Smart Stadium: College Football Play Classification

Smart Stadium VIP Machine Learning Team:
Team Leaders: Pranav Raj, Saahil Sharma
Team Members: Shail Patel, Heng-Ju Chen, Tawfiq Mohammad,
Ryan Rodriguez, Yash Arora, Amit Singh,
Rushil Sudunagunta, Andrew Gao, Nick Eliacin
Advisers: Deepa Phanish, Pedro Pinto, Edward J. Coyle
VIP Program, Georgia Institute of Technology, Atlanta, Georgia 30332, USA
{ejc, dnagendra3}@gatech.edu, https://vip.gatech.edu/teams/vp3

***Abstract*—TODO!!! by Dr.Coyle, Pedro, and Deepa! likely short as paper is already at 8 pages of content.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

With hundreds of millions of yearly spectators, American football leagues at all levels are constantly adopting new technologies to advance the sport, enhance viewers' experience and improve performance for players and coaching staffs. From the legendary *Yellow Line* developed by *Sportvision* to Machine Learning powered statistics like AWS' *Next Gen Stats*, technological innovation has become an integral part of the American football experience.

Over the years, the Smart Stadium VIP (Vertically Integrated Project) research team has researched and developed multiple tools and applications focused at enhancing game-day experience for fans and teams. Our pioneering eStadium web application, for example, was one of the first websites to offer features such as a near real-time delivery of play-by-play video replays. In order to generate those replays, however, clips had to be manually sliced and labeled by students without any automation to match the videos with the proper play descriptions.

Using this limitation as an initial inspiration, we decided to start developing machine learning solutions using the years of visual and tabular data collected by our team. On this publication, we explore the current state of our efforts of developing a computer vision model to classify videos of football plays by type as well as other novel machine learning approaches to play prediction. Potential use cases for these technologies include provisioning labeled videos to direct-to-consumer applications, quick retrieval of classified videos for coaches and players, or creating better predictive models to help defensive coordinators better prepare for opposing offenses.

II. Previous Literature

In recent years, there has been an increase in the usage of machine learning (ML) algorithms to analyze and classify video footage of sports. Several studies have explored the application of ML to sports analysis, including action recognition and classification of various sports activities, while other studies have used the predictive nature of machine learning algorithms in order to predict sequences of events within sports. However, the task of classifying plays of team sports are often more difficult due to the involvement of actions from multiple players. Furthermore, in order to achieve better accuracy in these difficult classification tasks, we will use the concept of model fusion and combine the strengths of different machine learning models and compensate for their individual shortcomings. This section provides an overview of previous literature, with a focus on play classification and sports video analysis using deep learning techniques, as well as the fusion of machine learning models.

In particular, past studies have analyzed American football plays and have produced predictions. One such example is Ötting's 2021 paper, studying the potential use of Hidden Markov Models to predict play calls within the NFL. Using a dataset of almost 300k observations, this study was able to get a prediction accuracy of 71.5% for these such play calls in the 2018 NFL season [1].

Another seminal work in this area is the 2017 study by Chen et al. titled ”Play Type Recognition in Real-World Football Video” [2]. The authors proposed a two-stage Convolutional Neural Network (CNN) architecture to recognize offensive and defensive play types in American Football videos. In the first stage, they used a CNN to extract spatial features from video frames, while the second stage utilized temporal CNNs to capture the temporal relationships between frames. The study achieved impressive results, with an overall recognition accuracy of 75.3%.

In addition to these studies, several other works have explored the use of ML algorithms for sports clip analysis, including action recognition and event detection. For example, Li et al. (2018) proposed a novel framework based on CNNs and Long Short-Term Memory (LSTM) networks [3] for recognizing and classifying different types of human action, which has been a fundamental challenge in the world of computing vision. The training set used included all types of sports, which is great input for modeling complex and active human interactions. With the two-layer LSTM structure, their model was able to represent the temporal pattern of visual input much better than the state-of-the-art LSTM model. The study was able to achieve an accuracy rate of 92% in identifying different types of activities for video clips in RGB format.

A key reference in the domain of American Football play recognition is the paper by Siddiquie et al., titled "Recognizing Plays in American Football." The authors presented a novel method for recognizing football plays by utilizing a combination of player tracking and modeling techniques [4]. They developed a probabilistic framework that incorporated spatial and temporal relationships among players, enabling the identification of distinctive play patterns. This approach laid the groundwork for understanding the complex nature of American Football plays and their recognition using machine learning techniques. Furthermore, this study was performed on a similar dataset which included clips of football plays from NCAA teams, including Georgia Tech. We faced similar difficulties as the video clips had different time intervals, camera angles, and quality of the footage.

In the discussion of model fusion, Baccouche et al. (2011) proposed a sequential deep learning method for human action recognition in videos using a combination of 3D Convolutional Neural Networks (3D CNNs) and Long Short-Term Memory (LSTM) networks [5]. This fusion of 3D CNNs and LSTM networks allowed for efficient handling of both spatial and temporal information, leading to improved performance in human action classification tasks. The study demonstrated the benefits of combining different machine learning models to effectively address the challenges of capturing complex patterns in real-world data.

Patel et al. (2015) investigated the fusion of multiple machine learning techniques, including Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and k-Nearest Neighbors (k-NN), for stock market index prediction [6]. By training individual models on historical data and combining their predictions using weighted averaging, the study achieved improved forecasting accuracy compared to standalone models. This demonstrates the effectiveness of fusing diverse models for enhanced predictive capabilities. This study also shows the variety of problems that model fusion can be used to solve and improve accuracy.

Overall, these studies demonstrate the potential of ML algorithms and Model Fusion, including CNNs and Markov Chains, in classifying American football plays and analyzing sports video footage. These techniques can help coaches and players to identify patterns and make more informed decisions during games, improving their overall performance. Furthermore, they also give us an insight into the many ways we could combine different models to achieve better accuracy and get the best out of each standalone model, as these previous works provide a valuable reference for exploring the fusion of machine-learning models in all types of fields and not just sports analytics and play classification. We are inspired by the possibility to combine, create, and develop ingenuity across the field of using Machine Learning to analyze sports clips, specifically in regard to classifying American football plays.

III. Play Classification Computer Vision Model

The main objective of our effort was to develop a computer vision model to automatically classify videos of football plays into one of 5 categories:

’R’	Rushing Play
‘P’	Passing Play
‘X/F’	Extra Point/ Field Goal
‘U’	Punt
‘K’	Kickoff

A. Data Reprocessing and Feature Engineering

Pedro/Delete

B. Model Architecture

This model (Figure 1) utilizes a pre-trained Convolutional Neural Network (CNN) to extract features from static frames and inputs a series of feature vectors to a custom Recurrent Neural Network (RNN) trained on our custom dataset of labeled plays. By leveraging transfer learning via automatic feature extraction and and training our own network for the multi-class classification task, we were able to achieve a final accuracy, after extensive hyperamater tuning and Data Cleaning, of **81.8%** in the test set.

C. Input Datasets

Pedro/Delete

1) Visual Data:

a) Data Description:

b) Data Acquisition:

c) Feature Engineering:

D. Technology Stack

Pedro/Deepa/Delete

E. Hyperparameter Tuning

Before exploring different models, we wanted to maximize the performance of the current architecture. To do so, we implemented a Grid Search algorithm for hyperparameter tuning. The specific parameters we included were the duration of the play clip, batch size, frames per second, epochs, activation function, loss function, and optimization algorithm. We tested all combinations of these hyperparameters on both the ResNet50 and VGG16 Convolutional Neural Network architectures, along with the images in either Grayscale or RGB. Hence, there were a total of 9 parameters that we tuned. We performed this extensive hyperparameter tuning process locally on the 800-play sample dataset, and later on the entire dataset once we obtained access to the PACE cluster of Georgia Tech’s Phoenix Supercomputer. On the 800 play sample dataset, we obtained a maximum accuracy of **[83.33, 91.26, 59.62, 64.29, 38.46]**, representing the accuracy of Kickoff, Run, Pass, Extra Point/Field Goal, and Punt plays respectively. The accuracy of a play type is defined as the correct number of plays classified as that play type divided by the total number of actual play types in the dataset. We decided to combine the Extra Point and Field Goal categories

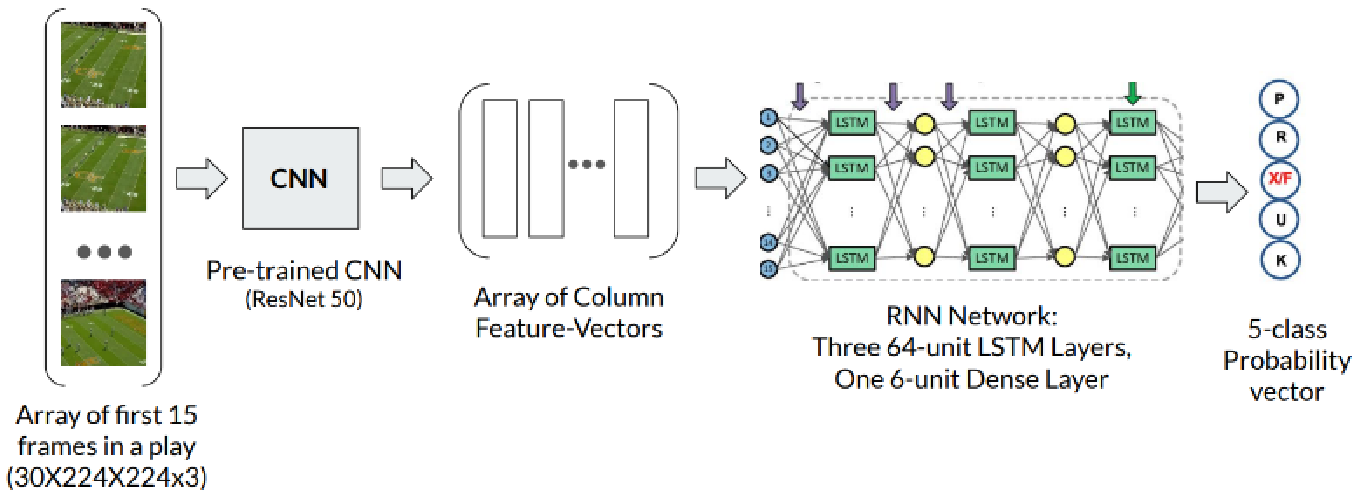

Fig. 1. Computer Vision Neural Network Model Flow Chart; [P = Pass, R = Run, X/F = Extra Point/Field Goal, U = Punt, K = Kickoff].
We consider a frame of the play clip for each of the first 15 seconds; Run the frames on the pre-trained ResNet50 architecture: 50 layer optimized convolutional neural network; Obtain output of feature vectors; Pass in feature vectors to the Long Short Term Memory Neural Network which outputs our final 5-class probability vector.

due to the similarity in play type and to mitigate small sample size issues. The total accuracy was **76.8%**. The optimal parameters were a FPS of 5, batch size of 32, clip duration of 15 secs, 8 epochs, Sigmoid activation, Adam optimizer, and Crossentropy loss function on the ResNet50 architecture with frames in RGB. In addition to tuning in search of optimal test accuracy results, we ensured that our training and test loss and accuracy functions were similar, avoiding overfitting. The middling results provided us with motivation and hope of better accuracies when utilizing the full dataset on a more powerful compute system. Performing the same Grid Search hyperparameter tuning on the complete dataset, the optimal parameters were a batch size of 16, 6 epochs, Sigmoid activation, Adam optimizer, 3 dropout layers of 0.1, and Crossentropy loss function on the ResNet50 architecture, leading to a maximum accuracy of **[82.84, 88.56, 69.9, 70.83, 52.48]**, representing the accuracy of Kickoff, Run, Pass, Extra Point/Field Goal, and Punt plays respectively, giving us a total accuracy of **79.65%**.

IV. Data Cleaning, Resampling, and Exploration

Further exploring ways to improve the accuracy of our work, we observed that the video files used to train and test our model are recorded on a personal camera or are obtained from screen-grabbed jumbotron footage, therefore subject to video recording errors. We hypothesized that certain errors, such as starting to record after the play had started or an on-screen overlay from the jumbotron, seriously affected model effectiveness and hindered further progression. To test our hypothesis, we aimed to produce a clean dataset. We started by identifying common recording errors present in the 5,647 play dataset used for training and testing. In addition to video recording errors, we also wanted to track difficult or unique play types that may highlight weaknesses of the model. Plays were divided amongst the team members, who then watched and labeled each and every single video with one of the following labels:

Normal	No notable errors
Early	The recording started too far before the snap
Late	The recording started too far after the snap
No Play	The ball was not snapped or extraneous footage like a crowd closeup
Duplicate	The recording is a copy of another video
Quarterback Scramble	An improvised run by the quarterback outside of the pocket
Sack	The quarterback was tackled behind the line of scrimmage without throwing a pass
Interception	A pass was caught by the opposing team
Lateral	A backwards pass
Fumble	A loss of possession during the play not due to an interception
Long Kick Return	A kick return longer than fifteen yards
Long Run	A run longer than fifteen yards
Fake Kick	A false punt or field goal attempt where the ball is run or thrown instead
Minor Graphic	A graphic covering part of the screen
Full-Screen Graphic	A graphic covering the majority of the screen
Miscellaneous	A recording error not covered by another label

We decided that plays with extreme errors, such as “No play”, or “Duplicate”, will indiscriminately be excluded from the dataset. On the other hand, minor errors like “Late” or “Minor graphic” were reviewed on a case-by-case basis: in some cases, the error was deemed not significant enough to affect how the play was classified. For the plays labeled ”Early”, we clipped the start of the play video to mitigate the issue. Once again, all the play clips that initially started early were evenly split amongst all team members. Each member locally fixed the recording and then reuploaded the play clip back to the dataset. Through our intensive, manual, and rigorous Data

Cleaning process, consisting of first finding errors and then immediately taking action on them, we successfully scrubbed our dataset of recording errors, reducing the size of the dataset down to 4722 plays.

TABLE I

Model Version	**Kickoff**	**Run**	**Pass**	**Extra Point/Field Goal**	**Punt**	**Total Accuracy**
Fundamental Model on 800 Play Sample with Hyperparameter Tuning	83.33	91.26	59.62	64.29	38.46	76.8
PACE before Data Cleaning with Hyperparameter Tuning	82.84	88.56	69.9	70.83	52.48	79.65
After Data Cleaning	85.96	86.85	70.33	80.00	59.79	81.00

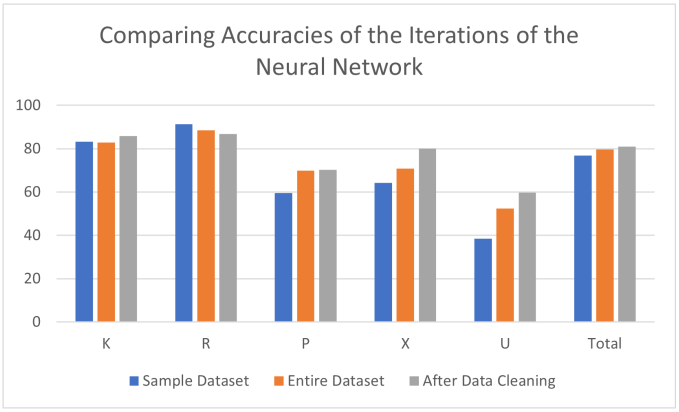

Fig. 2. Comparison between the accuracies of the fundamental model on the 800 play Sample dataset; model using the entire dataset; model after Data Cleaning.

Before the Data Cleaning process, our model had an average total accuracy of **79.65%** and a play-specific accuracy of **[82.84, 88.56, 69.9, 70.83, 52.48]**, again representing the accuracy of Kickoff, Run, Pass, Extra Point/Field Goal, and Punt plays respectively. For a fair comparison, we ran the model 20 times before and after Data Cleaning with optimally tuned hyperparameters. After completion of our rigorous Data Cleaning phase, our model improved to an average accuracy of **[85.96, 86.85, 70.33, 80.00, 59.79]**, which is a total accuracy of **81%**. We compare the iterations of our Neural Network in Table 1 and Figure 2. We observe that the accuracy of all play types, other than run plays, increases. While discussing the accuracy of run plays, we must mention that our dataset consists solely of Georgia Tech football plays, ranging from the 2010 to 2017 season. The coach during this entire range was Paul Johnson, who frequently executed the triple option play where three players are in a threat to run the ball. This led to Georgia Tech football having an extremely high frequency of run plays during his tenure. We hypothesized that this has led the model to overpredict plays as a run type. We can confirm this now as we see a reduction in the accuracy of the run type after cleaning our dataset. We see that the model was predicting plays to be run types even when it could not properly evaluate the clip due to it starting late, a full-screen graphic, etc. Hence, we will happily accept a slight decrease in the accuracy of run plays for an increase in the accuracy of all other play types, which shows that our model is not excessively predicting run plays and is rather learning better overall.

On the topic of having a run play-heavy dataset, here is the exact Georgia Tech play distribution vs that of other NCAA teams:

	Georgia Tech	**Other Teams**
Field Goal	1.5	1.98
Extra Point	5.44	3.58
Punt	4.29	5.74
Pass	23.28	38.65
Run	58.18	44.3

As mentioned earlier, we observe that Georgia Tech has a much higher percentage of run plays, making up for that imbalance with a lower percentage of pass plays. Prior to obtaining access to the PACE Phoenix cluster, when running the model on our local systems using the 800 play sample dataset, we realized that we should use a different sampling method to better reflect the play distribution of all teams as we aim for the model to be used across the country. Initially, we were randomly sampling 800 plays from the 5647 total plays. We changed our sampling distribution to the following:

1) Kickoff: 0.12
2) Punt: 0.12
3) Extra Point + Field Goal: 0.06 + 0.06 = 0.12
4) Pass: 0.3
5) Run: 0.34

In addition to balancing the distribution of run and pass plays, we increased the presence of rarer plays to give the model more opportunity to learn these plays.

Moving on, noticing that we, ourselves, often struggle to identify more unique and rare football plays, we were curious to see how the model performs against these more difficult play types. Hence, we identified these play types during our previously described labeling process and then examined how the model performs on the play types. Here are the overall model accuracies (correct play classifications/ total number of plays) specific to the play type after the Data Cleaning process:

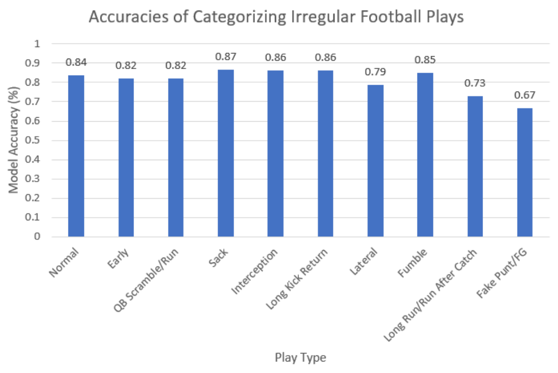

Fig. 3. Accuracies of Specific, Rare Play Types.

We notice that the accuracy of play types that once started early now has an accuracy extremely close to that of Normal play types, once again showing the effectiveness of clipping the ”early” plays. Moreover, the accuracy of the QB Run, Interception, Sack, Interception, Long Kick Return, and Fumble play types range from 82 to 86.4 percent, indicating the model performs well on these plays. The model performs worse on Fake Punt/FG (sample size of 6 total plays), Lateral (sample size of 47), and Long Run/Run After Catch play types. We hypothesize that increasing the number of Lateral and especially Fake Punt/FG play types in our dataset will increase model accuracy, one of our key next steps. In addition, by examining incorrectly classified Long Run and Run After Catch plays, we observed that these plays look extremely similar once the run starts to develop. If humans viewed one of these plays after a couple of seconds, they would be unable to tell if it was a Long Run or Run After Catch play. Hence, the start of these plays is extremely important in prediction and something we seeked to prioritize in future iterations of the model.

V. Binary Classification Model

In regards to future iterations of the model, we continued to tune the parameters of the multi-class Machine Learning classifier, continuously striving for higher accuracy. Simultaneously, we theorized that it would be helpful to task the model with a simpler task: identifying whether a play is Run or Not Run. Then, we would attempt to identify whether the plays that were classified as Not Run were either Pass or not. Continuing this process for 5 levels (corresponding to the 5 play classifications), we created a Binary Tree classifier as shown below in Figure 3.

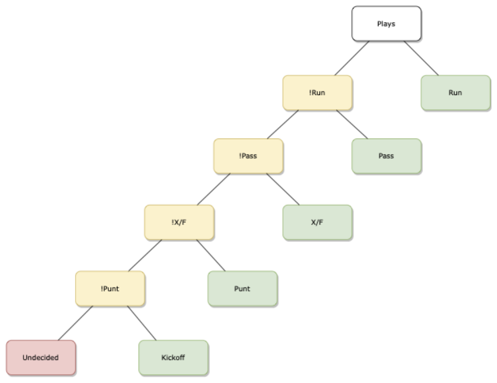

Fig. 4. Binary Classification Model Structure

We hypothesized that the binary tree classification model can be more effective than a multi-class classifier in certain situations. Generally speaking, the primary advantage of a binary tree classification model is that it can more accurately classify complex and hierarchical data by breaking down the classification task into a series of more straightforward binary decisions. Our model is a multi-level binary classifier where at each level of the tree, it classifies a play into either one of the five play types or not that play type. As mentioned, the first level of the tree classifies a given play into either a run play or not a run play (run, !run). Then the plays classified as not run would propagate to the next level, pass or not pass. Every play makes its way down the tree until it is classified into a play type or reaches the end of the tree, where we deem it as unclassified. This model had a welcoming self-cleaning property to it because many of the plays that were unclassified in the tree were plays that should be removed from the dataset due to one of the video errors described in the Data Cleaning section. Although this may seem more computationally expensive as a collective due to running the model multiple times as you proceed down the tree, the opposite is true as the play is classified into one of two categories which is much easier than having the model choose between five classes at once. Additionally, the number of plays in the dataset exponentially decreases as you proceed down the tree, making the cost of running the model computationally insignificant in the lower levels of the tree. In fact, the lack of plays, as you proceeded down the tree, posed a major issue. By the time we reached the Kickoff level, the last level of the tree, there were under 100 plays remaining in the dataset. This was clearly not sufficient to properly train the model. To get around this issue, we innovatively trained and tested the model at each level using all of the available plays. To elaborate, we trained the LSTM (Long Short-term Memory Network) models that classify a play as Run/Not Run; Pass/Not Pass; Kickoff/Not Kickoff; Extra Point/Field Goal or Not; and Punt/Not Punt each using the entire dataset. This mediated the issue of our dataset size decreasing as we went down the Binary Tree. It is essential to observe that each of these models is independent of the other so there is no issue with using the same training and test set for them. They are only connected in the way that they would be used in an application. The overall accuracy of the best iteration of this model was **83.36%**. Similar to the main multi-class classification model, we tuned our parameters through a Grid Search. The hyperparameters resulting in the best accuracy included 16 epochs, a batch size of 64, Adam optimizer, and the Binary Cross entropy loss function.

VI. Markov Chain

Having NCAA XML files containing play-by-play data for each of the 277 football games played by Georgia Tech from 2000 to 2021, we have automated parsing of these files to extract contextual and statistical data for each of the 56,596 plays that occurred. To supplement our findings from the Neural Network architecture, we have implemented a Markov chain that contains the probabilities associated with all possible state transitions that could occur in a football game. The main characteristics that comprise each state are:

1) Down (4): 1, 2, 3, 4

2) Play type (7): run (R), pass (P), PAT (X), field goal (F), punt (U), kickoff (K), other (O)

3) Field Position (10): 100-yard field divided into 10 10-yd buckets on the home side: [0,9], [10,19] . . . [40,50] and opposition side: [50,40], [39,30] . . . [9,0]

4) Distance to first down (6): [1,3], [4,6], [7,9], [10,14], [15,19], [20,99]

We have a different number of options for each state: Down (4), Type (7), Field Position (10), Distance to 1st (6); this gives us 4 * 7 * 10 * 6 = 1680 distinct possible states that are grouped in the order of characteristics. Thus, we have a 1680 x 1680 square matrix with state transition probabilities recorded at each entry; the matrix is built in row-major order, meaning that an entry M[i, j] = P(i $\rightarrow$ j) and the sum of each row will be 0 or 1. Some rows will have a cumulative probability of 0 because certain play transitions, such as a kickoff on 2nd down, are impossible. Other play transitions that are unlikely, such as a field goal or punt not on a 4th down, will generally have lower probabilities as target states. In addition to the complete Down-Type-FieldPosition-DistanceToGo 1680x1680 matrix, we also built smaller Down, Down-Type, and Down-Type-FieldPosition matrices. A simple example of the down matrix, which is a simple 4x4 matrix is displayed below with labels/indices:

$Down$	1	2	3	4	$Total$
1	0.403	0.597	0.000	0.000	1
2	0.375	0.005	0.619	0.000	1
3	0.454	0.000	0.003	0.543	1
4	1.000	0.000	0.000	0.000	1

From this matrix, we then added the Type, Field Position, and Distance to the first down states grouped in this order to the respective matrices. Alongside the neural network model, the matrix is then used to influence the final decision by providing calculated probabilities of the most likely next state/play type. Although not developed as a standalone model, but for reference, the accuracy of solely the Markov Model in predicting the next play is **[94.05, 69.35, 19.49, 77.01, 56.34]**, which is a total accuracy of **55.47%**. We observe that the Markov Model performs spectacularly in regards to predicting Kick plays, a quality we hope to use to increase the accuracy of Kick plays in our general model.

VII. Model Fusion

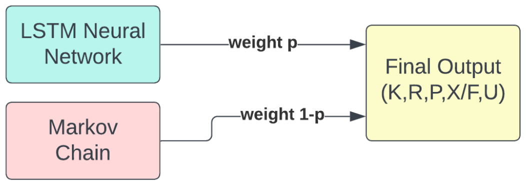

Fig. 5. Model Fusion Approach 1

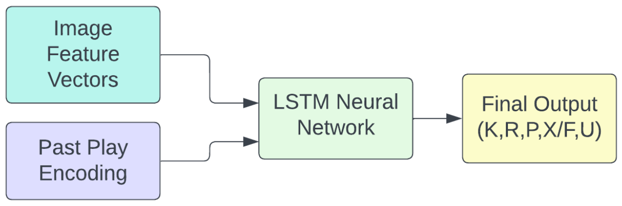

Fig. 6. Model Fusion Approach 2

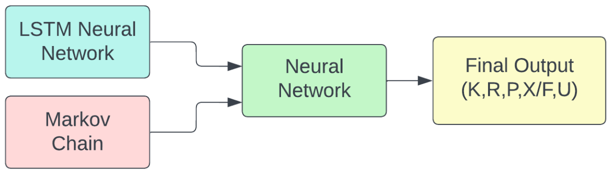

Fig. 7. Model Fusion Approach 3

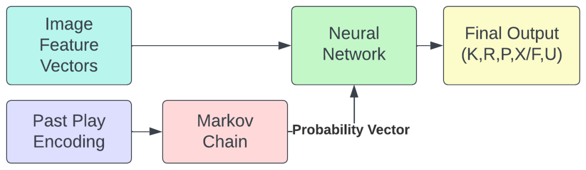

Fig. 8. Model Fusion Approach 4

We explored several strategies for combining the Matrix and our Neural Network model. First, we performed a weighting of the results from the respective models. Regardless of the exact matrix we are using, we condensed the columns of the matrix into 5: representing the probability of the next play being a Run, Pass, Punt, Kickoff, or Extra Point/Field Goal. For example, the 1680x1680 full matrix is converted into a 1680x5 matrix. Hence, given the necessary details about the previous play, we can obtain a normalized 5-element prediction array from the matrix, representing the probabilities of the current play. The Neural Network model provides us with this same normalized 5-element prediction array given a clip of the current play. Hence, we can weight the results. If the maximum value of both prediction arrays represents the same play type, there is a consensus on the prediction, making it the clear final choice. Otherwise, in the case of disagreement, we added the arrays together, tuning the specific weights of the linear combination. We discovered that p=0.6 was the optimal weight for the Neural Network model, giving a weight of 0.4 to the matrix output. We also explored accepting the output of either model if it was extremely confident, tuning the confidence threshold, in its prediction before combining the models. As an example, if the Markov model predicted a run play occurring at 0.9 chance while the Neural Network model assigned a 0.6 chance to a pass play, we would have run play as the final decision, given our confidence threshold was less than 0.9. We found the optimal threshold to be 0.8. We also explored weighting based on the individual play types rather than on the entire vector outputs. We independently followed this weighting strategy on the unique matrices, obtaining the following results. The approach is also visualized in Figure 5.

Down-Type results	
K Accuracy	88.10
R Accuracy	79.70
P Accuracy	70.07
X/F Accuracy	71.43
U Accuracy	76.84

Down-Type-FieldPosition results	
K Accuracy	75.00
R Accuracy	91.26
P Accuracy	61.02
X/F Accuracy	73.00
U Accuracy	57.89

Down-Type-FieldPosition-DistanceToGo results	
K Accuracy	75.00
R Accuracy	91.26
P Accuracy	62.41
X/F Accuracy	74.60
U Accuracy	57.89

Likely due to a decrease in the sample size of each entry in the larger matrices, the simpler matrix, only consisting of the Down and Type of the previous play, led to the best results.

Next, as shown in Figure 6, we explored adding previous play information to the neural network input itself. As noted in the pipeline diagram (Figure 1), an array of column feature vectors is passed to the LSTM network. Specifically, the input is a 30x2048 array, representing 2048 pixels in each of the 30 frames. Representing the previous play information, specifically the past down and type, as a 30 element vector, we were able to append a column to the input. The input was now 30x2049. The accuracy of this approach also bettered the prediction accuracies of the modified neural network: **[82.14, 88.31, 76.33, 85.71, 69.47]**, which is a total accuracy of **83.1%**.

However, we note that the input to the architecture still heavily contains information regarding the clip, only slightly representing past play information (2048 vs 1 column). Furthermore, we hypothesized that introducing past play information in vectorized form may confuse the model.

Hence, we moved on to our next approach to fusion: appending a final, new Neural Network that takes in the results of the LSTM model and the results of the matrix. We built a simple Sequential Network consisting of multiple Dense layers. The input is a 10-element vector, consisting of the 5-class probability array outputs from the Neural Network and the Markov Model appended to each other. The goal of this final Neural Network is to essentially find the best weighting of the two outputs, much better than we could through tuning weighting parameters, as we discussed previously in our first approach to Model Fusion. First, we built the neural network only using input where the Markov Model and the LSTM model disagreed on their prediction as the final choice is obvious when the models are in agreement as to what the play type is. This gave us a sample size of 562 plays and a maximum accuracy of 82.03 percent upon tuning the Neural Network (experimenting with the number and inputs of each of the Dense Layers along with general hyperparameter tuning). We realized that the sample size was already small as we could only test this final Neural Network with plays that were in the test set of the initial LSTM Neural Network model. This was because including plays in the training set of the LSTM Network would provide an unfair advantage to the LSTM model, possibly favoring it over the Markov Model. To combat the sample size problem, we decided to include all the plays from the test set, including the ones upon which both the models agreed on. This slightly helped our sample size problem, increasing the sample size to 1192 and accuracy to 84.05 percent. This approach is visualized in Figure 7.

However, we still seeked to use the entirety of the dataset. Hence, our final approach to Model Fusion was building a singular Neural Network incorporating all of the play clips and corresponding past play information. Rather than passing in encodings of the past play as we did in Approach 2, the input was the probability vector obtained from the Markov Model. This input was only considered in the middle of the Neural Network after the play clip information had been condensed to a 5-element probability vector, the output of the initial neural network. In other words, the network had 2 inputs: the feature vectors from the play clip and the probability vector output from the Markov Model. The latter information was only introduced in the later layers of the network, as depicted in Figure 8. This solved the imbalance of information problem between the video clip and the Markov Model discussed in Model Fusion Approach 2. This approach allowed us to use the entirety of the dataset and incorporate Markov information in a singular model, making the tuning process more defined. This final approach led to the following results: **[89.29, 89.65, 83.42, 87.36, 83.10]**, which results in an outstanding total accuracy of 87.26%, an over 11 percent increase from where we started. Below lies a comparison of the different Model Fusion Approaches.

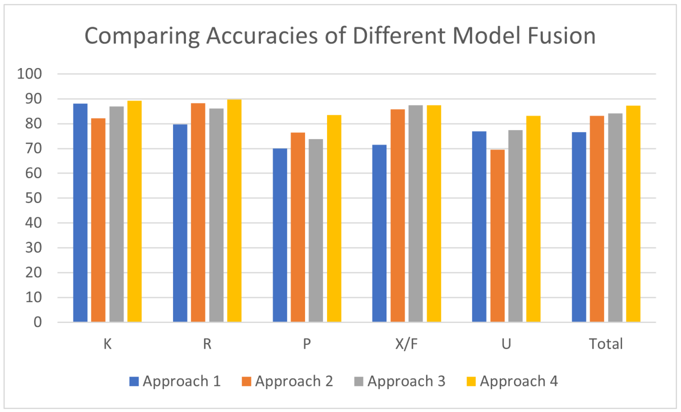

Fig. 9. Comparison of the Model Fusion (MF) Approaches: Manual Weighting, Adding past play encoding to Neural Network Input, Appending a final Neural Network, Creating a singular Neural Network that trains on both the Markov output and feature vectors using the entire dataset.

TABLE II

	Kickoff	Run	Pass	XP/FG	Punt	Total Accuracy
MF Approach 1	88.1	79.7	70.07	71.43	76.84	76.6
MF Approach 2	82.14	88.31	76.33	85.71	69.47	83.1
MF Approach 3	86.9	86.16	73.78	87.36	77.46	84.05
MF Approach 4	89.29	89.65	83.42	87.36	83.1	87.26

VIII. Computing Environment

Our model is trained and evaluated for performance on the Partnership for an Advanced Computing Environment (PACE) [7] Instructional Cluster Environment (ICE) at Georgia Tech. We have used the The Open OnDemand interface for Jupyter notebooks which allows you to access PACE clusters through a web browser. On this cluster, our model is trained and evaluated using a 24-core node hosting Dual Intel Xeon Gold 6226 CPUs @ 2.7 GHz, 192GB RAM and the Nvidia Tesla V-100 GPU. The environment has allowed multiple contributors to collaborate on a shared location. Our software stack consists of locally built anaconda3 python 3.9 environments with various packages including OpenCV, FFmpeg, pandas, TensorFlow, Keras, scikitlearn, and CUDA Toolkit installed in it.

IX. Performance Evaluation

Here is the overall accuracy and loss function of the Neural Network corresponding to our final work: Model Fusion Approach 4. These graphs highlight the similarity in results of the train and test set, showing the successful avoidance of overfitting.

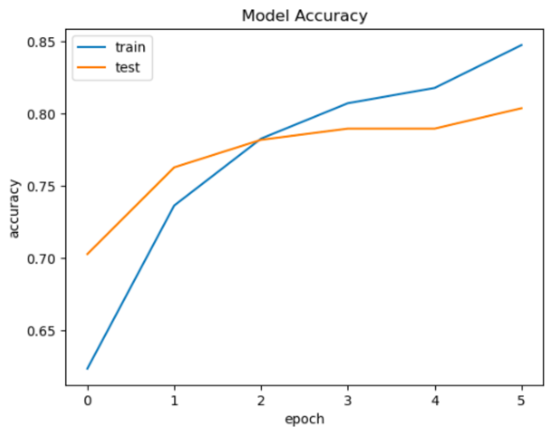

Fig. 10. Accuracy Function of the Model through first 6 epochs.

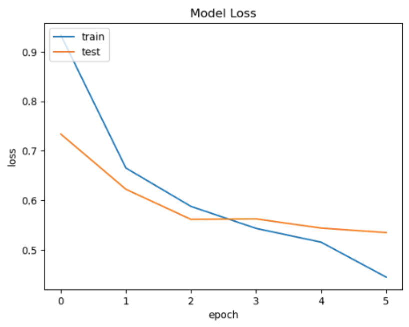

Fig. 11. Loss Function of the Model through first 6 epochs.

Here is the play classification accuracy chart of the final Model Fusion Approach: the singular Neural Network trained on the entire dataset using past play information and play video feature vectors.

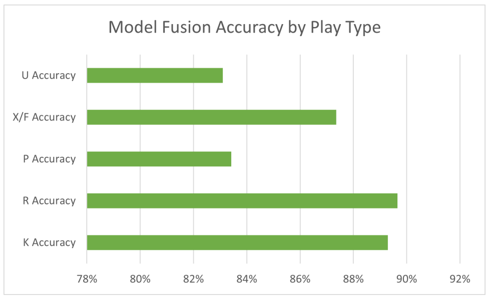

Fig. 12. Accuracy of each play type of the final model: Model Fusion Approach 4

Here is a comparison of all of our models:

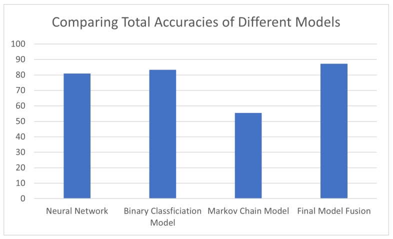

Fig. 13. Comparison of the best iteration of the Neural Network, Binary Classification Model, Markov Chain, and our best attempt at Model Fusion.

Acknowledgment

TODO by Dr. Coyle and Deepa

Conclusion

summary of paper; todo by Deepa + Dr. Coyle! likely short as paper is already at 8 pages!

References

[1] M. Ötting, "Predicting play calls in the national football league using hidden markov models," *IMA Journal of Management Mathematics*, vol. 32, no. 4, pp. 535–545, 2021.

[2] S. Chen, Z. Feng, Q. Lu, B. Mahasseni, T. Fiez, A. Fern, and S. Todorovic, "Play type recognition in real-world football video," in *IEEE Winter Conference on Applications of Computer Vision*, pp. 652–659, 2014.

[3] W. Li, W. Nie, and Y. Su, "Human action recognition based on selected spatio-temporal features via bidirectional lstm," *IEEE Access*, vol. 6, pp. 44211–44220, 2018.

[4] B. Siddiquie, Y. Yacoob, and L. Davis, "Recognizing plays in american football videos," 01 2009.

[5] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding* (A. A. Salah and B. Lepri, eds.), (Berlin, Heidelberg), pp. 29–39, Springer Berlin Heidelberg, 2011.

[6] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, 2015.

[7] PACE, *Partnership for an Advanced Computing Environment (PACE)*, 2017.