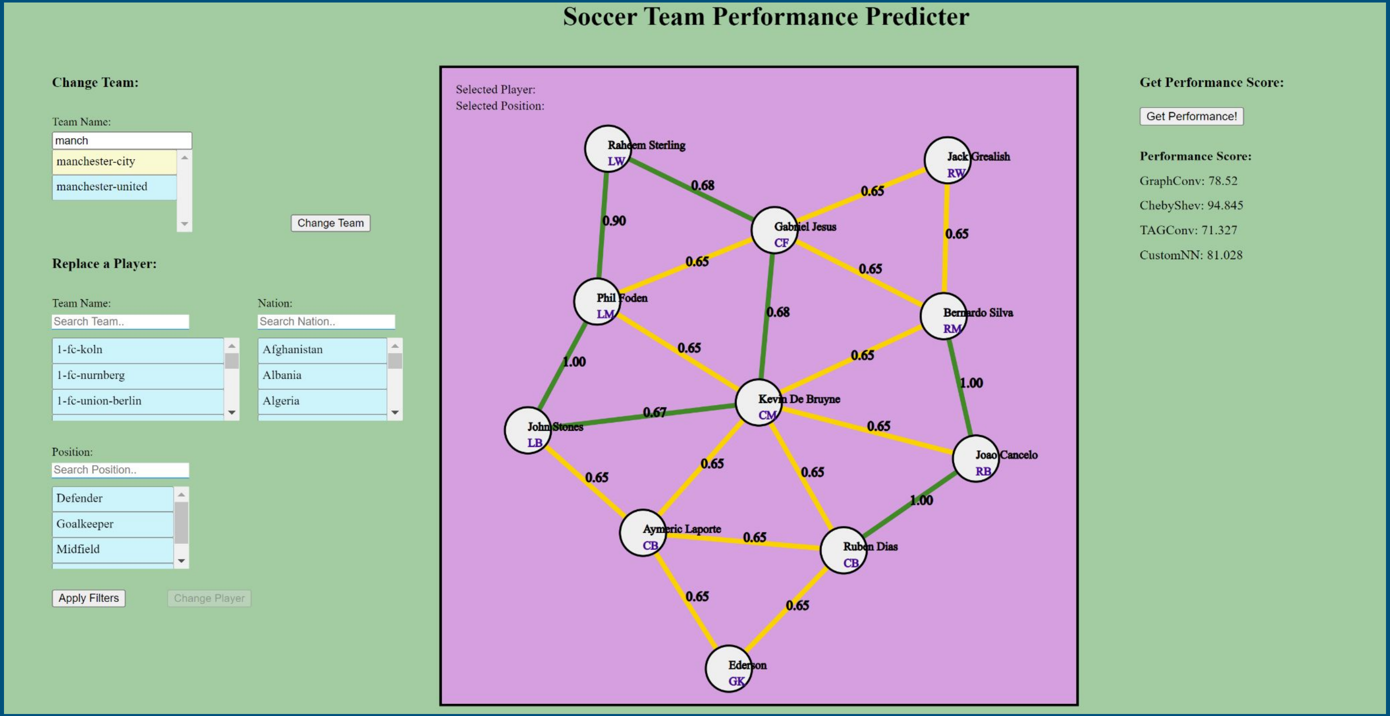
MODEL	Original Model	Model 2 - TAGConv	Model 3 - ChebyShev
STRENGTH SCORE BEFORE (out of 100)	77.656	80.301	88.696
STRENGTH SCORE AFTER (out of 100)	73.638	85.054	90.098

System Architecture



Viz Tool

- Data:
  - MySQL (.sqlite)
- Backend:
  - Python Flask
- Frontend:
  - ReactJS
- Database Hosting:
  - Heroku DB



API Endpoints [GET]

- GET Endpoints
  - /get_all_teams
    - Returns JSON object with all teams
  - /get_all_players_from_club/<club_name>
    - Returns JSON object with all pliers from one team
  - /search_player/<player_name>/<club_name>/<nationality>/<position>
    - Returns JSON object that represents a search value for a player from the player filter
    - All params are optional
  - /calculate_score
    - Returns JSON object with team performance score returned by various models
  - /get_edge_weight/<player_1_id>/<player_2_id>
    - Returns JSON object with edge weight between two players, player_1 and player_2 passed in

GET Endpoint Example

```python
@app.route('/search-player/<player_name>/<club_name>/<nationality>/<position>', methods=['GET'])
def search_player(player_name, club_name, nationality, position):
    conn, cursor = create_conn()
    # filter variables
    if player_name == '*': player_name = ''
    if club_name == '*': club_name = ''
    if nationality == '*': nationality = ''
    if position == '*': position = ''
    # default (all players)
    if player_name == '' and club_name == '' and nationality == '' and position == '':
        player_list = []
        query = 'SELECT pretty_name FROM players ORDER BY pretty_name ASC;'
        cursor.execute(query)
        res = cursor.fetchall()
        for i in res:
            player_list.append(i[0])
        return jsonify({'name': 'search-player', 'status': 'ACTIVE', 'number_of_players': len(player_list), 'players': player_list})
    # get club_id
    q = 'SELECT club_id FROM clubs WHERE name LIKE %s OR pretty_name LIKE %s;'
    cursor.execute(q, ['%' + str(club_name) + '%', '%' + str(club_name) + '%'])
    r = cursor.fetchall()
    if (len(r) == 0):
        return jsonify({'name': 'search-player', 'status': 'ERROR', 'message': 'INVALID CLUB NAME'})
    player_list = []
    # get all players based on club id
    for club_id in r:
        club_id = club_id[0]
        query = 'SELECT * FROM players WHERE current_club_id=%s AND (name LIKE %s OR pretty_name LIKE %s) ' \
                'AND country_of_citizenship LIKE %s AND position LIKE %s ORDER BY pretty_name ASC;'
        cursor.execute(query, [str(club_id), '%' + str(player_name) + '%', '%' + str(player_name) + '%', '%' + str(nationality) + '%', '%' + str(position) + '%'])
        res = cursor.fetchall()
        for i in res:
            player_list.append(i[0])
    close_conn(conn, cursor)
    return jsonify({'name': 'search-player', 'status': 'ACTIVE', 'number_of_players': len(player_list), 'players': player_list})
```

API Endpoints [POST]

- /replace_player/<old_player_id>/<new_player_id>
    - Replace old player id with a new player id in the current team representation

```
@app.route('/replace-player/<old_player_id>/<new_player_id>', methods=['POST'])
def replace_player(old_player_id, new_player_id):
    conn, cursor = create_conn()
    global current_team
    current_team = current_team[current_team.player_id ≠ int(old_player_id)]
    query = f'SELECT * FROM players WHERE player_id={new_player_id}'
    player_df = pd.read_sql(query, con=conn)
    current_team = pd.concat([current_team, player_df])
    close_conn(conn, cursor)
    return jsonify({'name': 'replace-player', 'status': 'ACTIVE'})
```

Readme Outline

- Readme contains **Description**, **Installation**, and **Execution** instructions
- Follow **Installation** instructions to install prerequisites and requirements
  - requirements.txt - root directory
- Execution steps to start backend Flask server are in **Execution** instructions
- UI features of the frontend viz tool are also outlined in the Readme file
- Demo video of installation, execution, and viz tool also in Readme

Link to Readme file -
https://github.gatech.edu/tgoli3/gt-bds-f22-team5/blob/main/README.md

Summary

Overall Summary

- Developed a data-driven approach to soccer team chemistry and performance
- Read lot of literature on existing team evaluation techniques
- Improves on existing soccer team evaluators
- Some models perform close to Vegas oddsmakers

Technologies Learned

- Python Flask
- API Design
- Pytorch
- Graph-based Neural Networks
  - GCNConv, TAGConv, GATConv, etc.
- ReactJS
- MySQL
- HerokuDB
- Version Control
  - Git

Thank you!