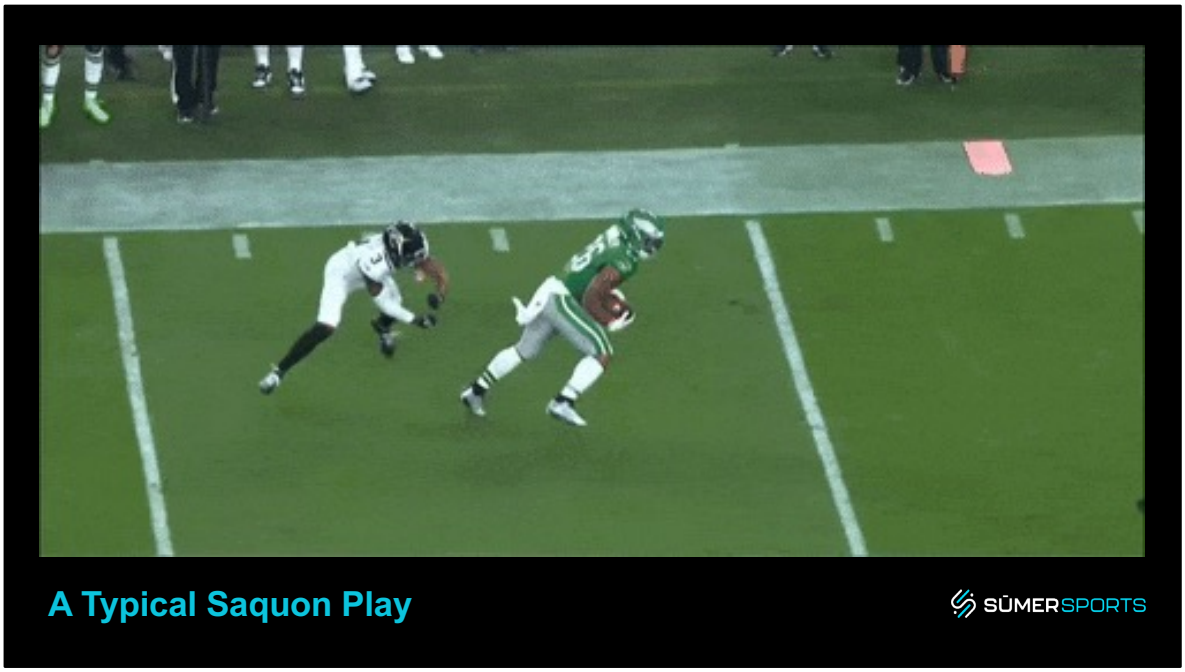

SŪMERSPORTS
Transforming Deep Learning in Football
Udit Ranasaria, Vishakh Sandwar, Pavel Vabishchevich
CMSAC Football Analytics Workshop, Oct 2025



A Typical Saquon Play

SÜMERSPORTS

- Everyone has seen this play, and it's a great reminder to the year of Saquon.
- Everyone is focused on Saquon in this play, but how can we analyze and uncover what else was happening on this play?

- In 2022, Udit and Pavel started started at SumerSports to basically "mess around with tracking data and build cool things", and it led down this beautiful rabbit hole of tracking data, where Vishakh (and Smit Bajaj) won BDB off their work last year.
- SumerSports hired me in June and the Eagles hired Smit.
- Today we want to share the early parts of that technical journey with you, and give you some of the foundational thinking that can help you succeed

Previous Deep Learning in Football

SÜMERSPORTS

Tracking Data Everywhere

- 2016: Sensor 2D Tracking
  - RFID chips in pads
  - Zebra + NGS
- 2020: CV 2D Tracking
  - Generated from All-22 Film
  - SportLogiq / Telemetry
  - Many others now!
- 2026? Optical 3D Tracking
  - Pose estimation
  - Hawk-Eye 👀👀👀👀👀

SÜMERSPORTS

- NFL Analytics has had access to rich high volume tracking data since 2018
- Expansion of Computer Vision added access to College and some high school
- With HawkEye being deployed across NFL stadiums, we should have an eye to the future!
- This presentation will be scoped to NFL Sensor-based 2D tracking data
  - The kind released in the Big Data Bowl every year

2D Tracking Data Form

- Each frame represents a moment in time
- Spatial features for each player
- Bonus: Events, ball info, etc.
- For today, scope to modeling assuming each frame represents an independent data sample

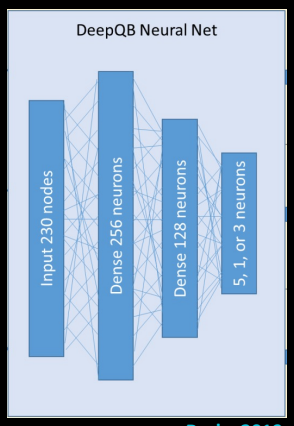
FrameID	Event	Player ID	Team	x	y	Has Ball
15	Handoff	80932	OFF	26.9	27.9	0
15	Handoff	87432	OFF	35.2	34.0	1
15	Handoff	09323	OFF	40.4	25.4	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	Handoff	78152	DEF	35.3	31.4	0

SUMERSPORTS

- For our paper and this presentation, we focus on modeling across players
- So we treat each frame of tracking (i.e. a moment in time) as a unique training sample, even if they come from the same play.
- We hope to see more public work on modeling across the time dimension from our work here 😊

Getting Started: Fully-Connected Feedforward Networks (FFN)

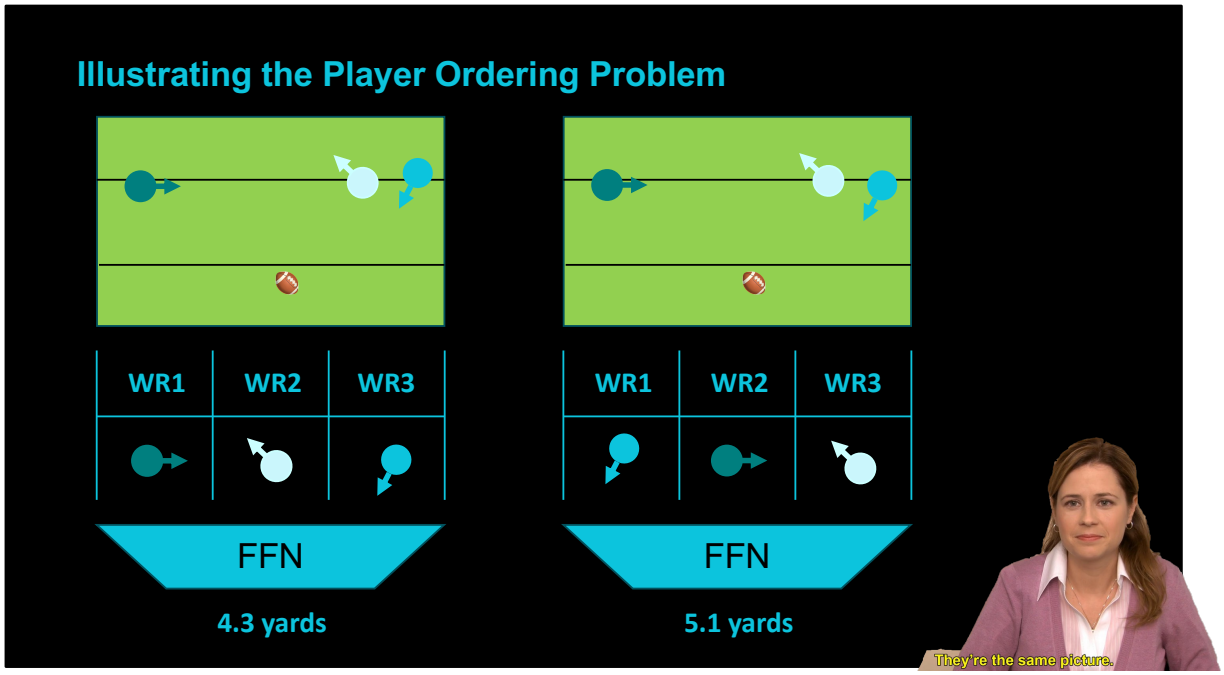
- Also known as Multilayer Perceptron (MLP)
- The original NN designed for tabular data
- Alternative to other black-box tabular approaches like XGBoost
- Notable Work:
  - DeepQB (Burke 2019)
  - Going Deep (Yurko et al. 2019)
- Limitations:
  - Have to collapse frame into wide-form table of features
  - Forces a heuristic for ordering players
  - Similar tracking data can look very different to model



Burke 2019

SÜMERSPORTS

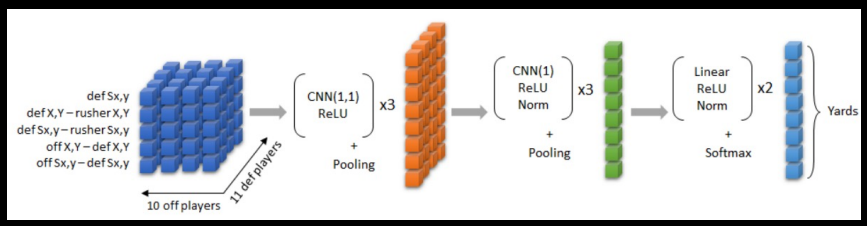
- Feedforward Neural Networks, also known as Multilayer Perceptrons, were the original neural net
- You feed in ordered features and have learnable weights + non linearities between every layer
- Extremely expressive black box for large data problems, but largely functioned the same as XGBoost in a broader sense
  - Still have to construct a single tabular dataframe
- This approach does allow the model to interact model features freely to find patterns (like xgboost, but unlike simple linear modeling)
- Cons:
  - A tracking data frame does not come in the form of "1 row per frame with an ordered set of features for that frame"
  - You have to pivot the player rows into columnar features. This requires hand-crafted feature engineering but also usually enforces an ordering on the players
  - Previous work tend to use a heuristic to decide the ordering for the players.
    - Yurko: Distance from ball carrier
    - Burke: Vertical Depth of WR



Lets look at a toy example to show how player ordering can affect FFNs or XGBoost models

- Illustrated is a moment from a toy pass concept where the frontside (right) has the inside receiver running a post with a curl on the outside. On the backside, we have a intermediate dig route.
- The left picture and right picture are BOTH this same situation, with only slight differences. We would expect a model to treat these as very similar situations from the tracking data
- But when we featurize this frame into a wide-format table (e.g. using Burke's heuristic of depth of WR) we have to decide which of these will be "the WR1", "the WR2", so on.
- Similar tracking data can have very different order of features, which can cause discrete changes in model output!
- We want a modeling architecture that is *invariant* to player order, because there is no obvious or consistent order in most sports problems!

The Zoo Architecture, a Big Leap Forward

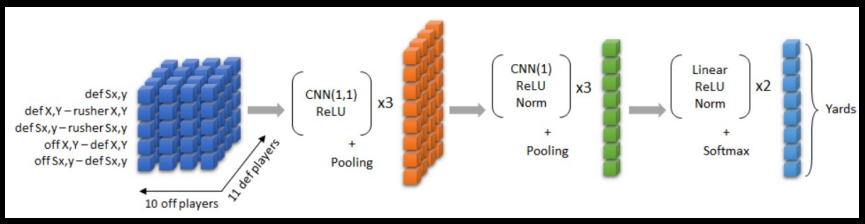


- Singer and Gordeev's legendary win in 2020 BDB
- Identified importance of player-order invariance
- Relied on pooling as an operation that was invariant to player order
  - Pooling is a fancy term for "take the mean or max over" a set of vectors

SUMERSPORTS

- Many have likely heard about the Zoo before: Didn't know much about football as Austrians, but won the prediction competition in 2020
- 2020 BDB were asked to predict **rushing yards from the handoff frame**
- They beat the second place competition handedly
- Relied on *average* and *max* pooling to solve the player-order invariance problem
  - This was progress, but pooling achieves invariance by collapsing a dimension
  - Once you pool, you lose the rich information. Can't go back!
    - E.g. Once you take the average of a set of numbers… you can't use that to infer the variance of the original set if that's needed for later learning
  - Consequence: While each pooling layer does operate across the player dimension, one you do it, all downstream layers lose access to the rich data in the player dimension
- Average and Max Pooling are also a *non-parametric* operation, there is no ability for the network to learn optimal weights while doing the pooling
- Ideally, we would have a *order invariant* Deep Learning operation that does NOT collapse the player dimension and has learnable weights parametrizing it

The Zoo Architecture: Convolutions??



- Generated a cube of interaction vectors between pairwise players
- Applied a 1×1 "convolution" over each interaction vector independently
  - 1×1 convolution is algebraically identical to FFN applied across just one dimension
  - True convolutions would have violated player-order invariance!
- Pool across interaction vectors afterwards
- Limitations:
  - Focused learning to only between 2 opposing players at a time
  - Pooling collapses player dimensions
  - Features are highly specific to rushing targets

SÜMERSPORTS

- Many people think the Zoo was "Computer Vision" model because they used Convolutional Neural Nets in their code
- The Zoo's innovation was *not* using Convolution Neural Nets
  - CNN(1,1) is exactly equivalent to FFN across one of the dimensions of the input (the other dimensions get treated as independent)
  - They used a "1x1 convolution" which is exactly equivalent to a FFN (across a specific dimension) with *no* convolving
  - Convolving across players using a partial-width kernel would violate player-order invariance!
- The Zoo's primary innovation was feature engineering a cube of "interaction" vectors between every player pairwise
  - This turned each frame of tracking data from 1 training sample into 110 training samples for the first FFN with a relatively small set of ordered features
  - Powerful regularization idea (focus on 2 players at time, instead of everyone) given the small training set released for BDB 2020!
- The Zoo was very big leap for creative NN usage, but it's biggest innovation was letting the model focus on 2 players at a time initially before pooling

across all.

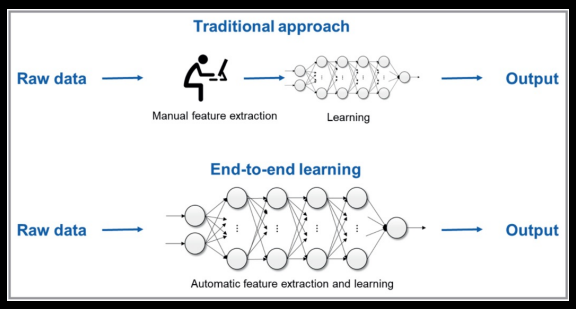
The Transformer Revolution

SÜMERSPORTS

- Quick detour to discuss the Transformer and Deep Learning outside of sports

End-to-End Learning

- Early 2000's: Reliance on hand-crafted features
- 📈 Data Volume
- 📉 Compute Cost
- 📈 Architecture Innovation
- Mid 2010s: Rise of end-to-end learning (AlexNet!)
  - Adapt your architecture to your raw data inputs and outputs.
  - Model will learn salient features in the network given enough data
- For most domains, matter of when not if

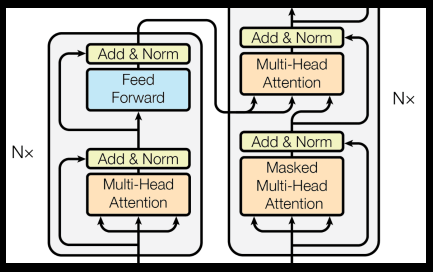


SÜMERSPORTS

- 2000s, ML and DL relied on hand-crafted features
  - There are so much complex science, physics and math that went into it with fancy names
- Data quantity exploded, compute costs shrunk, and people kept building new DL architectures
- **end-to-end learning**
  - Adapt your architecture to your raw data inputs and outputs.
  - With enough data, the architecture will learn the most salient features in latent space
  - AlexNet was pivotal in 2012 and kicked ass
- By mid-2010s, this was the standard in big data spaces for non-tabular data!
- CV, Speech Processing, Machine Translation, NLP, and Protein Folding have all fully transitioned away from complex feature engineering and multi-part modeling to E2E
- We are currently (in 2025-26) seeing this change happen in Robotics and Self-Driving...

Attention is All You Need

- Seminal Google paper published in June 2017
- Introduced the “Transformer”
  - The “T” in ChatGPT
  - Major reason for the LLM explosion in 2022
  - Key Innovation: Heavy use of “Self Attention”
  - The most used architecture in most AI domains
- Cited 200k+ times (and counting)
- Title inspired by “All you need is Love” by the Beatles
  - Shout out to Claude for this fun fact



SÜMERSPORTS

Why Were Transformers So Successful?

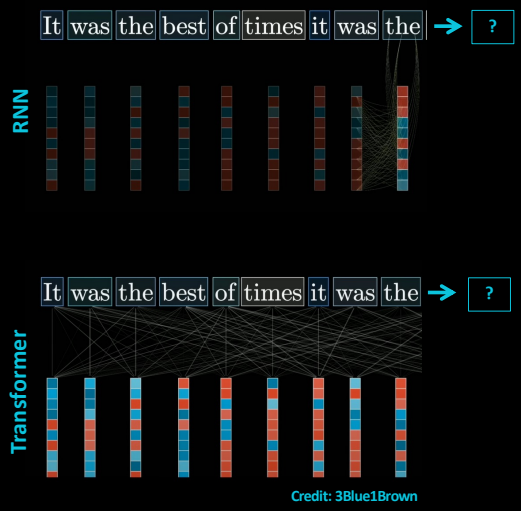
- Most data can be seen as a sequence of items
  - Each item has a corresponding vector of data

CNNs & RNNs: Order as a heuristic

- Designed to leverage relevant locality, order and adjacency
- Signal between items far apart can get muffled

Transformers: Unordered Bag of Items

- Self-Attention: Direct connections between all pairwise items equally, irrespective of order
- If needed, ordering features can be passed into each item's vector
- Model can learn importance of locality
  - End-to-end learning!
- Bonus: Much more parallel in computation which enabled model scaling for huge LLMs



- NNs can be thought about in terms of how information flows across their neural pathways
- For all these problems, we represent data as a sequence of items with each item having a vector of data that the model tracks as information about that item
  - Items can be words for NLP, pixels/patches for images, etc.
- CNNs and RNNs were designed with the idea that we should optimize neural pathways based on locality
  - Convolution in Images: Learn filters that operate on the Local neighborhood of pixels
  - RNNs in NLP: Process words left to right and each word adds in information.
    - The information pathaway from the first word and last word has to survive many intermediate updates!
- This made a lot of sense at the time.
  - But, it was still a heuristic… and violates E2E principals to some degree
  - The title of an essay is very important when predicting its concluding paragraph.
  - We should let a model learn the importance of locality, not bake that

into the model arch

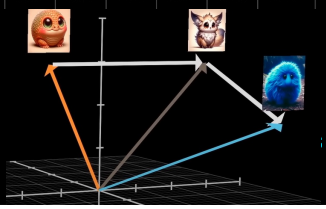
- Transformers treat information pathways as an unordered bag of items.
    - Each Pairwise item gets the same information pathway
    - The model can learn, from data, how to prioritize the importance of positioning or ordinality

Self Attention in Language

- Pairwise Matrix
- Use item vectors to learn attention weights between pairwise items
- Apply a weighted sum to update each items current vector
- Get an updated output vector for each item after it has been "attended" to by all other items

	a	fluffy	blue	creature	roamed	the	verdant	forest
a	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fluffy	0.00	1.00	0.00	0.42	0.00	0.00	0.00	0.00
blue	0.00	0.00	1.00	0.58	0.00	0.00	0.00	0.00
creature	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
roamed	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
the	0.00	0.00	0.00	0.00	0.99	1.00	0.00	0.00
verdant	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
forest	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Credit: 3Blue1Brown



- We could spend 3 hours going into the details about attention, but here is a simplified visual explanation in the Language domain (from 3Blue1Brown)
- The input to the transformer is a sequence of items where at each item index we have a vector representing data about that item
- Attention constructs a matrix comparing each item against each other item
  - This is why it is called "self" attention! It isn't using outside data to update it's the input sequence, it is updating the input sequence using data from the input sequence itself
- The matrix fills in each grid item with a value of "how heavily should the left item update the top item".
  - This value is a normalized by column so for each top item, we get a weight that sums to 1
  - This value is called the "attention weight".
- So in this case, the model could have learned that for a noun such as "creature", the only important updating words tend to be adjectives that ordinally fall right before the noun.
  - It calculates attention weights of 0.42 and 0.58 for those 2 adjectives for their update on "creature"
  - This results in the original creature vector (orange vector) being

updated with (0.42 * “fluffy” in) + (0.58 * “blue” info vector) = updated creature vector (blue vector)

- This process will also happen simultaneously and in parallel for all the items, so noun “forest” will be updated by (or attended to) adjective “verdant”
- The output is still vectors for each item in the exact same order as input … but now just with updated data from self-attention!

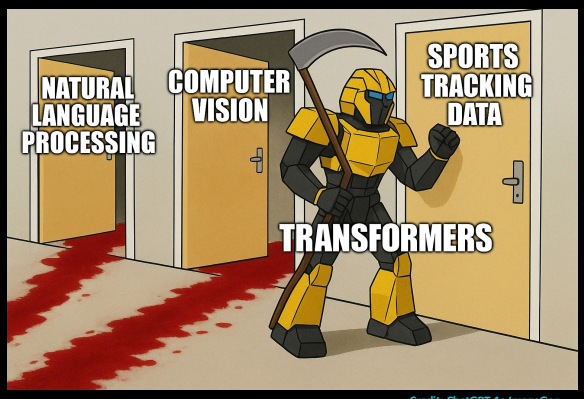
Transforming Deep Learning in Football

SÜMERSPORTS

Recap of Goals

- We want a solution that fits:
  - Player Order Invariance
  - End-to-End Learning
  - General across sports problems
  - Learns well across players

Transformers and Attention!!!



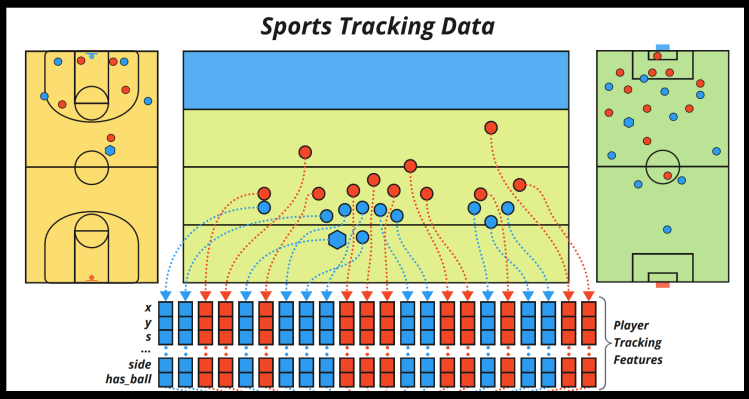
Credit: ChatGPT 4o ImageGen

SÜMERSPORTS

- We will go through how we would put sports tracking data into a transformer and how it achieves the goals!

SportsTrackingTransformer: Vector Creation

- Each player is an item in the “unordered bag”
- Spatial features go into respective item starting vector
- Any player ordering!
- No feature engineering needed
- Same for all sports or targets

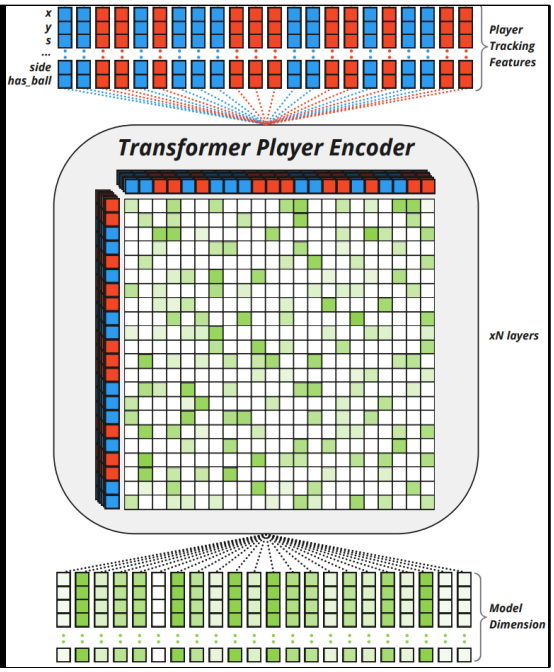


SÜMERSPORTS

- Each player’s spatial features becomes the initial vector for an item in the transformer sequence
  - Ordering of players doesn’t matter!
  - Just organizing the raw tracking data (without converting its shape) for the transformer
- No feature engineering, we learn directly off of these end-to-end

SportsTrackingTransformer: Stacking Transformer Layers

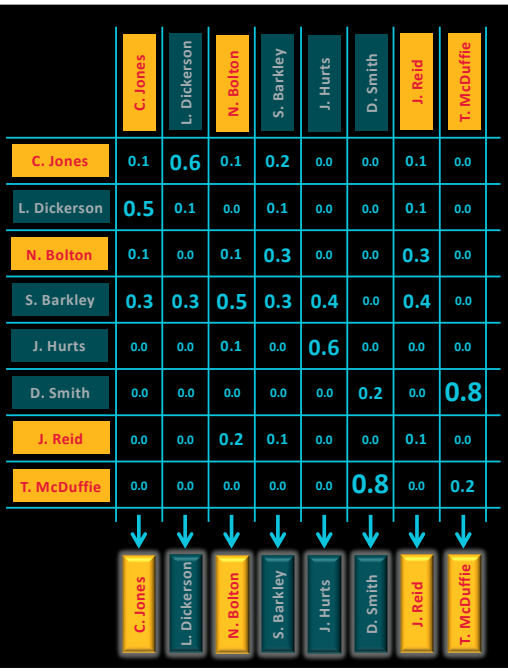
- Input and output of Transformer is the same data shape
- Chain these Transformer layers together
- Each layer gets maximal access to all other player vectors
- No collapsing like with pooling!
- Final output is a set of player vectors with salient values based on training objective



- Now that we have a sequence of items for each player and a vector with data for each player, we pass that directly into the transformer
- Transformers and Attention just update each player's vector with information from other players, without changing the order or shape of the data
- This lets us chain transformer layers back to back as much as we need.
- Every layer will learn across all players repeatedly and in a parametrized way
    - Remember pooling only operated across player dimension once and collapsed the player dimension to do so!

SportsTrackingTransformer: Self-Attention

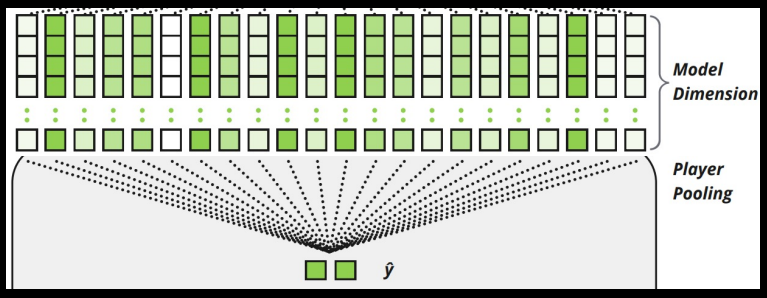
- Use each player vector to figure out attention weights between players
  - Columns are normalized to 1
  - Rows can give clues about "importance" of that player
- Attention weights are used to update the original vectors
- Re-ordering players simply re-orders the outputs
  - Each player's updated vector values are unchanged



- Lets take a look inside what an attention matrix might look like inside these Transformer layers for a run situation where the model is predicting tackle location
- Hypothetical run play handoff to Saquon with the Chiefs on defense.
  - Lets imagine Devonta Smith and Trent McDuffie are matched up in press detached from the formation… and Devonta runs a decoy route to keep McDuffie out of the run fit.
- Remember, attention takes in a sequence of item vectors and outputs updated vectors in the same order as the inputs
- Inside the matrix, the model is learning how to assign attention weights on which player on the left needs to update the vector for the player on the top
- Observations:
  - Columns are normalized to add up to sum to 1
  - Players who are matched up are likely to attend to each other
  - Teammates can and often attend to each other
  - Row wise attention sums can give clues as to which player is updating the most other playes (high importance)
    - The model has learned that on a run play, the RB has important information to update many of the other involved players on the

run play
    - Smith and McDuffie both attend to each other, but no one else on the play because they are largely uninvolved in the run
- This is a toy example to highlight how attention *might* work in a football problem
    - In practice, interpreting attention seldom comes out this nicely

SportsTrackingTransformer: Getting Outputs



- Need to get prediction from final player vectors
- Last Step: Pool across players to finally collapse that dimension
- Tiny FFN to convert to data shape of target variable
  - Slight modifications for regression vs classification problems

SUMERSPORTS

- Up till this point, the architecture has been the same for basically all sports and target variables
- After applying many Transformer layers, if we want a single prediction across all players then we do need to collapse the player dimension
- We apply our familiar pooling + FFN strategy to consolidate all this information into our final predictions
- Note: The vast majority of the network operates with the player dimension active, a big difference from the Zoo!
  - This lets the model learn weights to create very rich latent-space representations of the players, before having to collapse for the final output

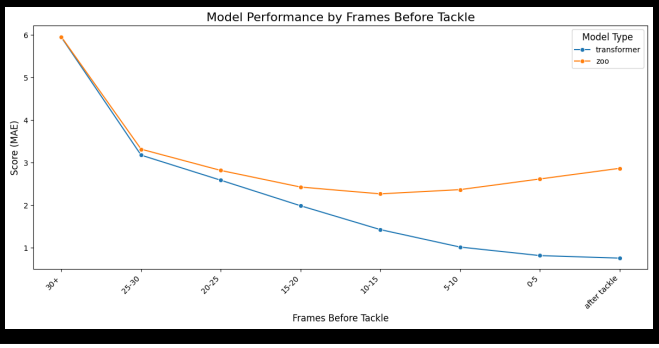
Ok, but that's all theory…

- Bakeoff to predict tackle location
  - Close cousin to the objective of rushing yards in 2020
- Greatly expanded the dataset
  - 2024 BDB
  - Rushing and Run after Catch
  - All frames, not just handoff
- Performance
  - Very similar at ball snap and handoff
  - Transformer generalizes much better as play progresses

Split	%	Plays	Frames
Train	70	8.7k	740k
Val	15	1.9k	160k
Test	15	1.9k	160k

Test Set: Mean Squared Error (Lower is Better)

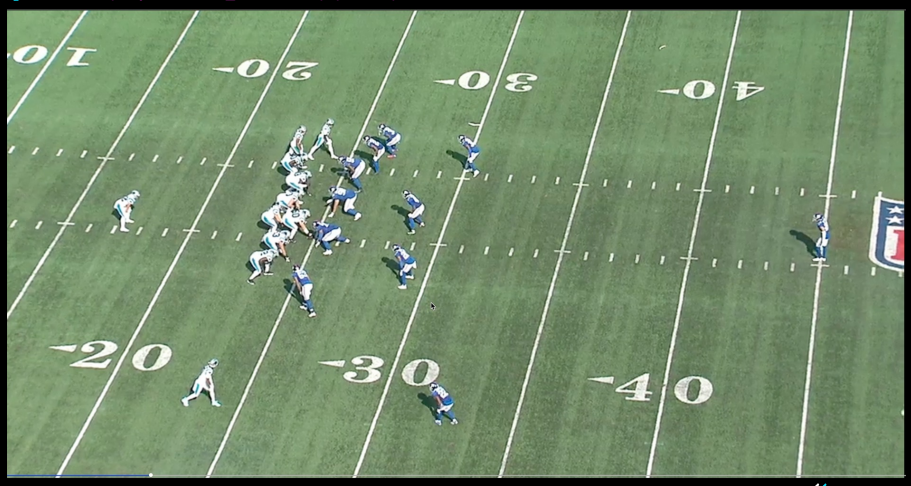
Event	n	Trans-former	Zoo	% Diff
Ball snap	1916	66.3	67.1	0.9
Handoff	1776	39.7	40.7	0.2
Pass caught	1684	16.0	18.2	10.5
First contact	3156	9.0	12.5	28.0



Public GitHub: SumerSports/SportsTrackingTransformer

- So we have given a lot of theory about the Transformer architecture but we still need to demonstrate it is empirically better
- Pavel and Udit picked an objective that was very similar to the Zoos home turf: tackle location (x and y)
- We use an expanded ball-carrier dataset from 2024 that includes both run and run-after-catch plays
  - Notably this dataset has all frames the ballcarrier was active instead of just the frame of handoff
- The validation set was used for early stopping for both models
- Findings:
  - Zoo and Transformer are about equal at the task Zoo was designed for: predicting outcomes at handoff
  - As the frames and events diverge from that situation, the Transformer outperforms the Zoo significantly
- Our code and methodology is all available on our public github: **https://github.com/SumerSports/SportsTrackingTransformer**

Example: Test Set: CAR vs NYG, 1st and 10 @ CAR 25, Frame 45
https://youtu.be/stsa_Llesak?si=n5pbplwMzFq1T8nn&t=573



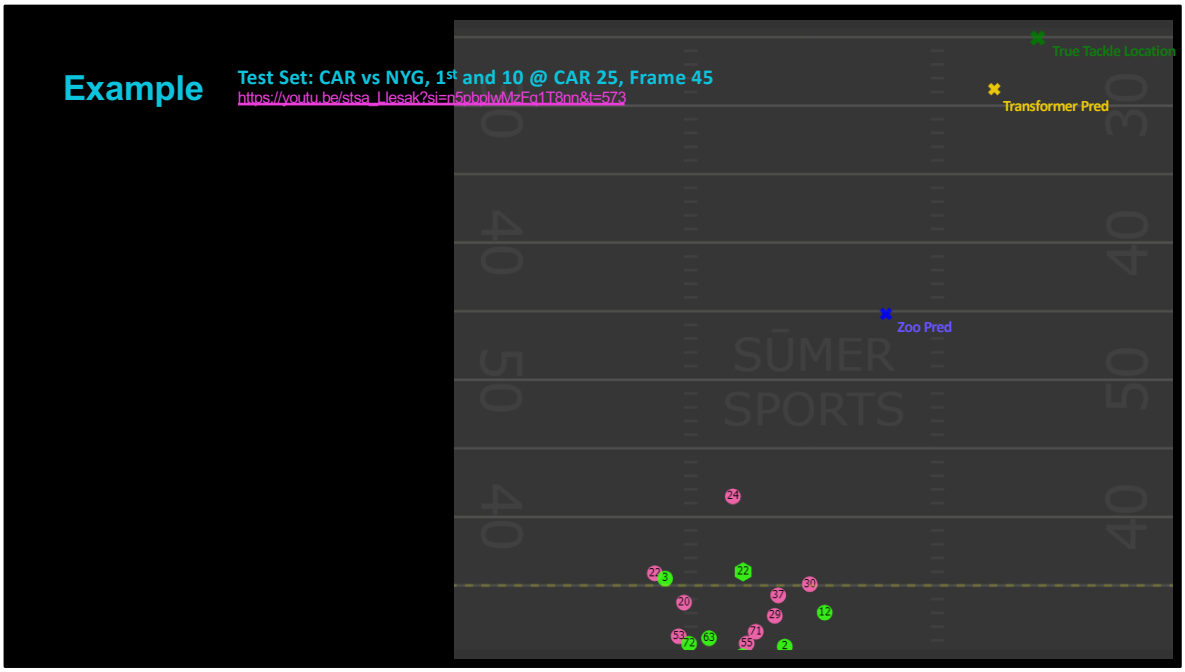
Example from test set so neither model was trained on this play

CMC as broken into the third level and just has a safety to beat.

How many yards do we think CMC will get?

Example

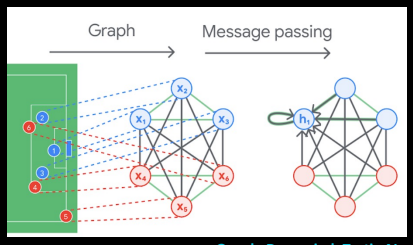
Test Set: CAR vs NYG, 1st and 10 @ CAR 25, Frame 45
https://youtu.be/stsa_Llesak?si=n5obplwMzEq1T8nn&t=573



- CMC beats the safety easily and breaks off a huge 49-yard run.
- Both models do a good job at predicting a big run, but transformer generalized the big upside of this play better.
- Example of a prediction from “late in the play” as RB has already broken to the third level.

Other Invariant Architectures

- Set-based Learning (Horton 2020, Zaheer 2017)
  - Uses an invariant architecture, before Transformers became a powerful standard
- Graph Neural Nets (Wang, Veličković, et al 2024)
  - Tracking Data, like Natural Language, are a special case of graphs
    - Quite small (~25 nodes)
    - Dense (fully connected)
    - Few or No edge features
  - For sports, we think modern GNNs with attention and SportsTrackingTransformer are very similar
  - Alcorn and Nguyen [2021] found Transformers outperformed a GNN empirically for Basketball
- Invite empirical comparisons to these options!



Google Deepmind: Tactic AI

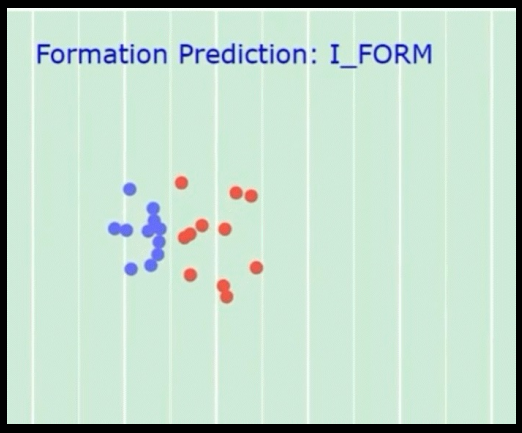
SUMERSPORTS

Can the Transformer be viewed as a special case of a Graph Neural Network (GNN)?

- We think Graph Neural Nets are promising and a cool direction
- Our belief is that the graph for sports is a special case that is covered by just using a simple Transformer
- The core idea of using Attention to learn across players is the same in both

NFL BigDataBowl 2025 starter notebook

- Set up SportsTracking Transformer on Kaggle
- Trained on only 1 week of data to predict offensive formation



https://www.kaggle.com/code/pvabish/modeling-with-transformers-by-sumersports

SUMERSPORTS

* This was the exact same model architecture just with slightly different final layer modified for a classification task instead of regression

Code Walkthrough

SumerSports/SportsTrackingTransformer

SÜMERSPORTS

- We don’t implement *any* custom NN stuff, it is all plug-n-play with built in Transformers!
- The Zoo Model implementation is considerably more complex
    - Generating all the custom features
    - Hardcoded pooling hyperparameters