



# Machine Learning Based ESports Winner Prediction

ShanghaiTech CS181 Project : LOL Esports Prediction  
连奕航/邬一闻/熊昕洋/吕钧霆/余诗博

A dark silhouette of a warrior in a dynamic pose, holding a sword and a shield. The warrior is positioned on the left side of the frame, facing right. The background is a deep blue gradient with stylized, layered mountain ranges. In the upper right corner, there are decorative elements including a bright lens flare and several overlapping semi-transparent circles.

01

# Introduction

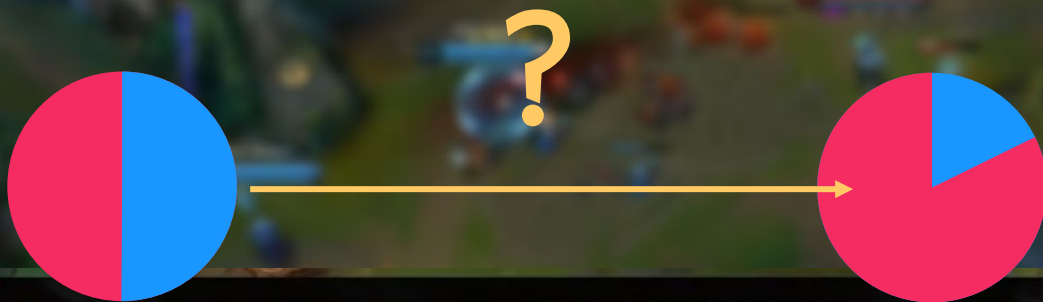
When watching esports matches,  
did you notice this?





# LOL Esports Winner Prediction

- Example: KI Colonel (KFC-AI) Realtime Win Rate
- Who to win the game?



02

# Our Methods

How to determine who can win the game?



# WHO WILL WIN THE GAME?



# WHO WILL WIN THE GAME?



# Supervised Machine Learning Approach

$$f\left(\begin{matrix} \text{CHAMPION} \\ \text{LINEUP} \end{matrix}, \begin{matrix} \text{IN-GAME} \\ \text{PERFORMANCE} \end{matrix}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



**ENCODING**

Extract feature vector  
from current game state



**CLASSIFICATION**

dimensions of vector  
= # of players



# Champion Lineup: Naive Bayes



# Champion Lineup: Naive Bayes

Laplace Smoothing →

$P(Y)$   $Y = \{ \text{Red}, \text{Blue} \}$

$$\rightarrow P(Y, F_1, F_2, \dots, F_n) \\ = P(Y) \prod_i P(F_i|Y)$$

$$\rightarrow P(Y|F_1, F_2, \dots, F_n)$$



$P(F_i|Y)$   $i=1, 2, \dots, 10$

# Champion Lineup: Naive Bayes

## Result & Thoughts

- Result:
  - 54.0% on 2021 pro matches dataset
  - 52.7% on 2015-2018 pro matches dataset
  - even monkey can get 50% accuracy...
- Why poor performance?
  - Naive Bayes assumes conditional independence on champions
    - however, champion cooperation&counter matters!
  - Pro players might be influenced little by champion lineup
    - they tend to pick OP champion lineup as possible
  - Game patch differs

# Champion Lineup: Logistic Regression

## Result & Thoughts

$$f\left(\begin{array}{c} \text{TOP JUN MID ADC SUP} \\ \text{TOP JUN MID ADC SUP} \end{array}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- Using Gradient Descent to optimize log loss with sigmoid probability
- Result:
  - 54.1% on 2021 pro matches dataset
  - 53.2% on 2015-2018 pro matches dataset
- This can prove champion lineup less matters to pros than expected

# In-Game Performance

$$f\left(\begin{array}{c} \text{Gold} \\ \text{Tower} \\ \text{Kill} \\ \text{Dragon} \end{array}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

# In-Game Performance: Logistic Regression

## Result & Thoughts

$$f\left(\begin{array}{c} \text{Gold} \\ \text{Tower} \\ \text{Kill} \\ \text{Dragon} \end{array}\right) = \begin{pmatrix} \text{red} \\ \text{blue} \end{pmatrix} \text{ or } \begin{pmatrix} \text{blue} \\ \text{red} \end{pmatrix}$$

- Similarly, using Gradient Descent to optimize loss
- Features captured:
  - Difference between blue and red in ...
  - ... gold earned/enemies killed/towers destroyed/monsters killed
  - At certain timestep (early stage), like t=15 minutes after beginning
- Result:
  - 76.0% on 2021 pro matches dataset
  - 72.3% on 2015-2018 pro matches dataset



# In-Game Performance: RNN

Realtime prediction based on Recurrent Neural Network

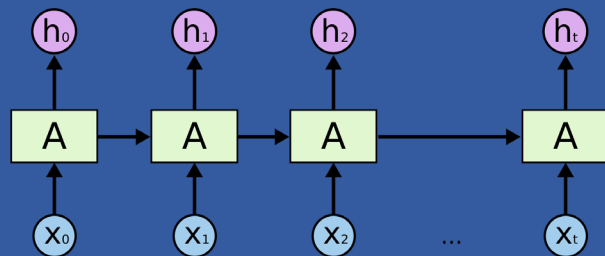
$$X_t = \left( \begin{array}{c} \text{Gold} \\ \text{Tower} \\ \text{Kill} \\ \text{Dragon} \end{array} \right)_t \quad t=1,2,\dots,T$$

- We want to implement real time prediction!
- Features at each previous time step are captured  $X_1, X_2, \dots, X_t$
- Our choice - RNN !
  - can encode **arbitrary-length** input ( $T=5, 10, 15, 20, 25, 30$  in our project)
  - can do **classification** through an additional output layer
  - With **LSTM** cells, can preserve more information in early time steps
  - LSTM: Long Short Term Memory networks

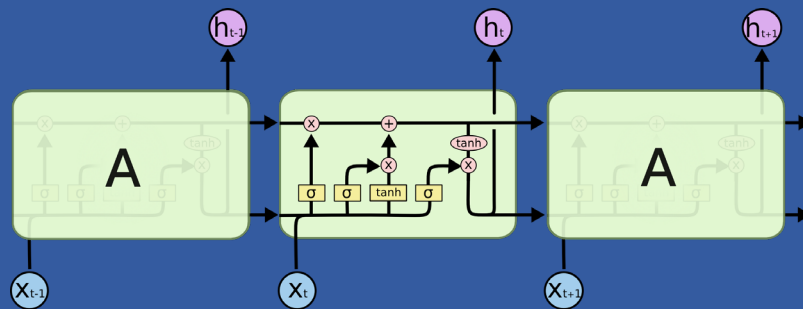
# In-Game Performance: RNN & LSTM

Using Toolkit: PyTorch

$$X_t = \left( \text{Gold}, \text{Tower}, \text{Kill}, \text{Dragon} \right)_t \quad t=1,2,\dots,T$$



nn.RNN



nn.LSTM

batch size = 32, learning\_rate = 0.001, loss\_func = CrossEntropy, activation = ReLU, RNN hidden size = 256, LSTM num layers = 1, train:valid:test split ratio = 6:2:2, early stopping on valid accuracy

# In-Game Performance: RNN

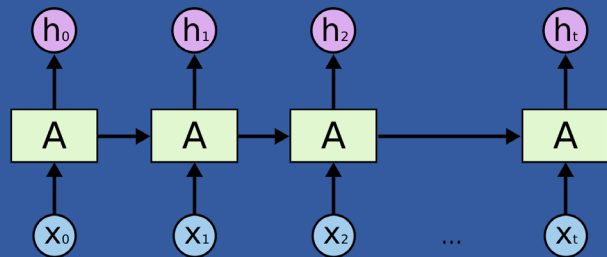
Realtime prediction based on Recurrent Neural Network

- Result on 2015-2018 pro matches dataset :

58.0%	05 min RNN LSTM	58.9%
66.5%	10 min RNN LSTM	66.7%
73.1%	15 min RNN LSTM	73.9%
77.3%	20 min RNN LSTM	77.4%
82.7%	25 min RNN LSTM	82.9%
84.4%	30 min RNN LSTM	85.0%

# In-Game Performance: RNN & LSTM

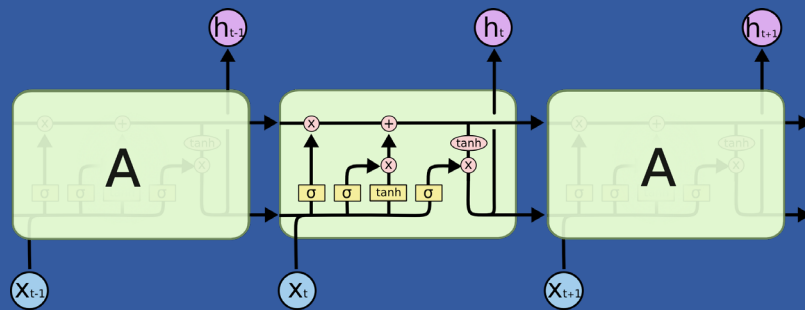
Using Toolkit: PyTorch



nn.RNN

84.4%

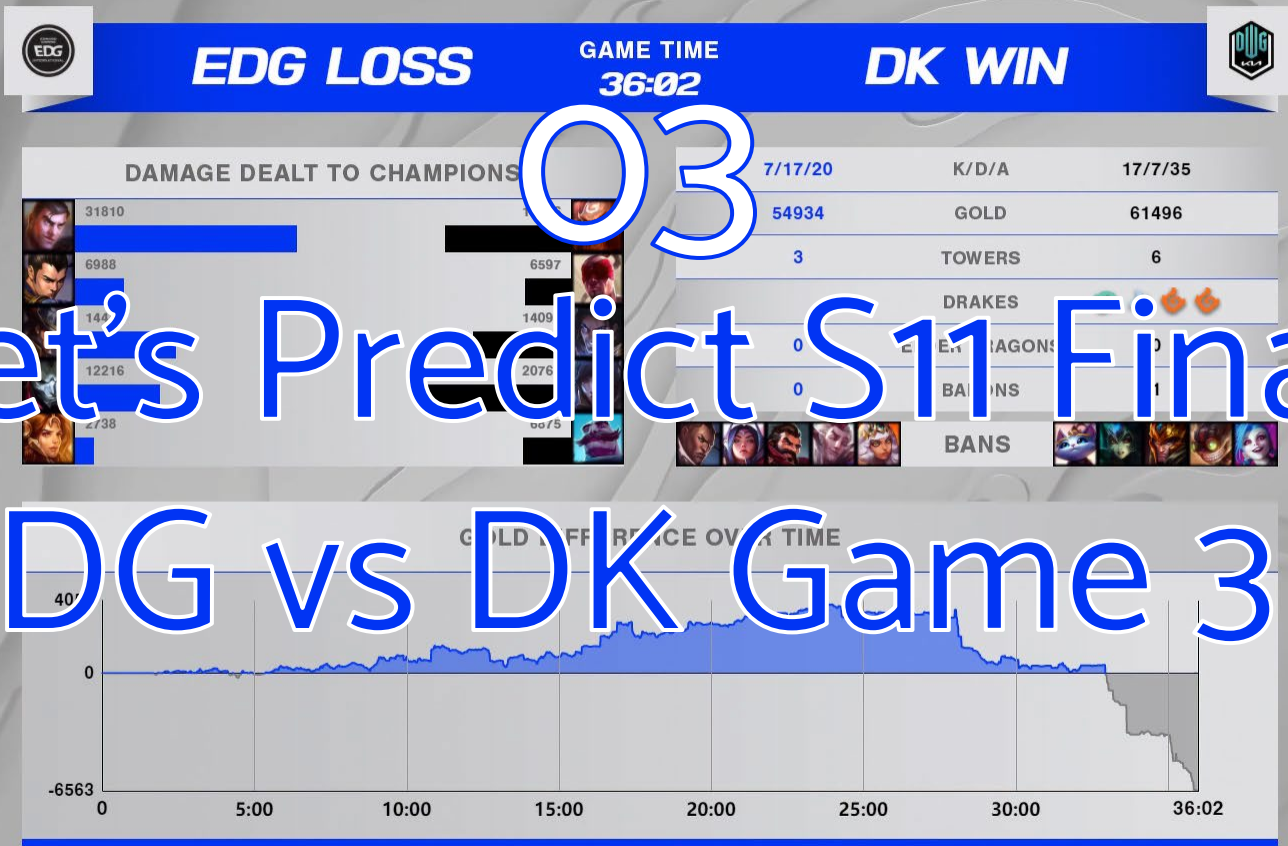
30 min  
RNN LSTM



nn.LSTM

85.0%

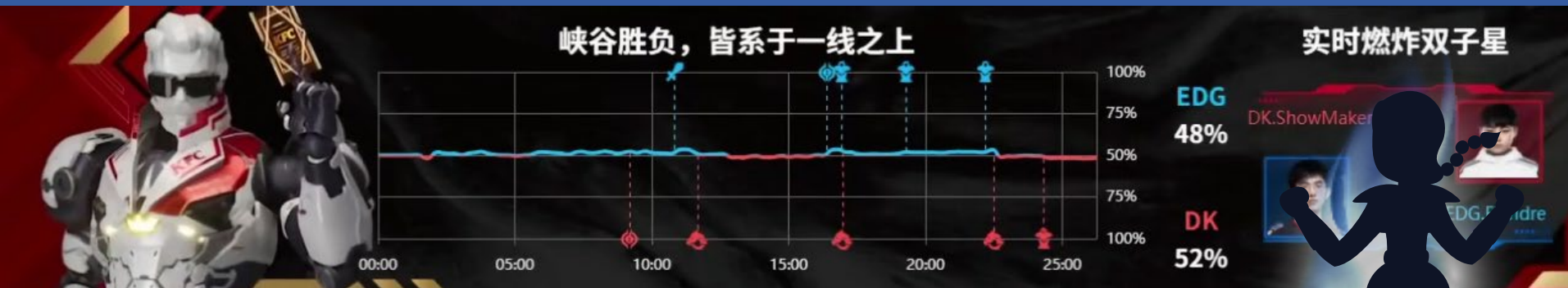
- Improvement of LSTM seems not obvious from accuracy metric
- But we can demonstrate their difference by following example



Let's Predict S11 Final  
EDG vs DK Game 3!

# Predict S11 EDG vs DK Game 3

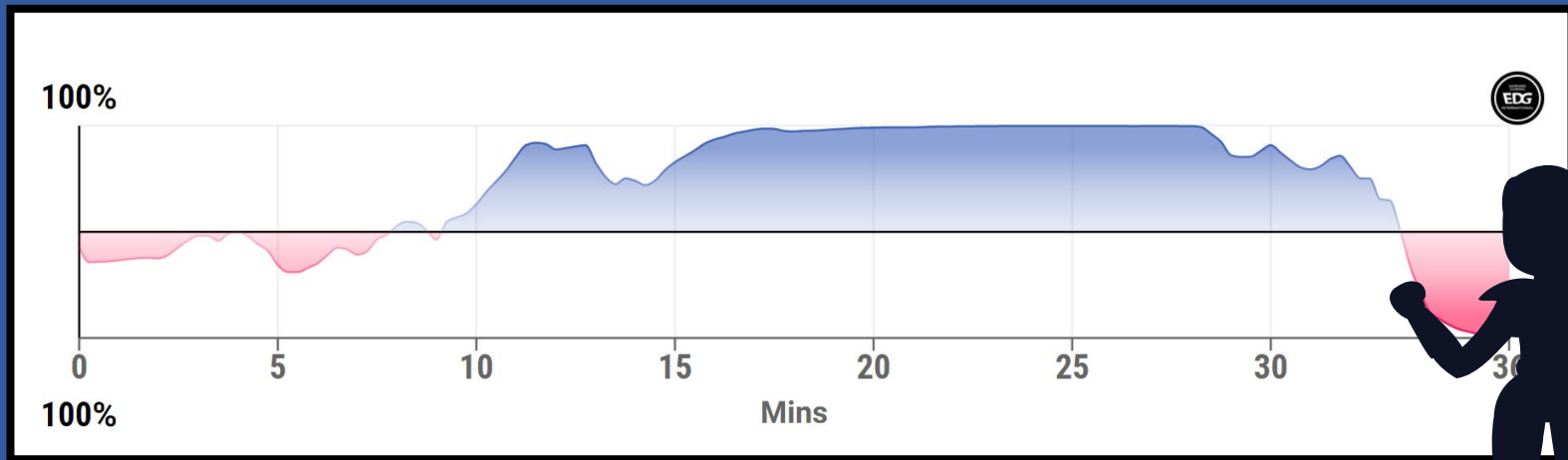
Source: KI Colonel (KFC-AI)





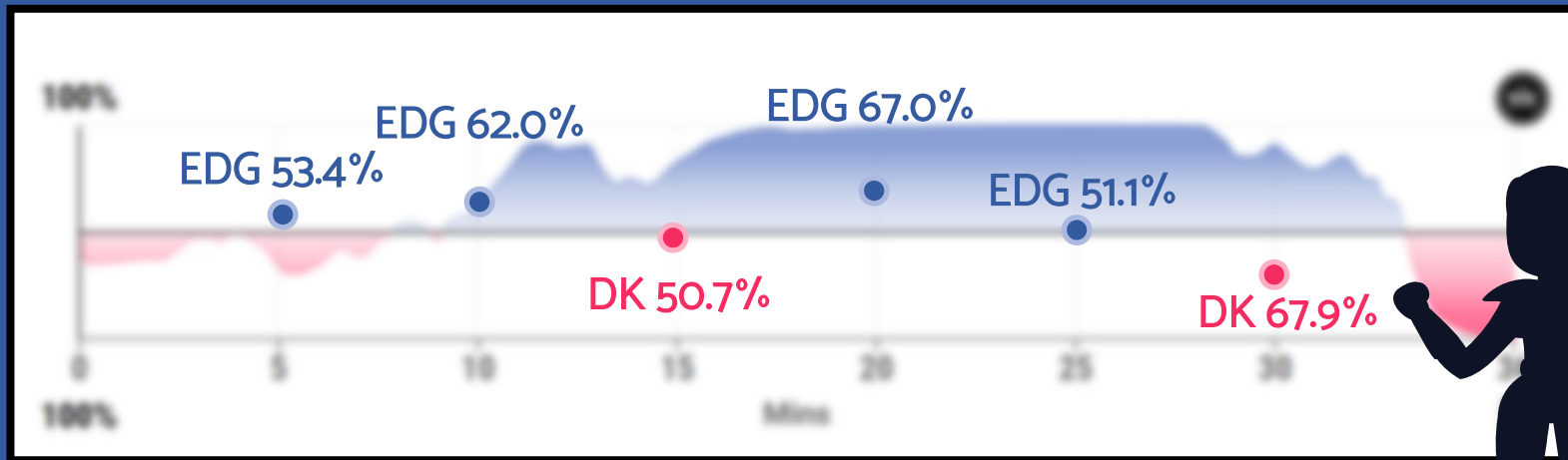
# Predict S11 EDG vs DK Game 3

Source: FACTOR.GG



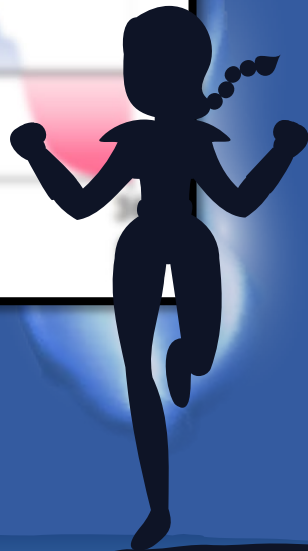
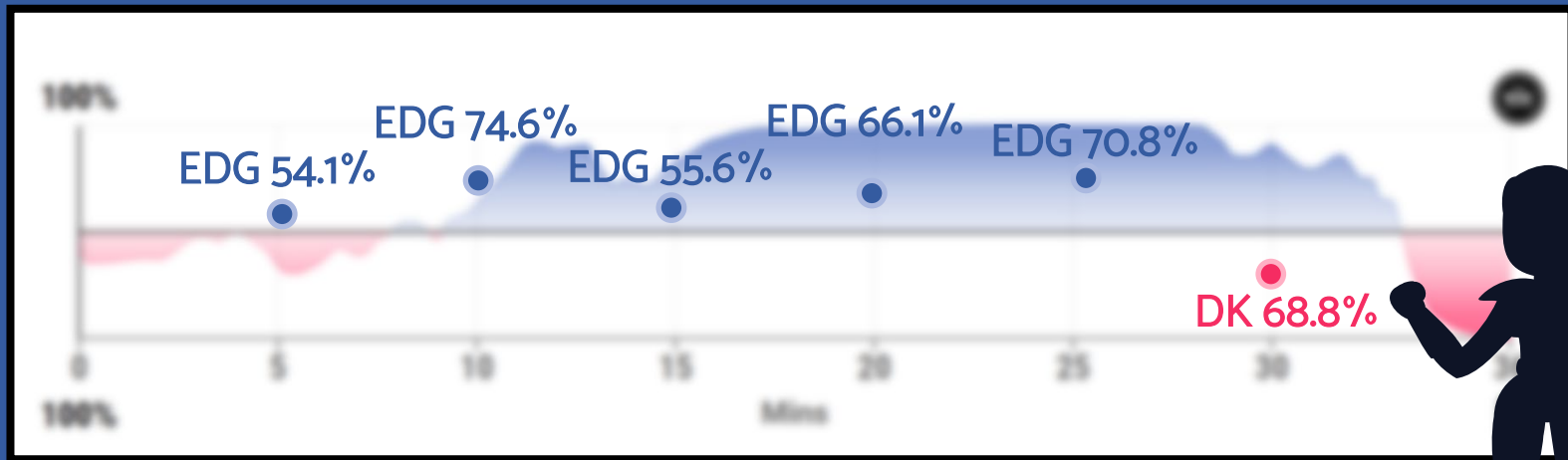
# Predict S11 EDG vs DK Game 3

Our LSTM RNN Model Prediction!



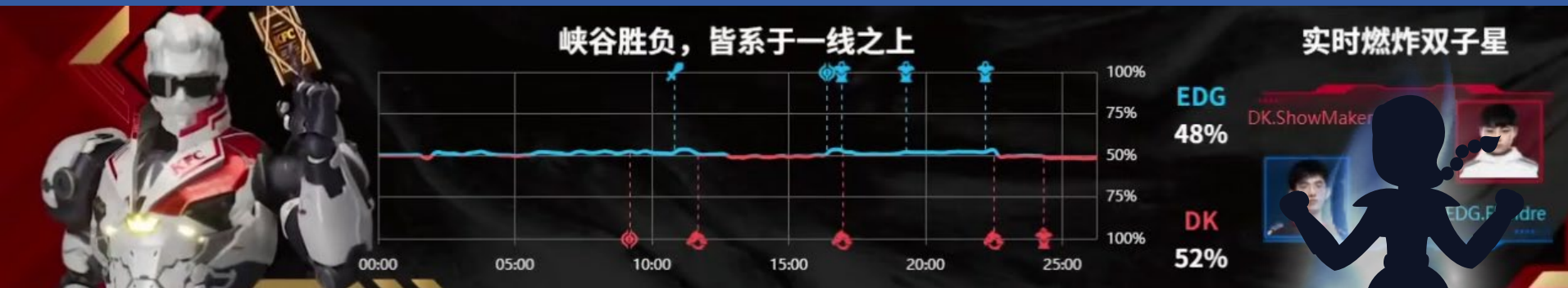
# Predict S11 EDG vs DK Game 3

Our RNN Model Prediction!



# Predict S11 EDG vs DK Game 3

Source: KI Colonel (KFC-AI)



# Predict S11 EDG vs DK Game 3

Our LSTM RNN Model Prediction!



A black silhouette of a dragon with its wings spread, breathing a stream of fire. The fire is depicted as several bright yellow and orange streaks that curve upwards and to the right, illuminating the word 'THANKS!'.

# THANKS!

CREDITS: This presentation template was created  
by **Slidesgo**, including icons by **Flaticon**, and  
infographics & images by **Freepik**

Please keep this slide for attribution

