Deep Single Table Learning

Teacher: Zheng Wang

TA: Jianwu Zheng

Social Network Analysis (NIS8023)

Shanghai Jiao Tong University

Outline

- **Deep Table Learning**
- Data Preprocessing
- Neural Network Structure

Deep Learning Development History

- 1940s–1960s: Neuron Models, Perceptron.
- 1969: "*Perceptrons*".
- 1980s–1990s: Backpropagation, Early CNNs & RNNs.
- 2006: Deep Belief Networks (DBNs).
- 2012: AlexNet.
- 2014–Present: GANs, Transformers, BERT, GPT-3, ChatGPT.

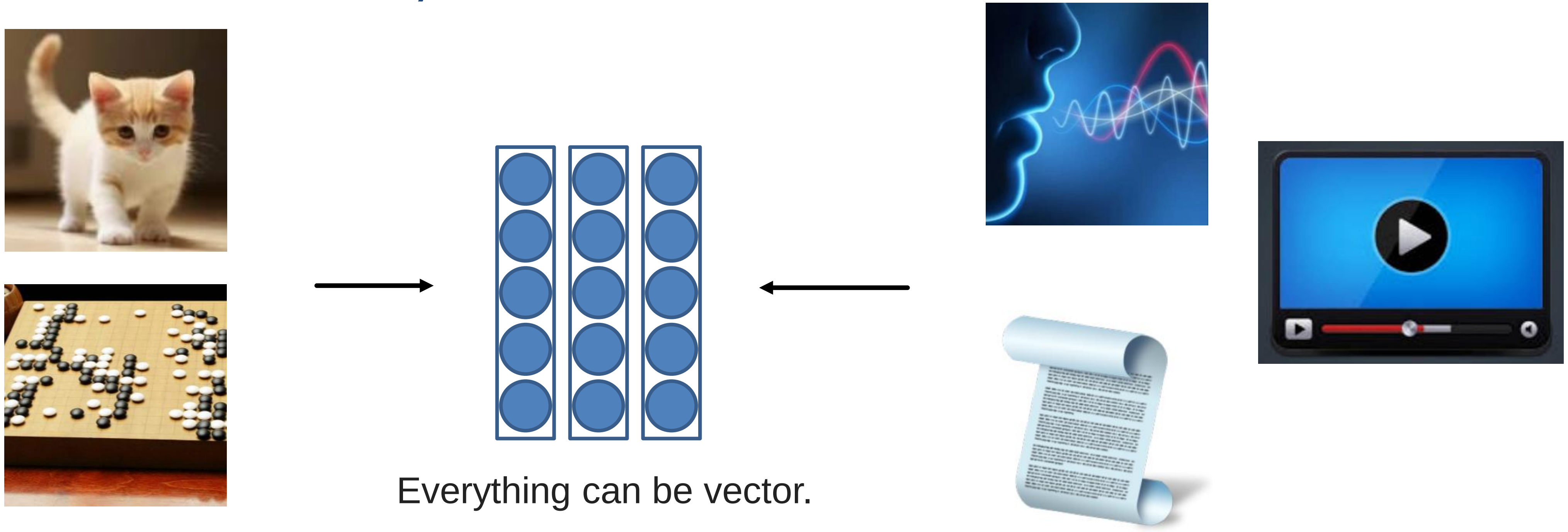
Deep Learning

- Pioneers of Deep Learning honored with Turing Award, the Nobel Prize of computing.

From left to right: Yoshua Bengio, Geoffrey Hinton, Yann Lecun

What is Deep Learning?

- Deep learning can be :
  - A way of learning.
  - A way to get data.
  - A way to combine data freely.

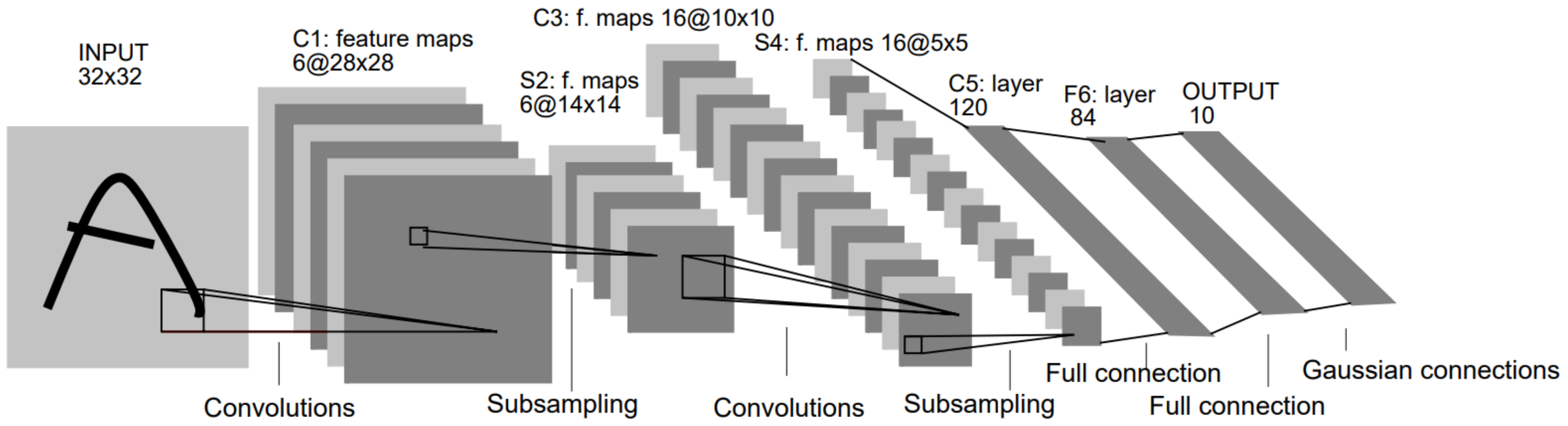

Everything can be vector.

Advantages of Deep Neural Networks (DNNs)

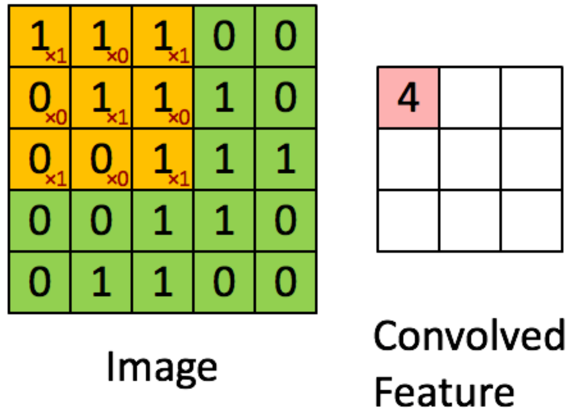
1. Performance improvements particularly for large datasets.
2. Efficiently encoding multiple data types like images along with tabular data.
3. Alleviating the need for feature engineering, which is currently a key aspect in tree-based tabular data learning methods.
4. End-to-end models allow representation learning which enables many valuable application scenarios including data-efficient domain adaptation.

Convolutional Neural Networks (CNNs)

- A convolutional neural network is a regularized type of feedforward neural network that learns features by itself via filter (or kernel) optimization.



An example of a CNN structure.

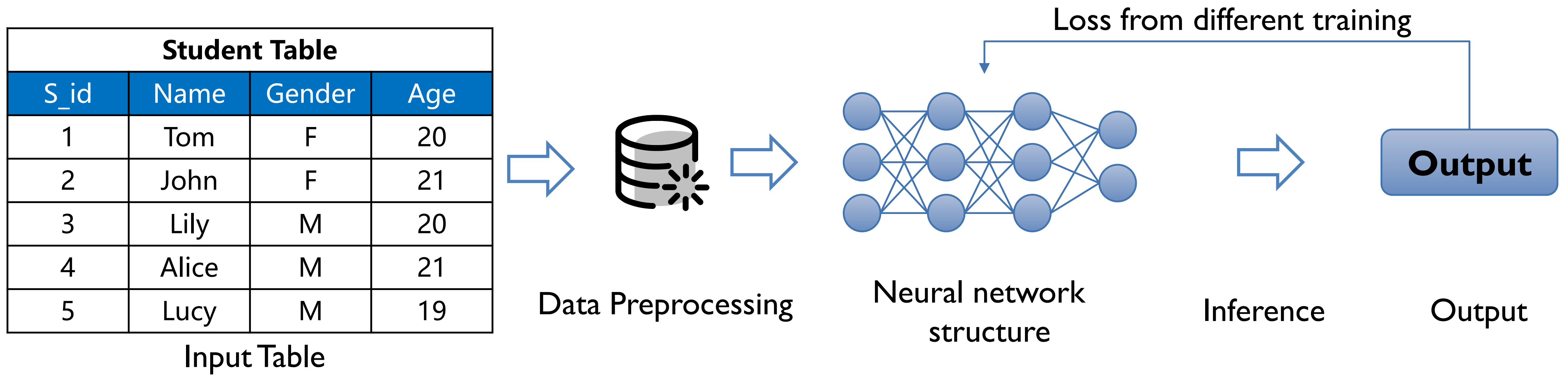


A demonstration of CNN principles.

LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

Tabular Neural Networks (TNNs)

- TNNs refer to these deep neural network methods on tabular data
  - Data preprocessing
  - Neural network structure
  - Training strategy



Student Table

S_id	Name	Gender	Age
1	Tom	F	20
2	John	F	21
3	Lily	M	20
4	Alice	M	21
5	Lucy	M	19

Deep Table Learning Framework

Challenges in TNNs

1. Low-quality training data
2. Missing or complex irregular spatial dependencies
3. Dependency on preprocessing
4. Importance of single features

Outline

- Deep Learning
- **Data Preprocessing**
- Neural Network Structure

Recap: Data Preprocessing in Traditional Methods (1)

- Empty cell imputation
  1. Tabular data tends to have a lot of missing values.
  2. Missing values can affect the training effect.
  3. Do not change the original data distribution.
- Methods

Age
20
21
?
21
22

Method	Fill value
most frequent	21
sampling	~ N
0 value	0
KNN/Cluster	-

Categorical features

Score
96
86
?
94
86
88

Method	Fill value
mean	90
median	88
mode	86
0 value	0
forward fill	86
backward fill	94
interpolation	90

Continuous features

Recap: Data Preprocessing in Traditional Methods (II)

- Reduction and normalization
  - Column-wise & Row-wise
  - Min-Max Normalization, Z-Score/Standard.
- Error data repair
  - Data repair is the process of fixing errors and inconsistencies in data to ensure its accuracy and reliability for data analysis.

Student Table

S_id	Name	Gender	Age	Score
1	Tom	F	20	96
2	John	**FA**	21	86
3	Lily	M	?	89
4	Alice	?	22	94
5	Lucy	?	?	80

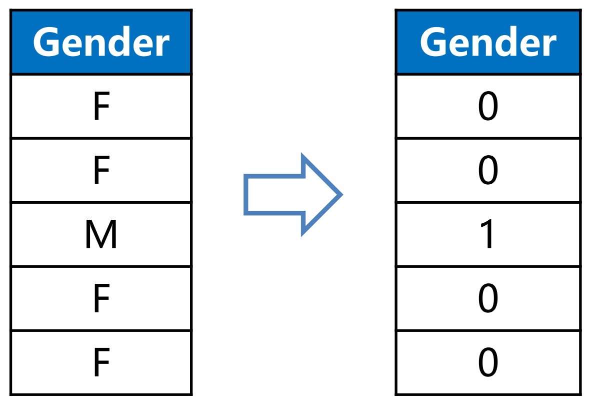
→

Student Table

S_id	Name	Gender	Age	Score
1	Tom	F	20	96
2	John	**FA**	21	86
3	Lily	M	?	89
4	Alice	?	22	94
5	Lucy	?	?	80

Data Preprocessing in Deep Learning (I)

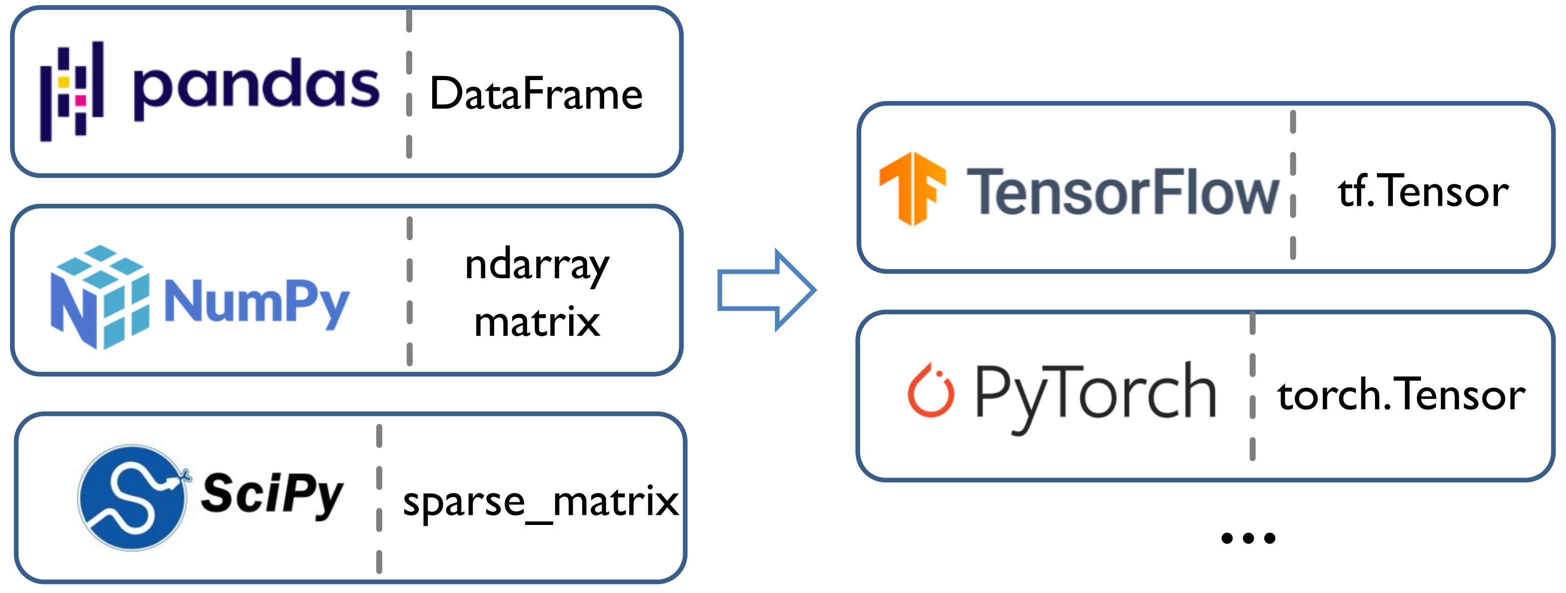
- Data conversion:
  - Convert non-numerical to numerical type values.
  - Take the data from the format it was read in to data formats in deep learning platforms.



Gender
F
F
M
F
F

Gender
0
0
1
0
0

Value conversion



Data storage format conversion

Data Preprocessing in Deep Learning (II)

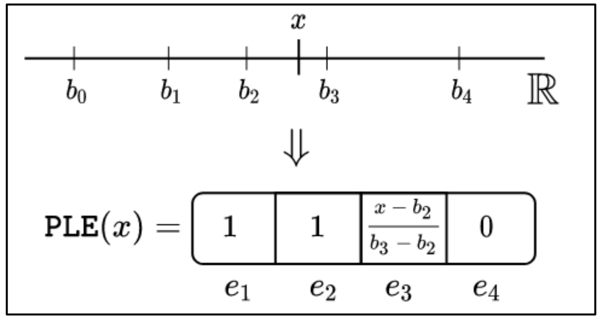
- Dimension enhancement:
  - Essentially a coding technique that requires no training.
  - Transform the feature dimension from low to high.
- Why?
  - Retain more information for better model utilization.

Data Preprocessing in Deep Learning (III)

- Methods
  1. Categorical features: One-hot Encoding
  2. Continuous features: Stack Encoding, Piecewise Linear Encoding(PLE), Periodic Activation Function

$$f_i(x) = \texttt{Periodic}(x) = \texttt{concat}[\sin(v),\ \cos(v)], \qquad v = [2\pi c_1 x,\ \ldots,\ 2\pi c_k x]$$

Periodic Activation Function



PLE

Feature	Stack	PLE	Periodic(c1=1,c2=2)
1.0	[1.0, 1.0, 1.0]	[1, 0, 0]	[0.0, 0.0, 1.0, 1.0]
1.25	[1.25, 1.25, 1.25]	[1, 0.25 , 0]	[1.0, 0.0, 0.0, -1.0]
3.75	[3.5, 3.5, 3.5]	[1, 1, 1]	[-1.0, 0.0, 0.0, -1.0]

Examples of different dimensionality raising methods for continuous features.

Outline

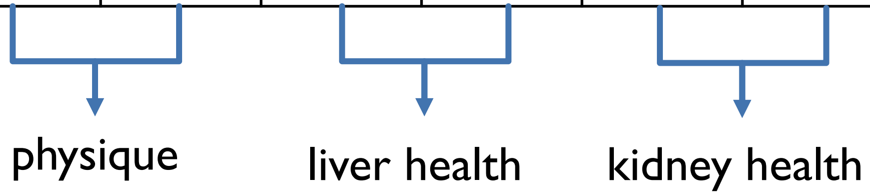
- Deep Learning Methods
- Data Preprocessing
- **Neural Network Structure**
  - Column Pre-encoder
  - Backbone Structure

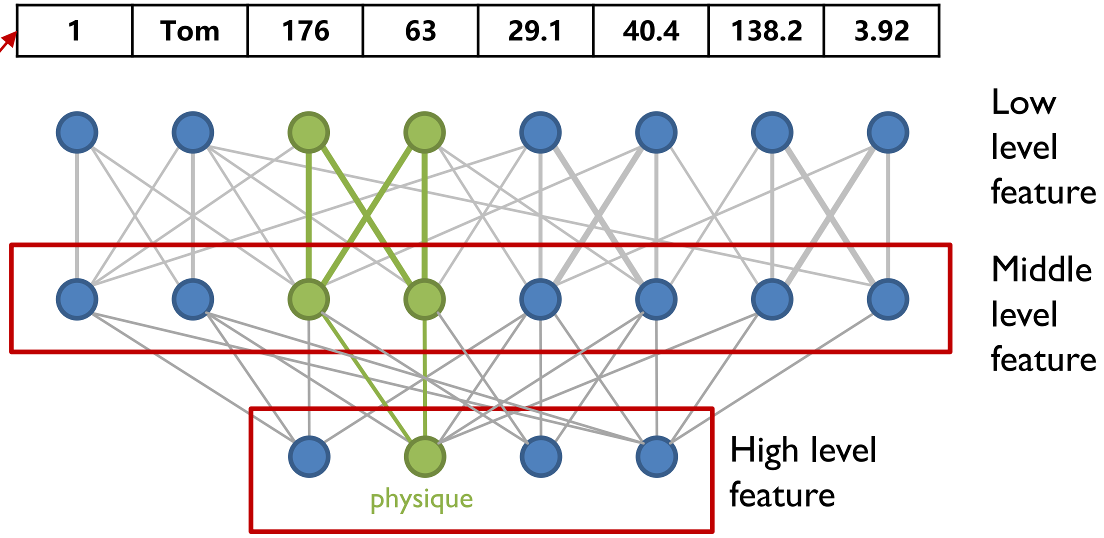
Intuition

- The purpose is to explore the relationship between columns.
- The relationships between columns are complex.

Physical Examination Table

Id	Name	Height	Weight	Globulin	Albumin	Na+	K+
1	Tom	176	63	29.1	40.4	138.2	3.92
2	John	183	85	34.2	49.3	147.0	4.62
3	Lily	162	47	32.1	37.5	136.3	3.68
4	Alice	170	54	28.5	43.7	140.2	3.97
5	Lucy	167	53	33.4	41.2	144.3	4.25

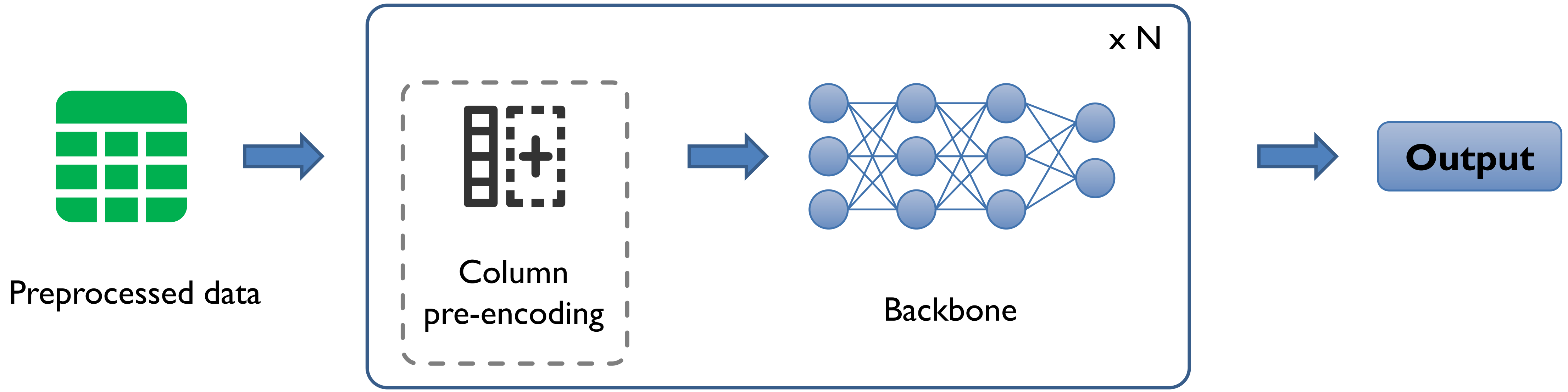




Chen J, Liao K, Wan Y, et al. Danets: Deep abstract networks for tabular data classification and regression[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2022, 36(4): 3930-3938.

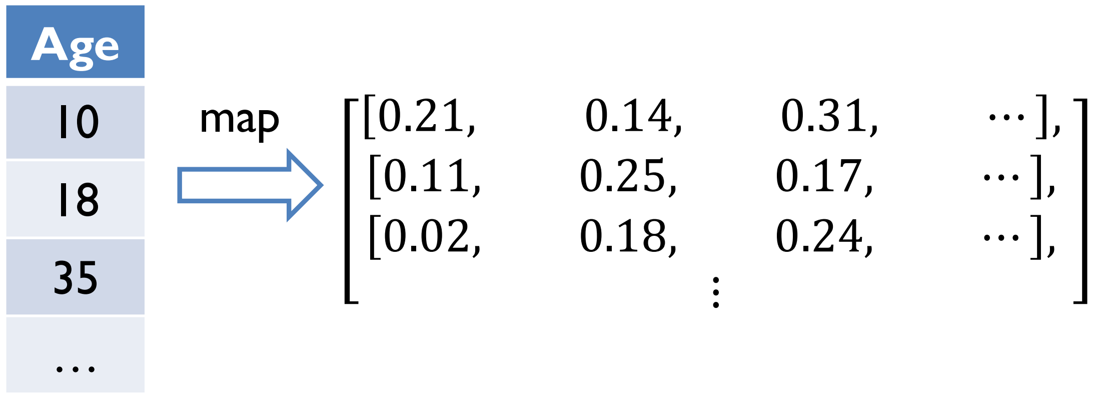
Neural Network Structure

- TNNs’ structure consist of two parts :
  - Column type pre-encoding
  - Backbone construction



What's Column Type Pre-encoding?

- Mapping the scalar values of these features to high-dimensional embedding vectors, with benefits:
  - Intuition: Model the semantic information of "scalar-type" values, by introducing some **trainable** mapping functions.
  - Location: Often at the beginning of the backbone network.



Map (via trainable NNs) a column for "scalar" to "vectors"

Why Column Type Pre-encoding? (1)

- Diversity of tabular data:
  - There are heterogeneous data consisting of continuous, categorical, and text features.
  - Heterogeneous nature of tabular feature spaces.

Student Table

S_id	Name	Gender	Age	Score
1	Tom	F	20	89
2	John	F	21	85
3	Lily	M	22	92

Why Column Type Pre-encoding? (II)

- Problems with raw data:
  1. Difficult to extract hidden information from data.
  2. Difficult to capture complex relationships between features.

Student Table

S_id	Name	Gender	Age	Score
1	Tom	F	20	89
2	John	F	22	85
3	Lily	M	21	92

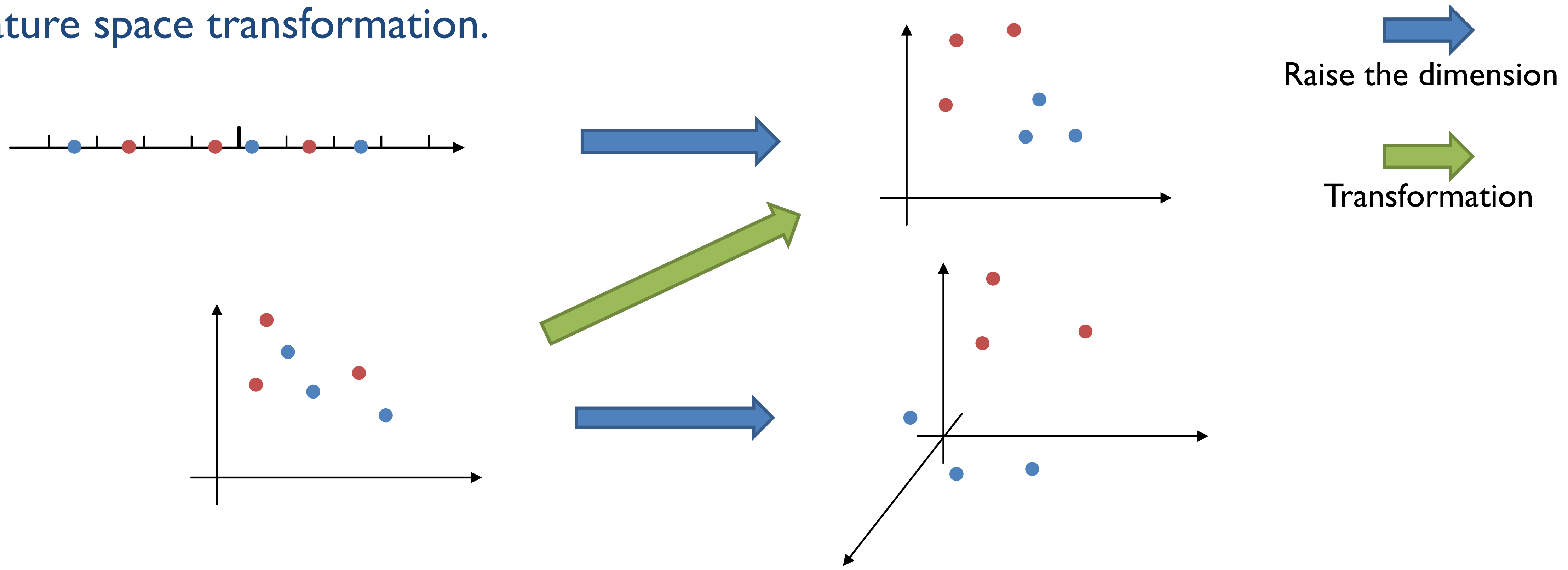
Raw values alone are difficult to compare. Do 20 and 89 mean the same thing?

Adding the original column name makes it easy to tell the meaning of 20 and 89.

The distribution of the six data points on the number line.

What Does Column Type Pre-encoding Do?

1. From low dimensions to high dimensions.
2. Feature space transformation.



For example, suppose you have a one-dimensional data distribution, and you can either raise the dimension or transform the feature space.

Advantages of Column Type Pre-encoding

1. Enrich the semantic expressiveness of these scalar types.
2. Increase the model capacity by introducing trainable mapping function.
3. Reduce the influence of data noise.
4. Facilitate feature interaction and fusion in the backbone network.

Column Type Pre-encoding (1)

- Multi-layer perceptron (MLP) / Linear
  - Add one or more MLPs.
  - Better pre-encoding can be achieved with the help of nonlinear activation functions.

3.3	✖	[0.21,	0.14,	0.31,	0.42]	✚	[0.43]
1.2	✖	[0.21,	0.14,	0.31,	0.42]	✚	[0.43]
5.2	✖	[0.21,	0.14,	0.31,	0.42]	✚	[0.43]

An example of using MLP pre-encoder.

Column Type Pre-encoding (II)

- Look-up table
  - Maintain a trainable coding table.
  - A scalar value corresponds to a vector.
- Notes:
  - Scalar ranges for different columns should not overlap.

$$\begin{array}{|c|}\hline 0\\ \hline 1\\ \hline 2\\ \hline 3\\ \hline 4\\ \hline \cdots\\ \hline\end{array} \Rightarrow \begin{bmatrix} [0.21, & 0.14, & 0.31, & \cdots], \\ [0.11, & 0.25, & 0.17, & \cdots], \\ [0.02, & 0.18, & 0.24, & \cdots], \\ [0.42, & 0.05, & 0.34, & \cdots], \\ [0.07, & 0.27, & 0.23, & \cdots], \\ & & \vdots & \end{bmatrix}$$

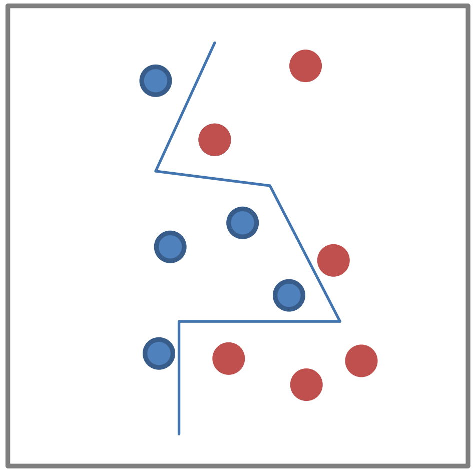
An example of using Look-up Table pre-encoder.

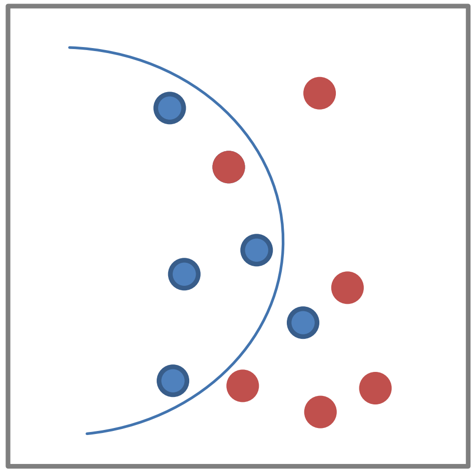
Backbone Construction

1. **Tree-mimic Structure**
2. Transformer-based Structure
3. Regularization-based Structure

Tree-mimic Structure

- Deepen the process using the principles of decision trees.
- The decision boundary of a decision tree can be irregular, aligning with the heterogeneity of tabular data.

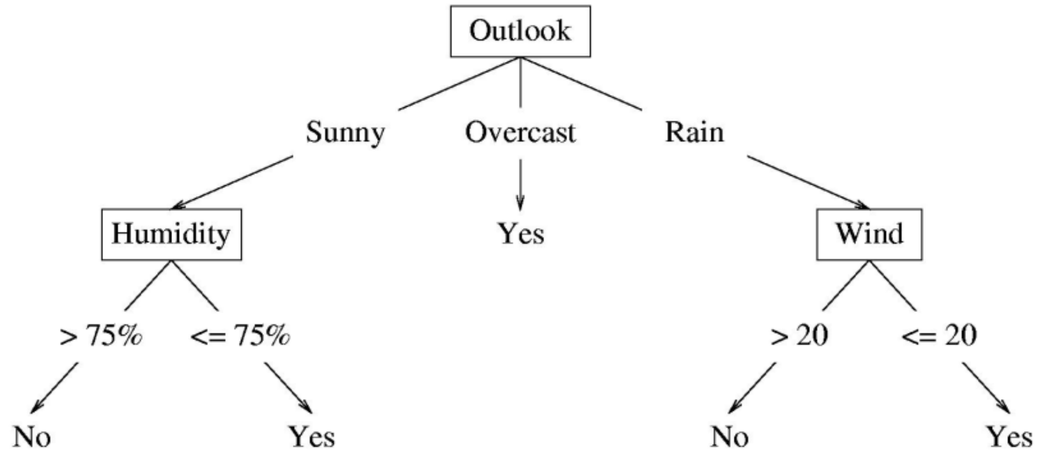

Decision Tree


MLP-like

An example of a two-dimensional plane used for classification.

Tree-mimic Structure

- Main idea
  1. Make decision trees differentiable for deep learning.
  2. Simulate the decision tree by training different thresholds corresponding to features.



1 → **Differentiable**

2 → **Simulated decision**

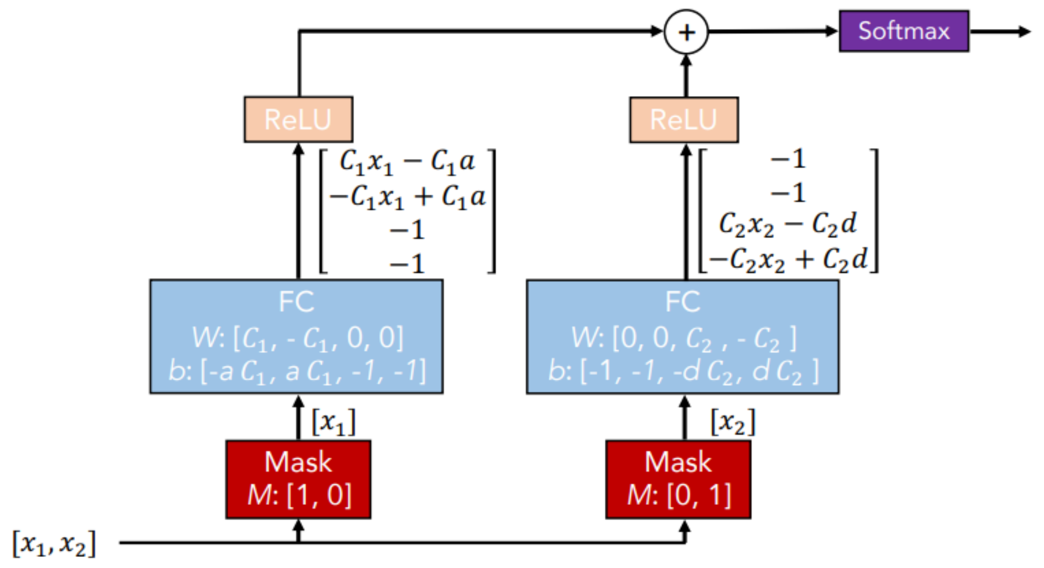
TabNet

- Motivation
  - Tree models such as XGBoost, LightGBM, etc. often work better than MLPS because they handle class features, missing values, and feature interactions well.
  - Neural networks perform poorly on structured data because:
    1. Important features cannot be selected efficiently and all inputs are processed;
    2. Lack of tree model decision path mechanism;
    3. Difficulty explaining model decisions (black box problem)

Arik S Ö, Pfister T. Tabnet: Attentive interpretable tabular learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(8): 6679-6687.

An example of DT-like classification

- DT-like classification using conventional DNN blocks



Softmax
+
ReLU
ReLU
$\begin{bmatrix} C_1x_1 - C_1a \\ -C_1x_1 + C_1a \\ -1 \\ -1 \end{bmatrix}$
$\begin{bmatrix} -1 \\ -1 \\ C_2x_2 - C_2d \\ -C_2x_2 + C_2d \end{bmatrix}$
FC
W: [C_1, - C_1, 0, 0]
b: [-a C_1, a C_1, -1, -1]
FC
W: [0, 0, C_2 , - C_2]
b: [-1, -1, -d C_2, d C_2]
$[x_1]$
$[x_2]$
Mask
M: [1, 0]
Mask
M: [0, 1]
$[x_1, x_2]$

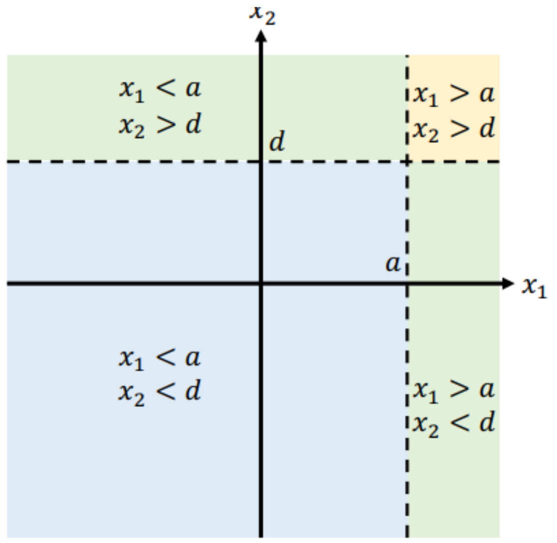
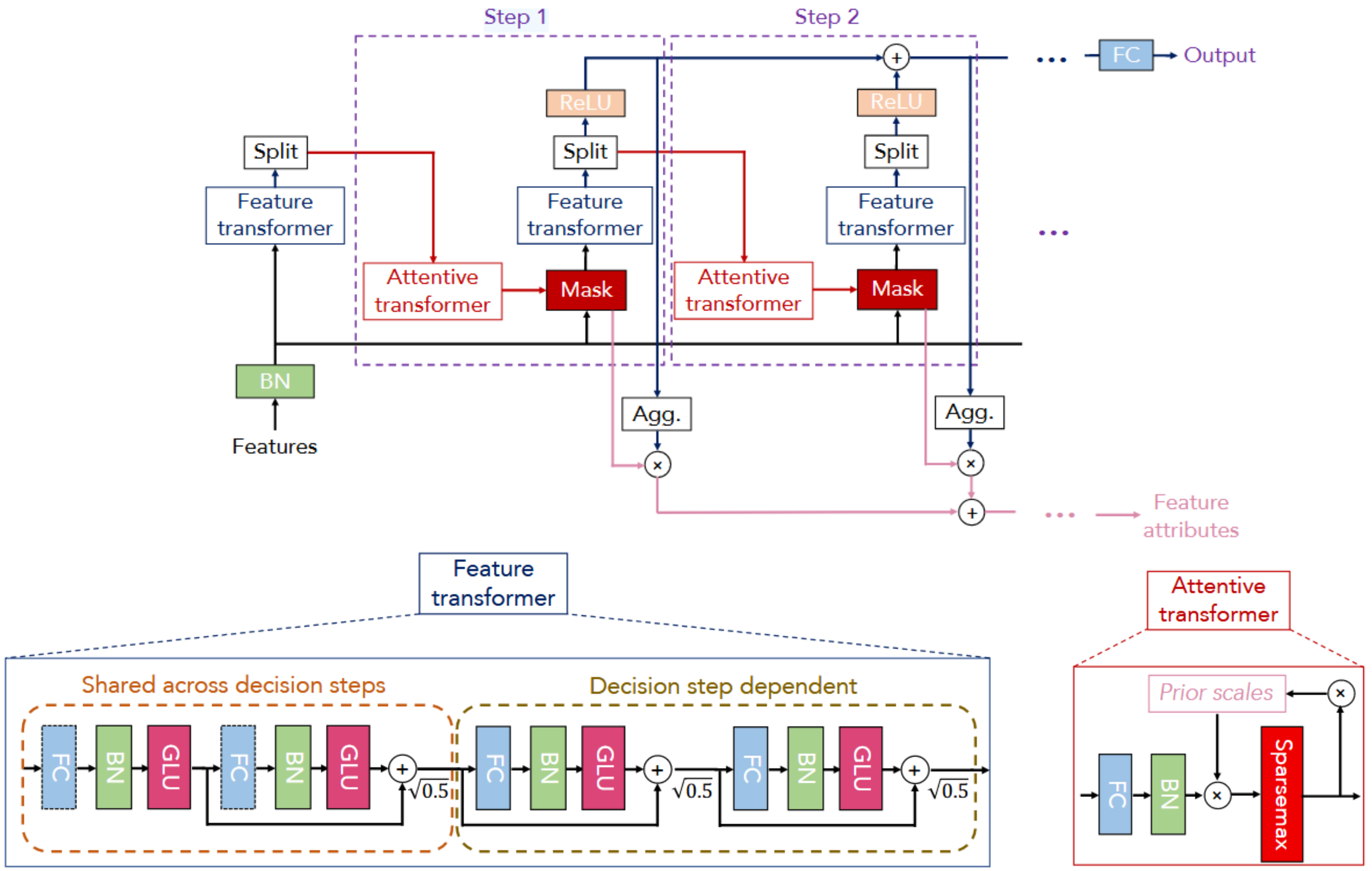


Figure 3: Illustration of DT-like classification using conventional DNN blocks (left) and the corresponding decision manifold (right). Relevant features are selected by using multiplicative sparse masks on inputs. The selected features are linearly transformed, and after a bias addition (to represent boundaries) ReLU performs region selection by zeroing the regions. Aggregation of multiple regions is based on addition. As C_1 and C_2 get larger, the decision boundary gets sharper.

The core of TabNet

1. Feature Selection:
   - Using the attention mechanism, each step selects the important features and ignores the unimportant ones.
   - This allows each step to focus on a different piece of information, similar to a different path in a decision tree.
2. Decision Step:
   - Each layer uses a feedforward neural network that processes the currently selected feature.
   - This is similar to how tree models learn different combinations of features at different depths.

Structure of TabNet



Arik S Ö, Pfister T. Tabnet: Attentive interpretable tabular learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(8): 6679-6687.

Sparse Feature Selection

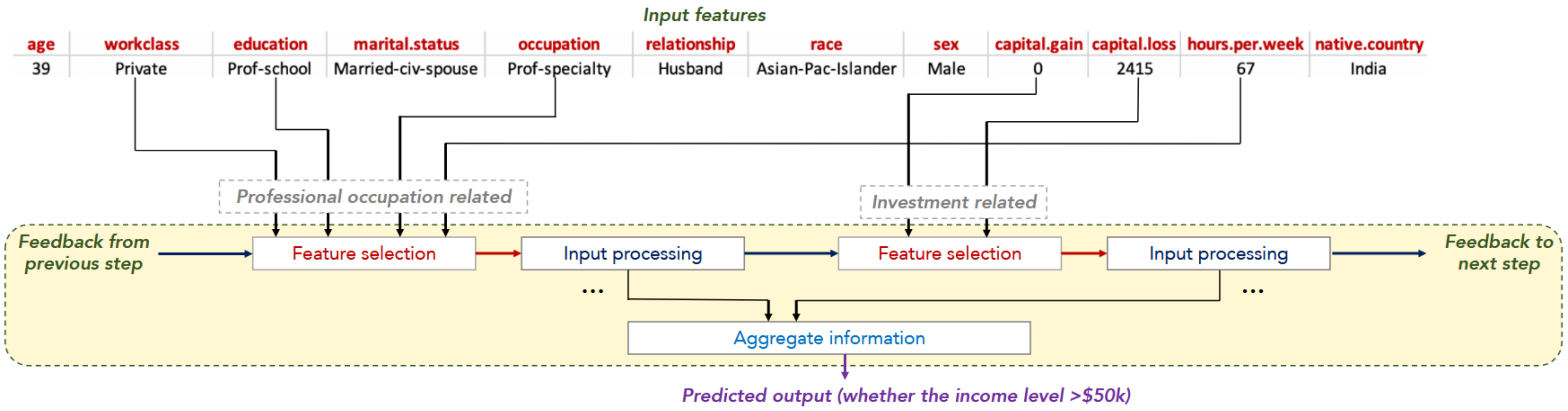


Figure 1: TabNet's sparse feature selection exemplified for Adult Census Income prediction (Dua and Graff 2017). Sparse feature selection enables interpretability and better learning as the capacity is used for the most salient features. TabNet employs multiple decision blocks that focus on processing a subset of input features for reasoning. Two decision blocks shown as examples process features that are related to professional occupation and investments, respectively, in order to predict the income level.

Arik S Ö, Pfister T. Tabnet: Attentive interpretable tabular learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(8): 6679-6687.

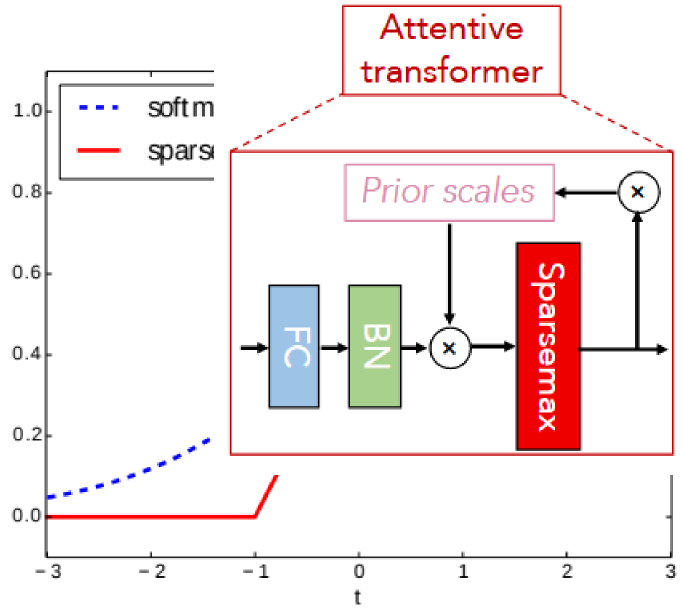
Sparse Feature Selection

- Soft feature selection with controllable sparsity in end-to-end learning.
- Employ a learnable mask $M[i] \in \mathbb{R}^{B \times D}$.
- The masking is multiplicative $M[i] \cdot f$, f is feature.

$$M[i] = \text{sparsemax}\big(P[i-1] \cdot h_i(a[i-1])\big)$$

$$P[i] = \prod_{j=1}^{i} (\gamma - M[j])$$

- $a[i-1]$ is attention score
- h_i is a trainable function, FC+BN
- γ is a relaxation parameter, if $\gamma = 1$, a feature is used only at once



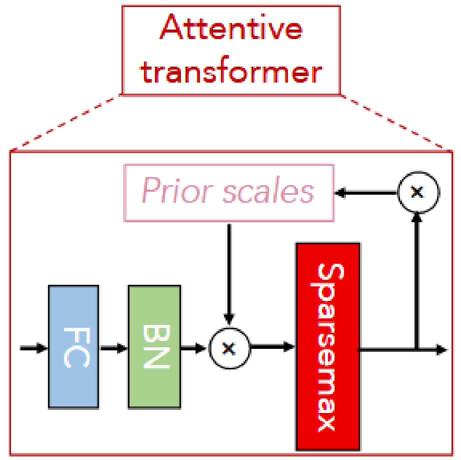
Martins A, Astudillo R. From softmax to sparsemax: A sparse model of attention and multi-label classification[C]//International conference on machine learning. PMLR, 2016: 1614-1623.

Sparse Feature Selection

- Soft feature selection with controllable sparsity in end-to-end learning.
- Employ a learnable mask $M[i] \in \mathbb{R}^{B\times D}$
- The masking is multiplicative $M[i] \cdot f$, f is feature

$$M[i] = \text{sparsemax}\big(P[i-1] \cdot h_i(a[i-1])\big)$$

$$P[i] = \prod_{j=1}^{i} (\gamma - M[j])$$



- Add sparsity regularization in the form of entropy.

$$L_{sparse} = \sum_{i=1}^{N_{steps}} \sum_{b=1}^{B} \sum_{j=1}^{D} \frac{-M_{b,j}[i] \log(M_{b,j}[i] + \epsilon)}{N_{steps} \cdot B}$$

Martins A, Astudillo R. From softmax to sparsemax: A sparse model of attention and multi-label classification[C]//International conference on machine learning. PMLR, 2016: 1614-1623.

Decision Step

Martins A, Astudillo R. From softmax to sparsemax: A sparse model of attention and multi-label classification[C]//International conference on machine learning. PMLR, 2016: 1614-1623.

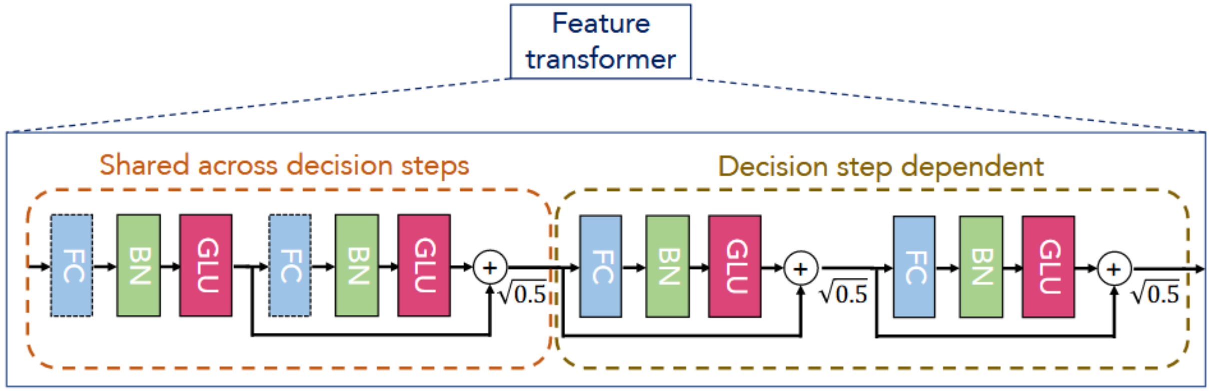
- Feature processing
  - Split for the decision step output and information for the subsequent step

$$[d[i], a[i]] = f_i(M[i] \cdot f)$$

  - The overall decision embedding:

$$d_{out} = \sum_{i}^{N_{steps}} ReLU(d[i])$$

$$out = W_{final} d_{out}$$



Result of Feature Importance

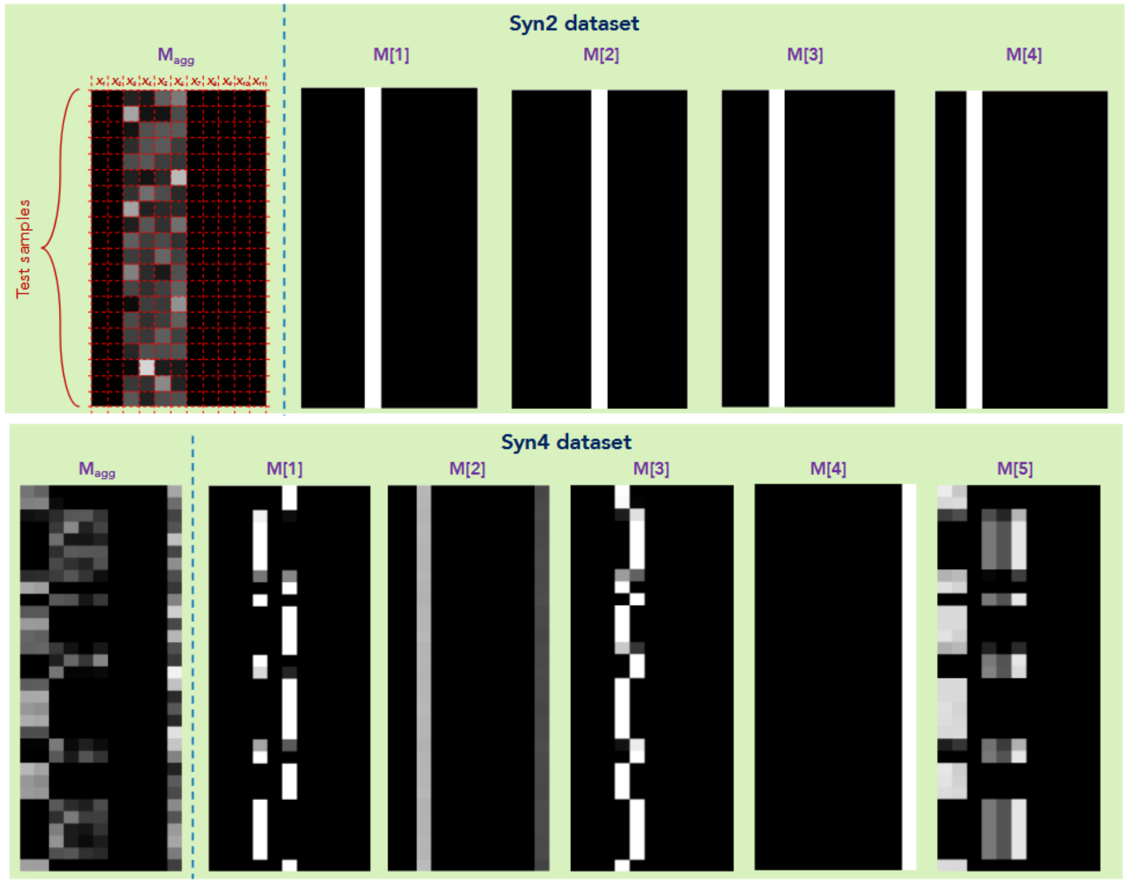
- The aggregate decision contribution at i-th decision step for the b-th sample.

$$\eta_b[i] = \sum_{c=1}^{N_d} ReLU(d_{b,c}[i])$$

- The aggregate feature importance mask

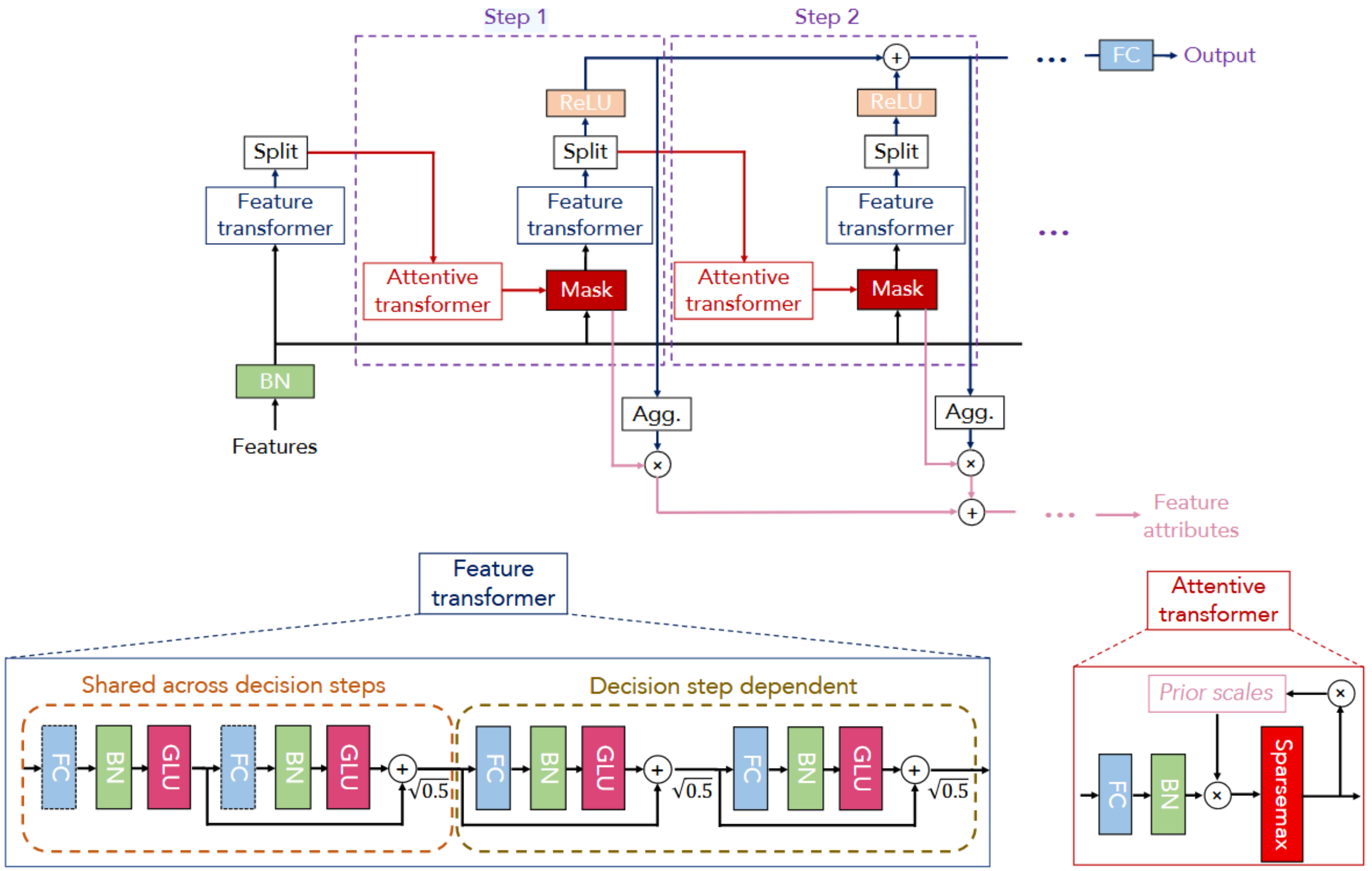
$$M_{agg-b,j} = \frac{\sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i]}{\sum_{j=1}^{D} \sum_{i=1}^{N_{steps}} \eta_b[i] M_{b,j}[i]}$$

Martins A, Astudillo R. From softmax to sparsemax: A sparse model of attention and multi-label classification[C]//International conference on machine learning. PMLR, 2016: 1614-1623.



Feature importance masks and the aggregate feature importance mask.

Structure of TabNet



Arik S Ö, Pfister T. Tabnet: Attentive interpretable tabular learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2021, 35(8): 6679-6687.

Neural network construction

1. Tree-mimic Structure
2. **Transformer-based Structure**
3. Regularization-based Structure

Transformer

- Sequence2Sequence
  - language translation
  - image captioning
  - text summarization
- Uses self-attention mechanisms to process sequential data.
  - Encoder-decoder structure
  - Attention Mechanism / Multi-Head Attention
  - Positional Encoding

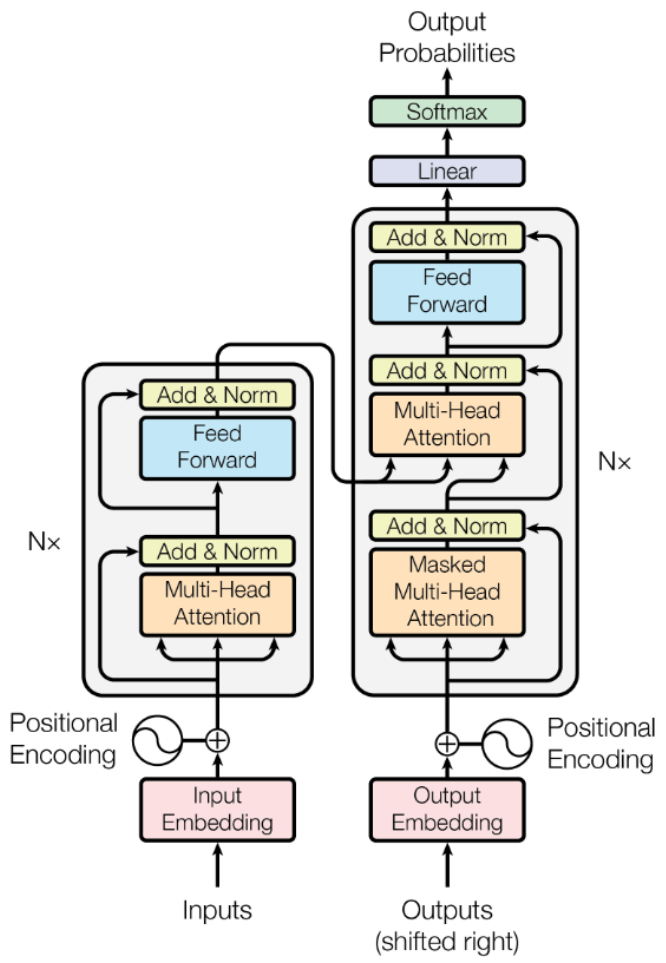
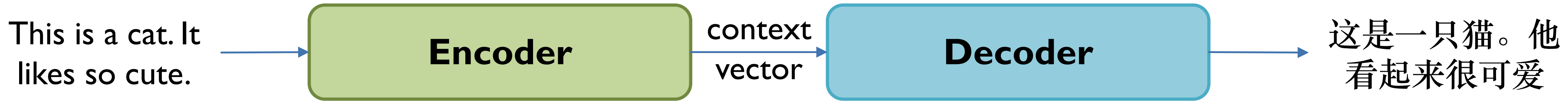


Figure 1: The Transformer - model architecture.

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

Encoder-Decoder structure

- Encoder:
  - Takes input token(word) by token and generates a set of numbers which is called context vector
- Decoder:
  - Takes context vector, process on it and generates output token by token



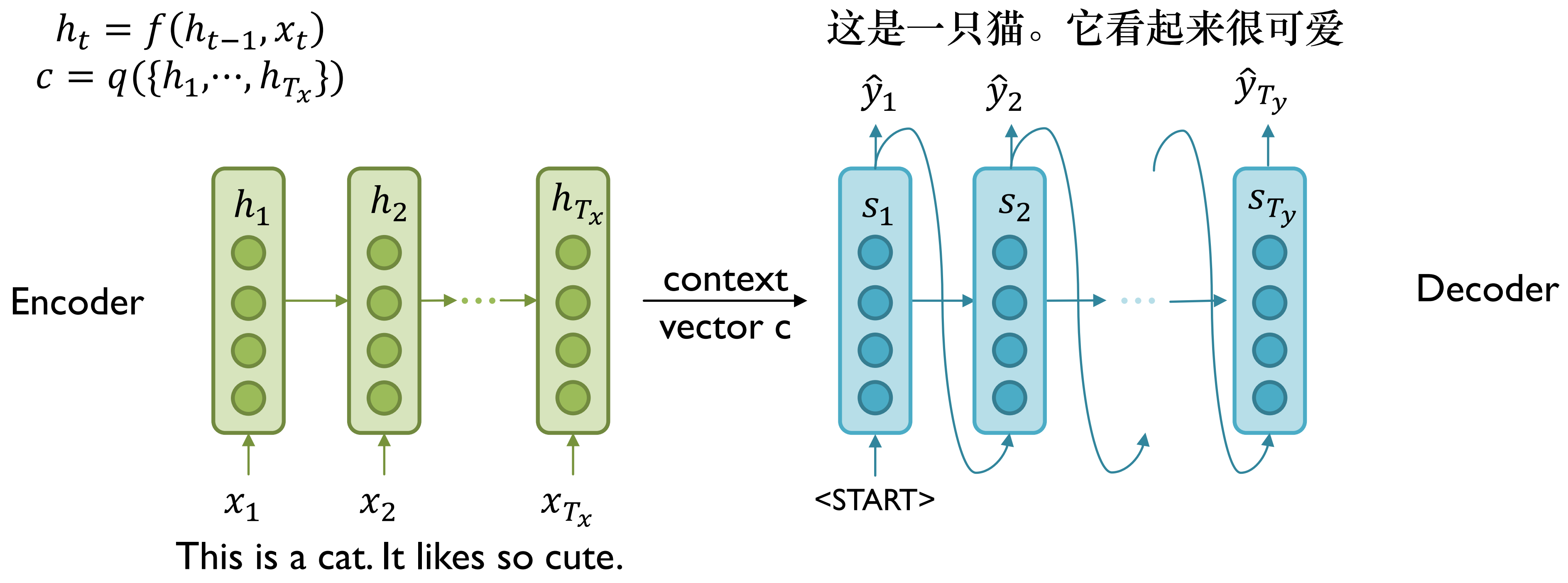
An example of Encoder-Decoder structure

Encoder-Decoder structure

- Take unidirectional RNN as an example:

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

$$h_t = f(h_{t-1}, x_t)$$
$$c = q(\{h_1, \cdots, h_{T_x}\})$$



Encoder-Decoder structure

- Encoder:
  - A stack of N=6 identical layers, each consisting of **two** sub-layers.
    1. Multi-head self-attention mechanism
    2. Position-wise fully connected feed-forward network
- Decoder:
  - A stack of N=6 identical layers, each consisting of **three** sub-layers.
    1. Masked multi-head self-attention mechanism
    2. Multi-head self-attention mechanism
    3. Position-wise fully connected feed-forward network
- residual connection + layer normalization

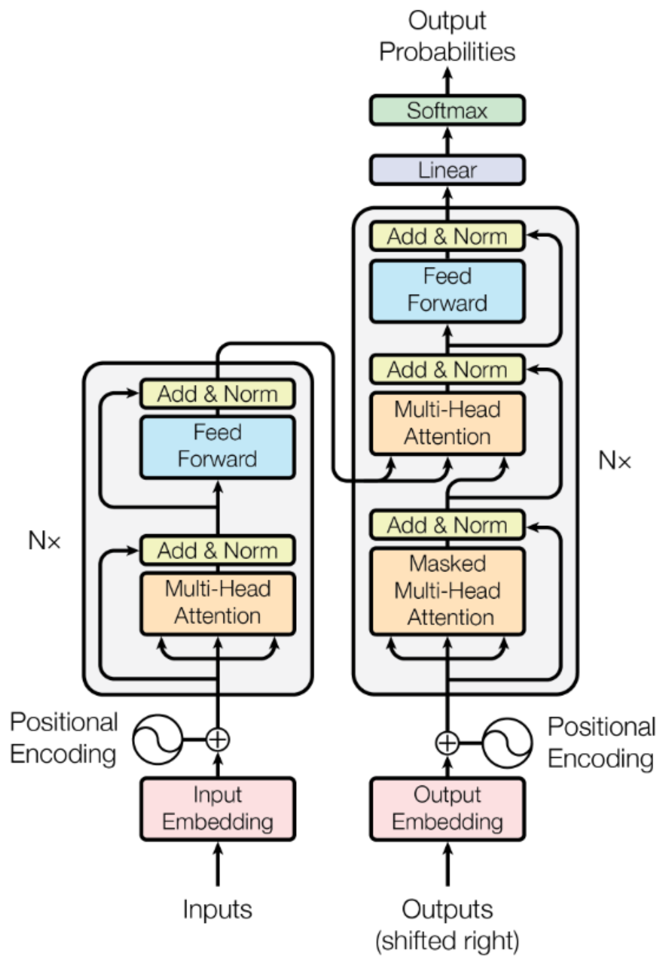
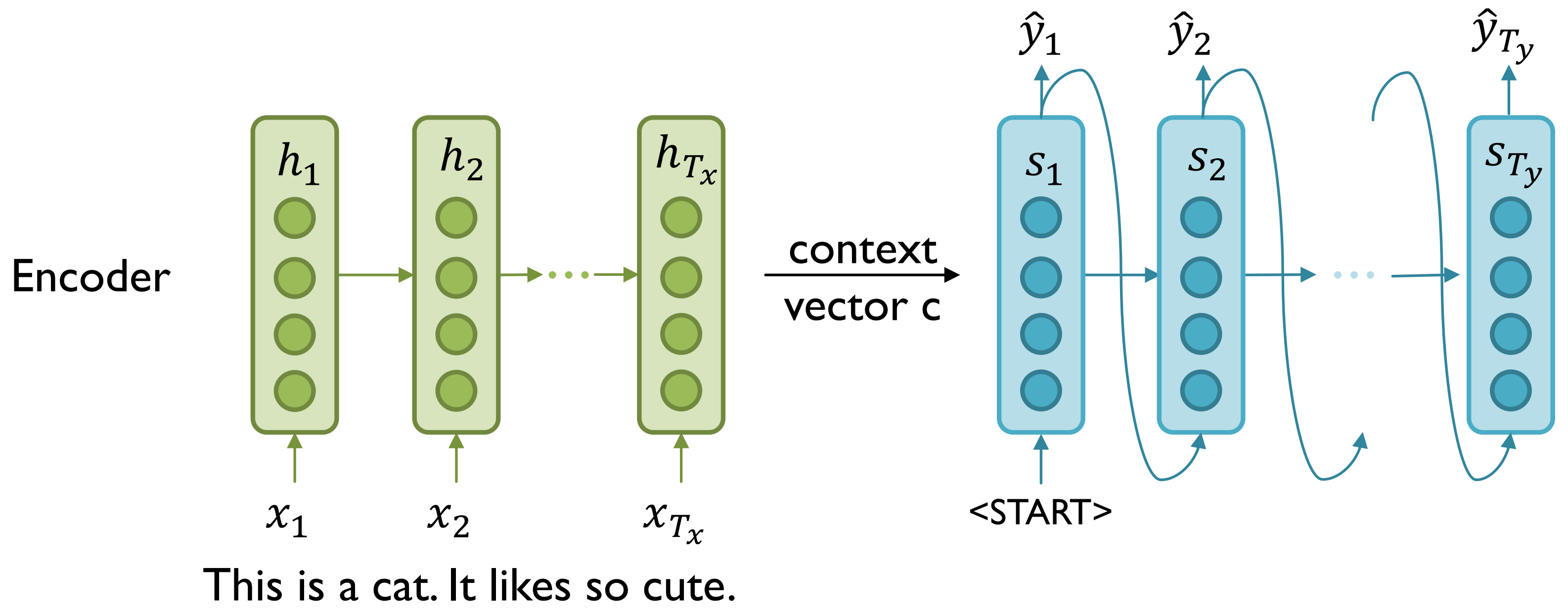


Figure 1: The Transformer - model architecture.

Encoder-Decoder structure

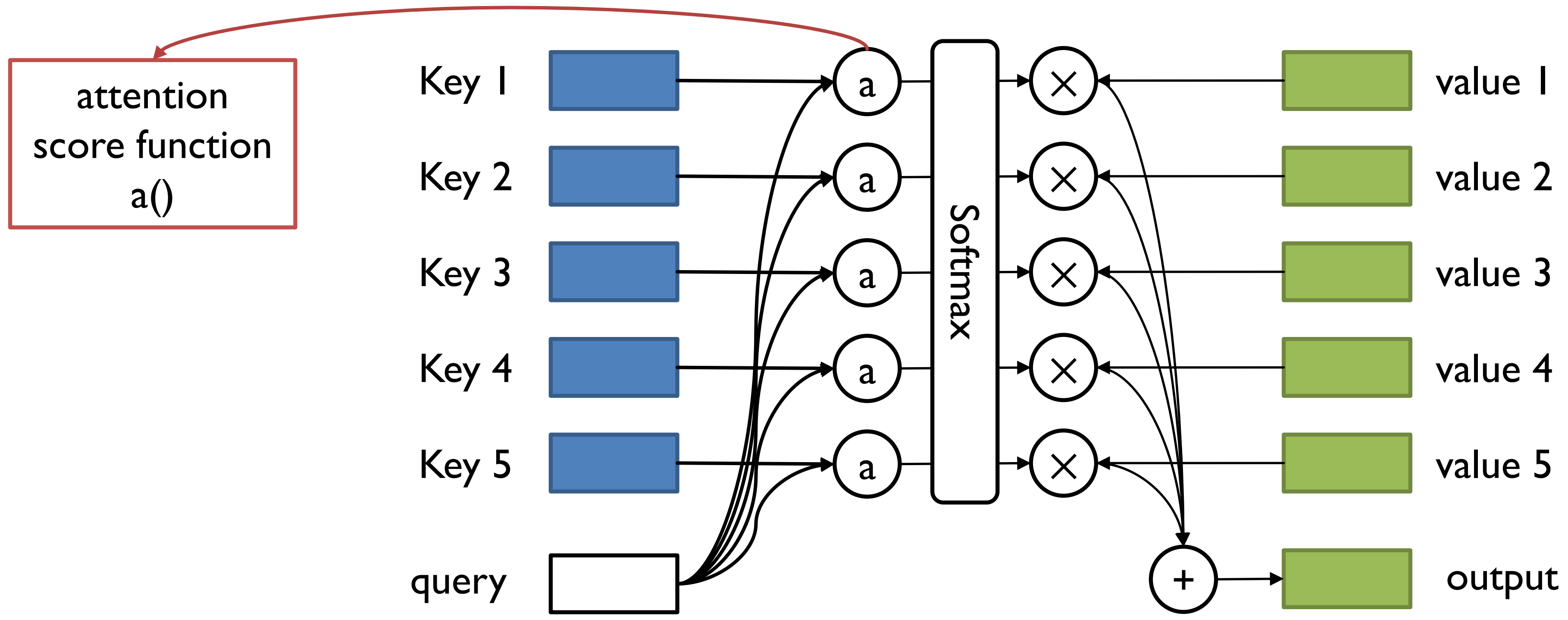
- Problem in Encoder-Decoder structure
  - Encoder: capturing complete context of a large sentence within a fixed-length context vector.
  - Decoder: in some cases the decoder only needs a particular word or group of words.



这是一只猫。
它看起来很可爱

What's Attention Mechanism?

- An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.



Why is the Attention Mechanism Useful?

- Selectively focus on relevant parts.
- Capture long-range dependencies.
- Improved interpretability.

How does the Attention Mechanism Work?

- No longer use only the last output context vector.
- Use an attention mechanism for all output states.

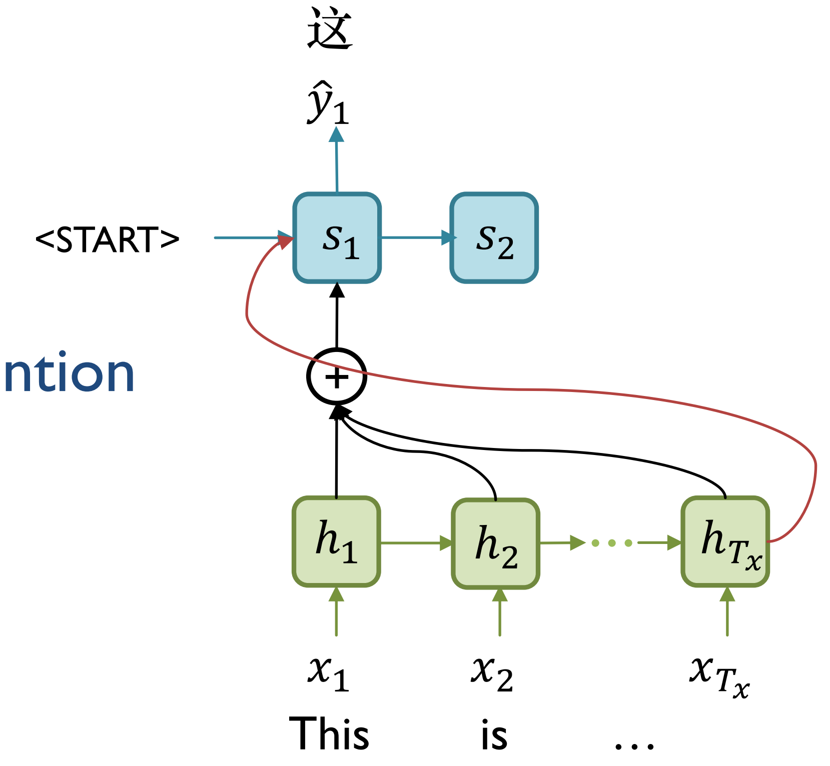
$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

$$\Downarrow$$

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c_i) = g(y_{t-1}, s_t, c_i)$$

- c_i represents the context vector aggregated by the attention mechanism. e_{ij} represents attention scores.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \qquad a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad e_{ij} = a(s_{i-1}, h_j)$$



Various Attention Mechanism

- Additive attention

$$a(q,k) = W_v^T \tanh\left(W_q q + W_k k\right) \in \mathbb{R}$$

$$W_q \in \mathbb{R}^{h\times q}, W_q \in \mathbb{R}^{h\times k}, W_v \in \mathbb{R}^{h}$$

  - The dimensions of q, k, v can be different.
  - It's like a multi-layer perceptron.

- Dot-product (multiplicative) attention

$$a(q,k) = q^T k \in \mathbb{R}$$

  - Can be implemented using highly optimized matrix multiplication code.

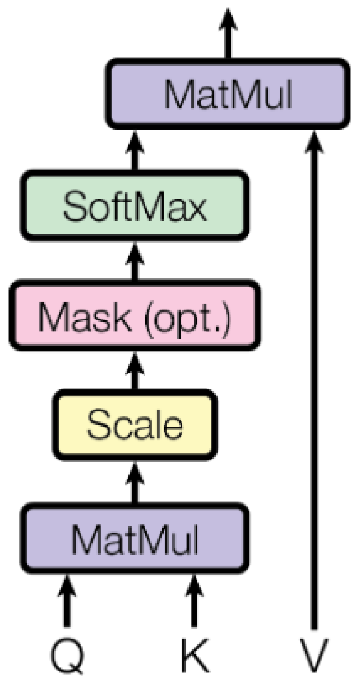
Scaled Dot-Product Attention

- Input:
  1. Q: queries of dimension d_k
  2. K: keys of dimension d_k
  3. V: values of dimension d_v
- Output:
  1. a weighted sum of the values

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\boxed{\sqrt{d_k}}}\right)V$$

$$\boxed{d_k \uparrow \quad \Rightarrow \quad \text{Dot product} \uparrow \quad \Rightarrow \quad \text{Gradient} \downarrow}$$

Scaled Dot-Product Attention



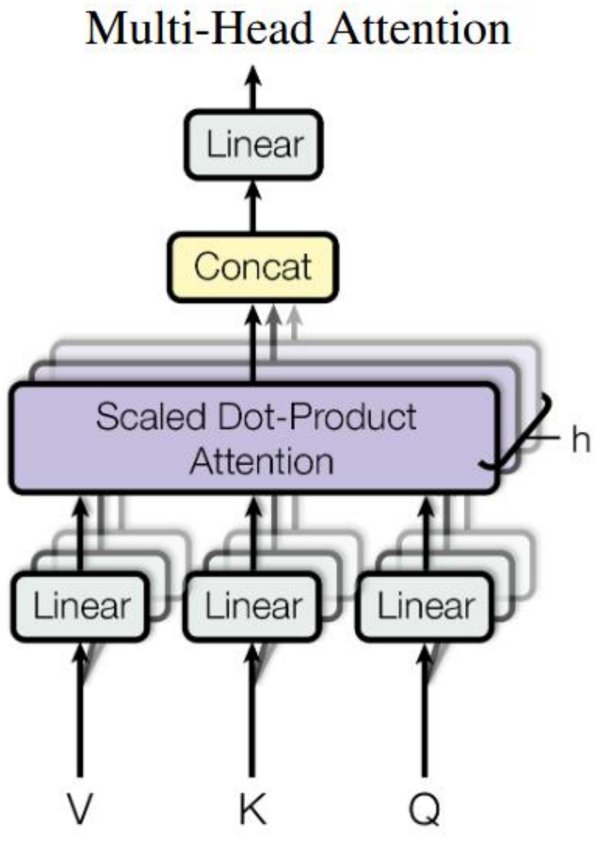
Multi-Head Attention

- Linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions.
- Perform the attention function in parallel.

$$MultiHead(Q, K, V, h) = Linear\left(Concat(head_1, \ldots, head_h)\right)$$

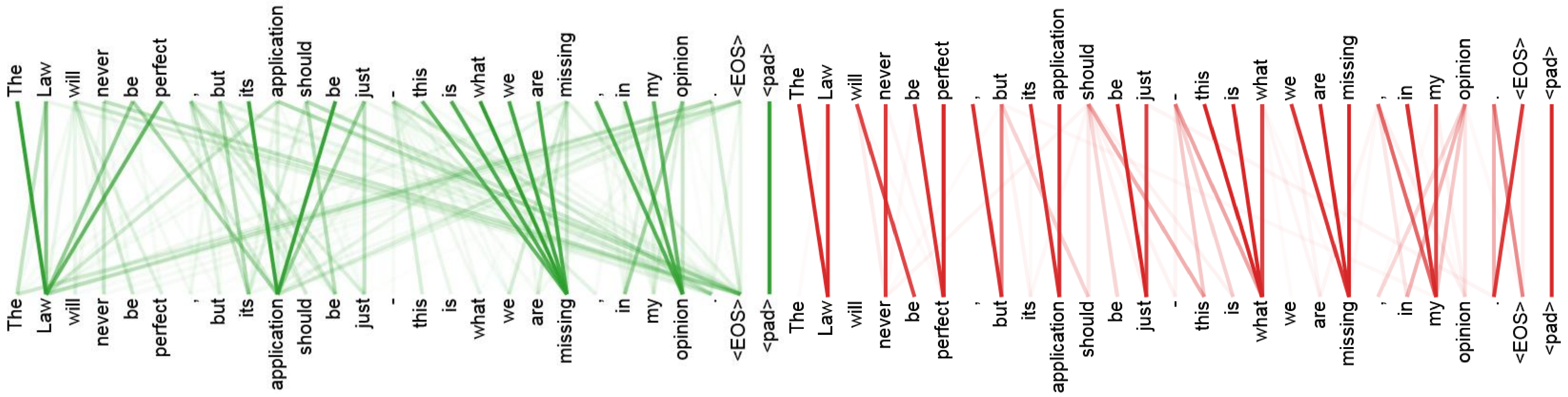
$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

$$W_i^Q \in \mathbb{R}^{d_{model}}, KW_i^K, VW_i^V$$



An Example of Multi-Head Attention

- Multi-head attention learn information from different representation subspaces at different positions.



Two different heads from the encoder self-attention at layer 5 of 6.

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

Positional Encoding

- The position and order of words are crucial to the meaning of a sentence.
- To make use of the order of the sequence.
- Sinusoidal Positional Encoding:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

- Composed of sine and cosine functions of different frequencies.

dimension	low	$\Rightarrow$	high
frequency	1	$\Rightarrow$	$\frac{1}{10000}$
wavelength	2π	$\Rightarrow$	$10000 \cdot 2\pi$

Sinusoidal Positional Encoding:

- The sine and cosine function is used to represent the absolute position, and the relative position is obtained by multiplying the two.
  - For any fixed offset k, $PE_{(pos+k,2i)}$ can be represented as a linear function of $PE_{(pos,2i)}$.

$$\begin{bmatrix} PE_{(pos+k,2i)} \\ PE_{(pos+k,2i+1)} \end{bmatrix} = \begin{bmatrix} u & v \\ -v & u \end{bmatrix} \begin{bmatrix} PE_{(pos,2i)} \\ PE_{(pos,2i+1)} \end{bmatrix}, \begin{cases} u = \cos(\omega_i \cdot k) \\ v = \sin(\omega_i \cdot k) \end{cases}, \omega_i = \frac{1}{10000^{\frac{2i}{d_{model}}}}$$

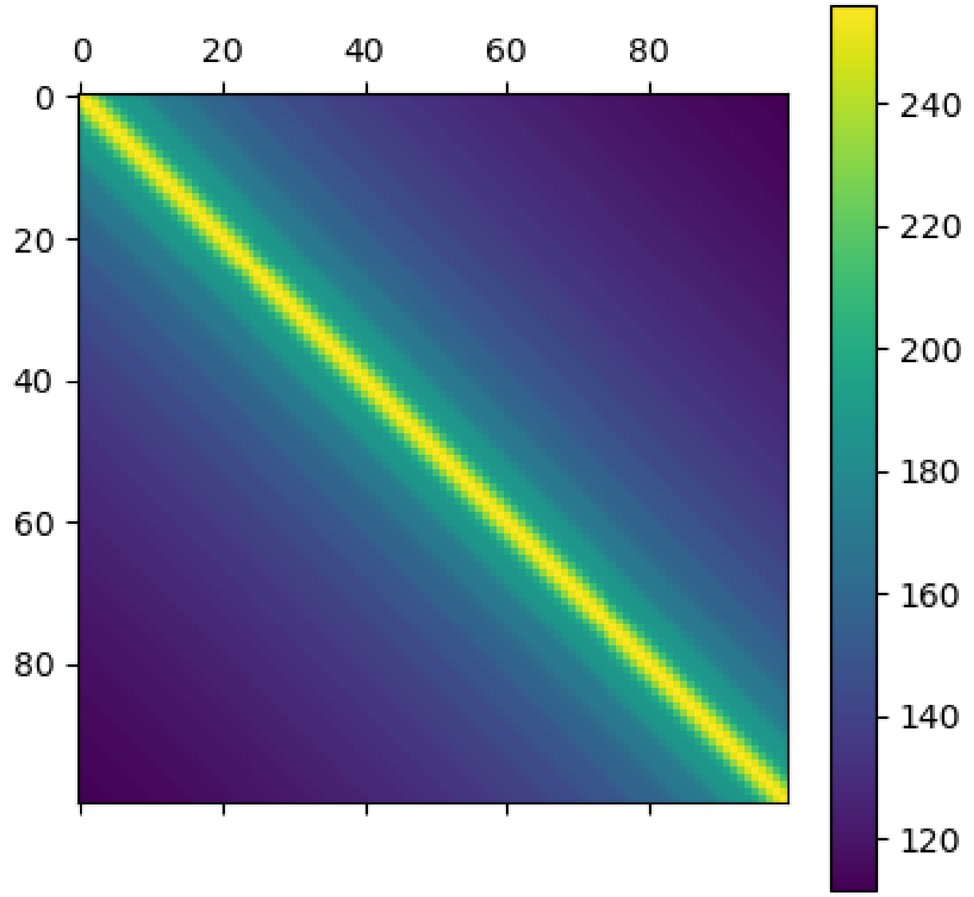
  - Compute the inner product of PE_{pos} and PE_{pos+k} to determine their relative position.

$$PE_{pos} \cdot PE_{pos+k} = \sum_{i=0}^{\frac{d}{2}-1} \sin(\omega_i pos) \cdot \sin[\omega_i(pos+k)] + \cos(\omega_i pos) \cdot \cos[\omega_i(pos+k)]$$

$$= \sum_{i=0}^{\frac{d}{2}-1} \cos(\omega_i k)$$

An Example of Relative Position

- Calculate the relative position of a hundred tokens, that is, the inner product between the two position codes.

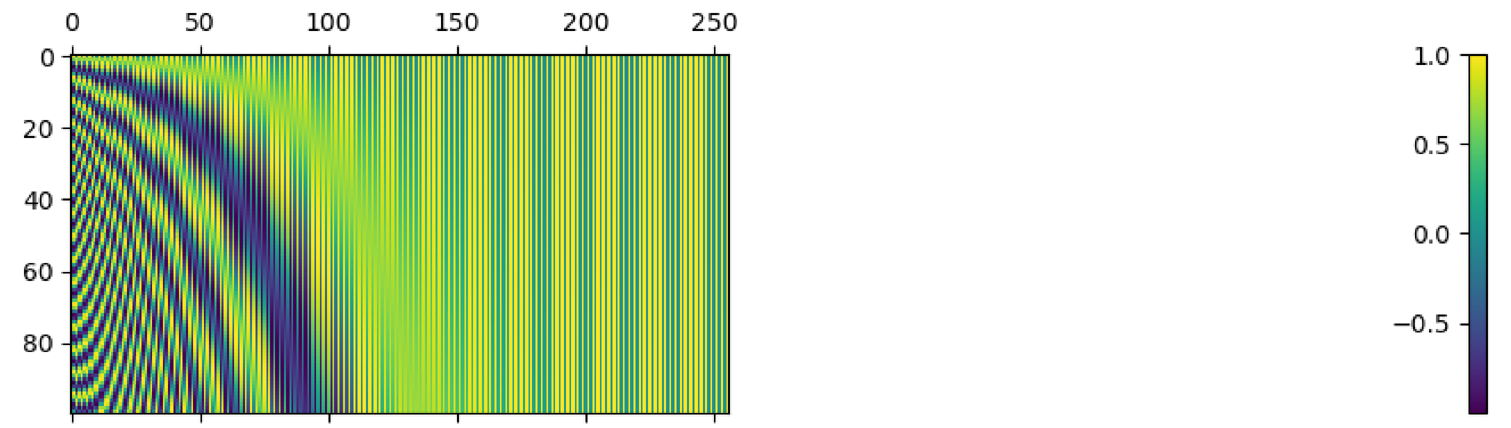


The greater the value, the closer the distance.

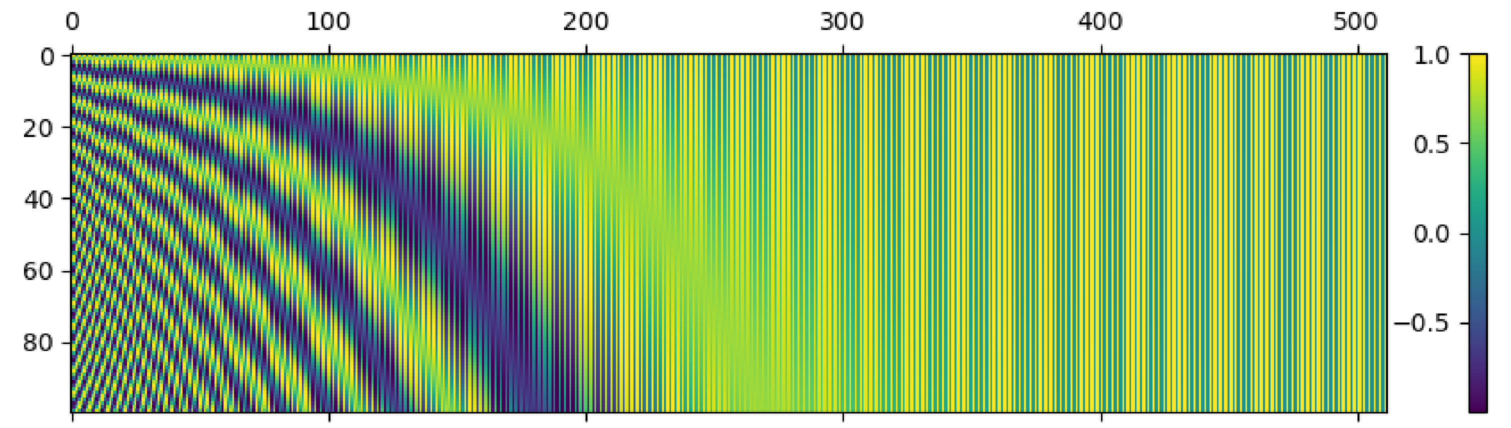
An Example of Positional Encoding

- Plot the coding situation of each location.

$d_{model} = 256$

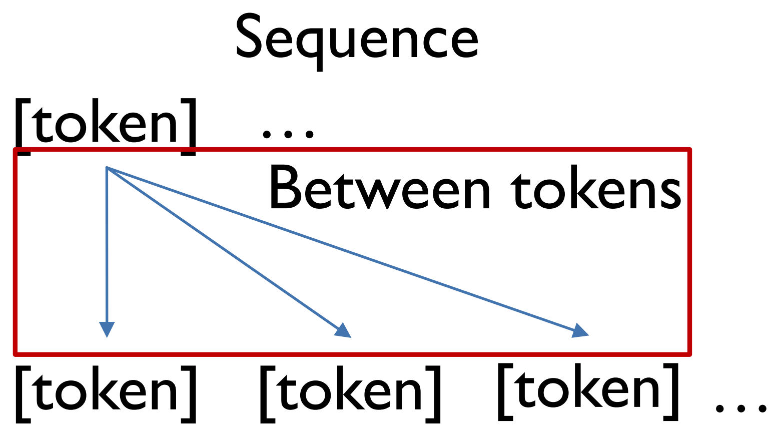


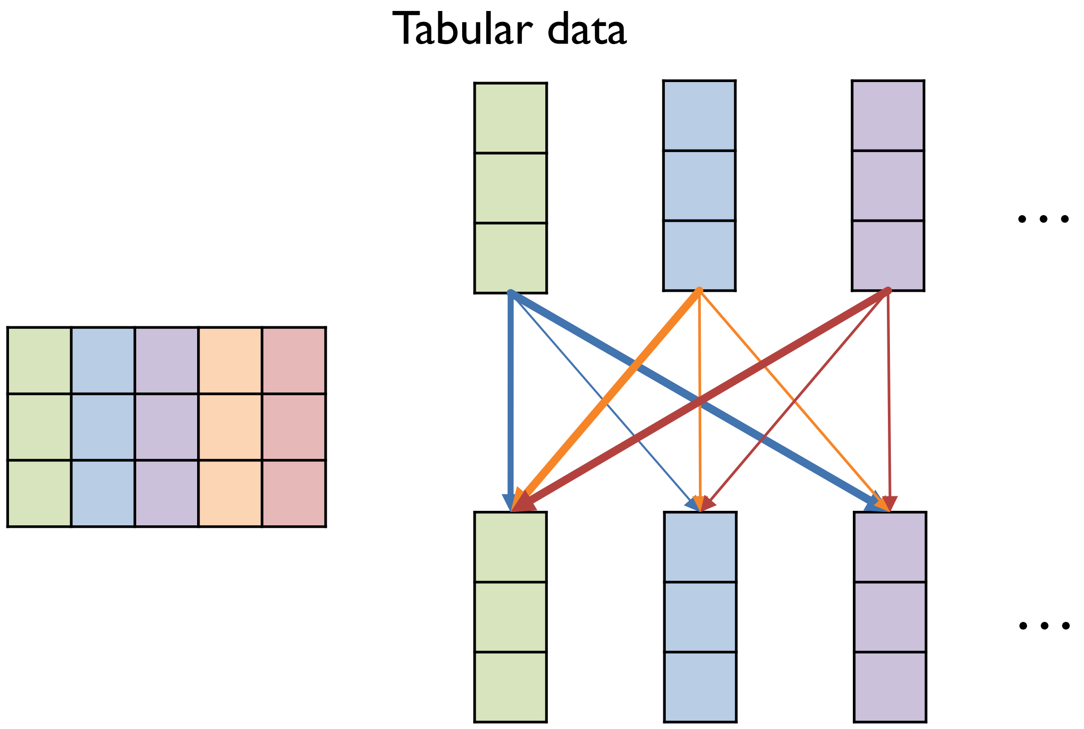
$d_{model} = 512$



Why can Transformer be used for tabular data?

- TNN's core purpose (Recap):
  - To explore the relationship between columns.
- Sequence and tabular data





Analogy between sequential data and tabular data.

Transformer on Tabular Data

- Intuition
  - The amount of attention that one column pays to another is the importance of the relationship between the two columns.
- Main idea
  - Calculate the attention between columns.
  - The attention score is the weight of the column over the other levels of relevance.

$$Relation_{\mathrm{col}} = Attention(Q, K, V) = Attention(XW^Q, XW^K, XW^V)$$

X is input embeddings.

FT-Transformer

- Feature Tokenizer + Transformer
  - Transforms all features (categorical and numerical) to embeddings.
  - A stack of Transformer layers.

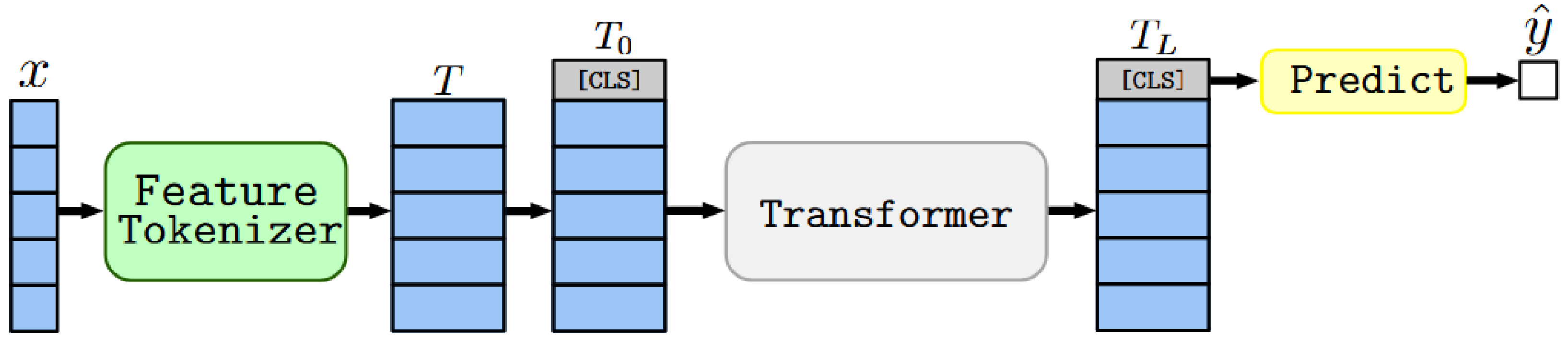


Figure 1: The FT-Transformer architecture. Firstly, Feature Tokenizer transforms features to embeddings. The embeddings are then processed by the Transformer module and the final representation of the [CLS] token is used for prediction.

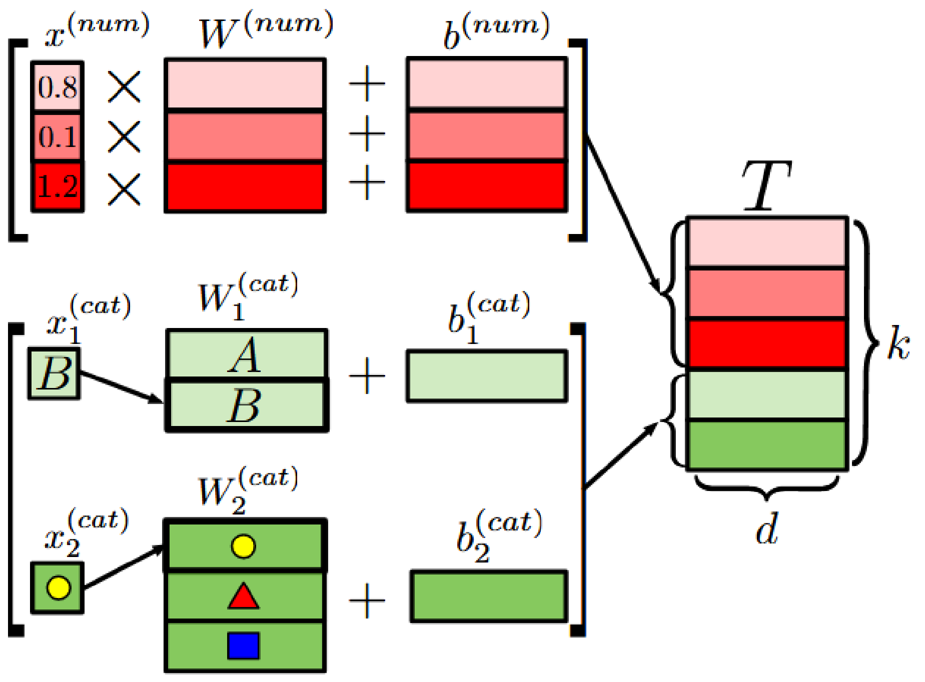
Feature Tokenizer

- Transforms the input features x to embeddings $T \in \mathbb{R}^{k\times d}$.

$$T_j = b_j + f_j(x_j) \in \mathbb{R}^d \qquad f_j : \mathbb{X}_j \to \mathbb{R}^d$$

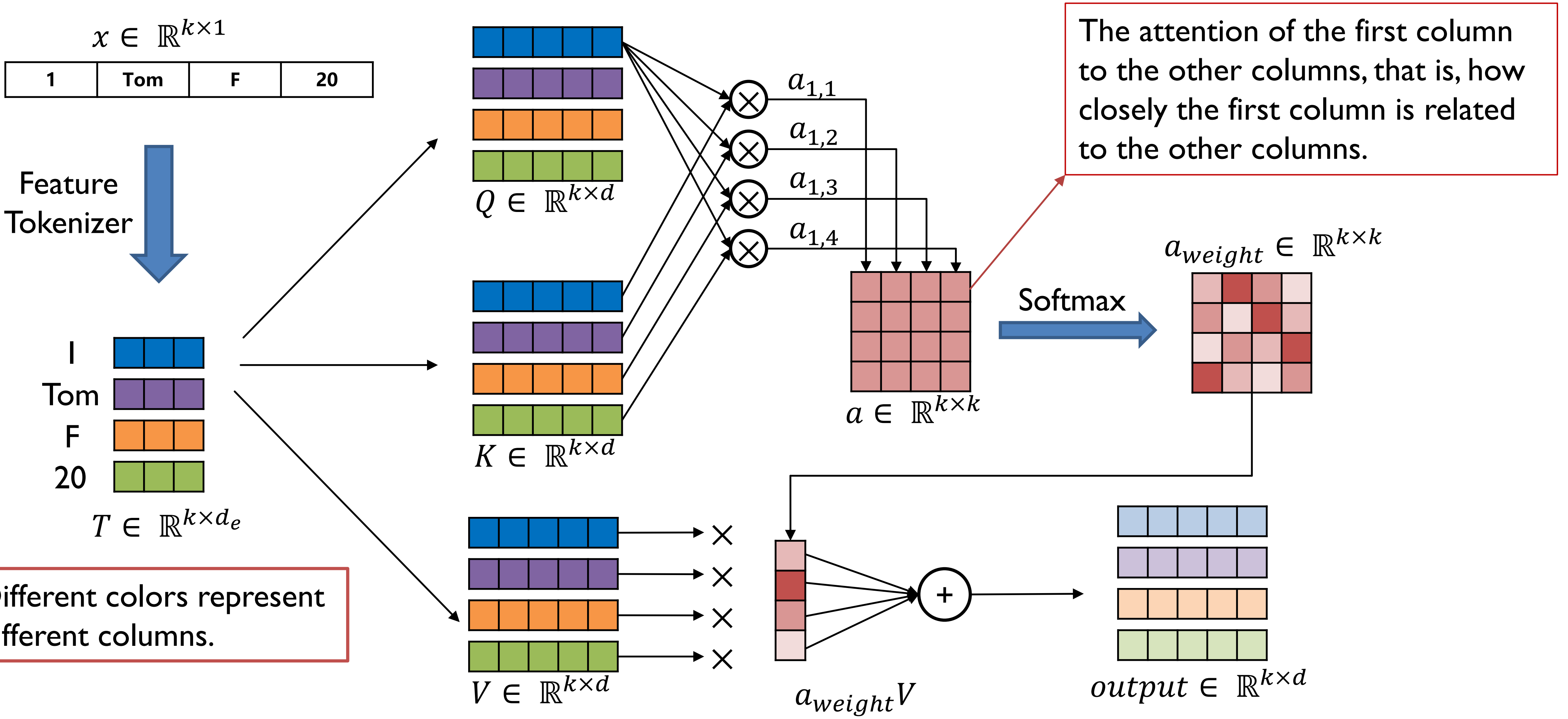
  - continuous features:
    - $f_j^{(num)}$ is implemented as the element-wise multiplication with the vector $W_j^{(num)} \in \mathbb{R}^d$
  - categorical features:
    - $f_j^{(cat)}$ is implemented as the lookup table $W_j^{(cat)} \in \mathbb{R}^{S_j\times d}$

$$T_j^{(num)} = b_j^{(num)} + x_j^{(num)} \cdot W_j^{(num)} \qquad \in \mathbb{R}^d,$$
$$T_j^{(cat)} = b_j^{(cat)} + e_j^T W_j^{(cat)} \qquad \in \mathbb{R}^d,$$
$$T = \mathtt{stack}\left[T_1^{(num)}, \ldots, T_{k^{(num)}}^{(num)}, T_1^{(cat)}, \ldots, T_{k^{(cat)}}^{(cat)}\right] \in \mathbb{R}^{k\times d}$$



Feature Tokenizer

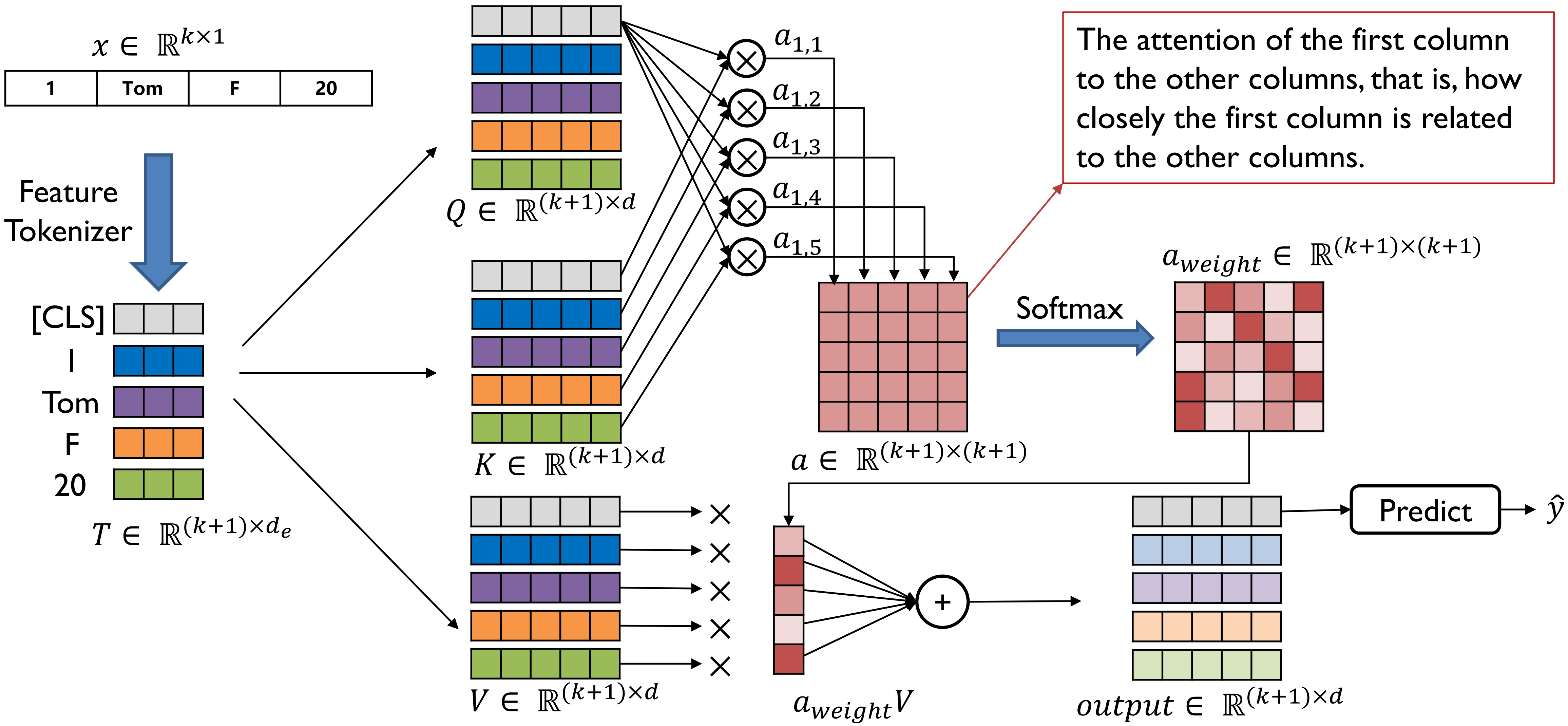
An Example of Single Sample.


$x \in \mathbb{R}^{k\times1}$
1
Tom
F
20
Feature Tokenizer
I
Tom
F
20
$T \in \mathbb{R}^{k\times d_e}$
$Q \in \mathbb{R}^{k\times d}$
$K \in \mathbb{R}^{k\times d}$
$V \in \mathbb{R}^{k\times d}$
$a_{1,1}$
$a_{1,2}$
$a_{1,3}$
$a_{1,4}$
$a \in \mathbb{R}^{k\times k}$
Softmax
$a_{weight} \in \mathbb{R}^{k\times k}$
$a_{weight}V$
$output \in \mathbb{R}^{k\times d}$

The attention of the first column to the other columns, that is, how closely the first column is related to the other columns.

Different colors represent different columns.

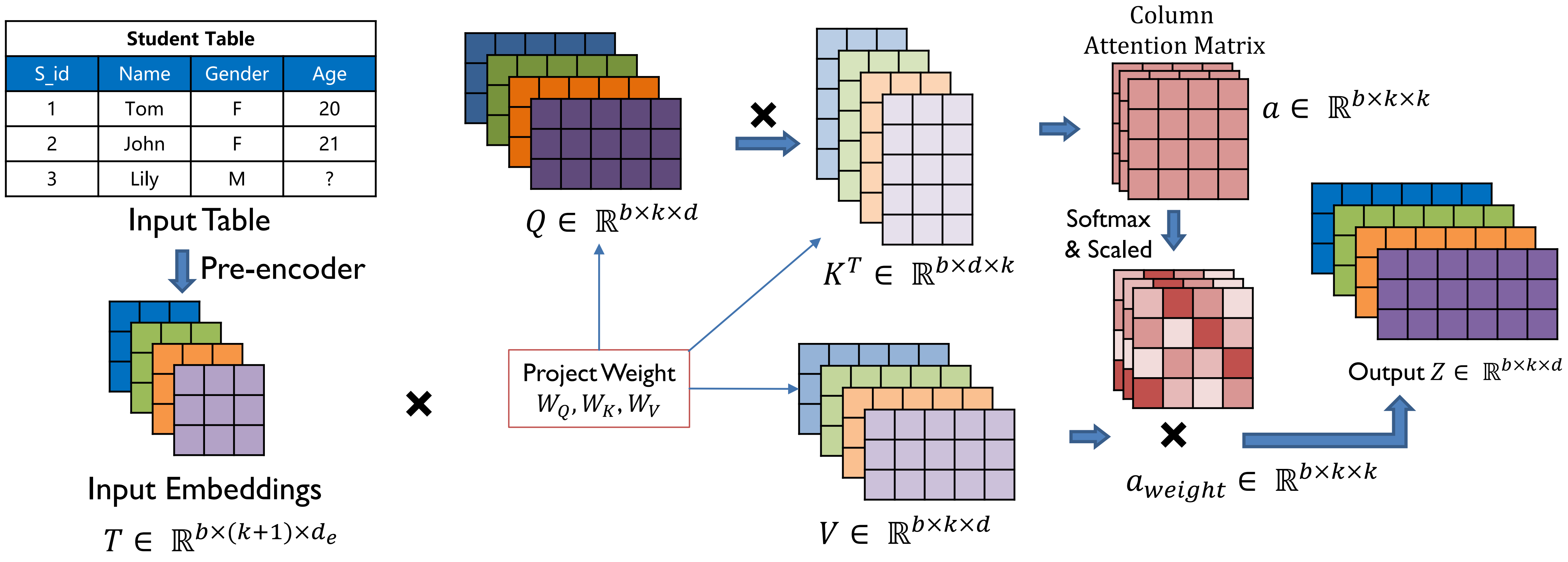
An Example of Single Sample with CLS.



$x \in \mathbb{R}^{k\times 1}$
1
Tom
F
20
Feature Tokenizer
[CLS]
I
Tom
F
20
$T \in \mathbb{R}^{(k+1)\times d_e}$
$Q \in \mathbb{R}^{(k+1)\times d}$
$K \in \mathbb{R}^{(k+1)\times d}$
$V \in \mathbb{R}^{(k+1)\times d}$
$a_{1,1}$
$a_{1,2}$
$a_{1,3}$
$a_{1,4}$
$a_{1,5}$
$a \in \mathbb{R}^{(k+1)\times(k+1)}$
Softmax
$a_{weight} \in \mathbb{R}^{(k+1)\times(k+1)}$
$a_{weight}V$
$output \in \mathbb{R}^{(k+1)\times d}$
Predict
$\hat{y}$

The attention of the first column to the other columns, that is, how closely the first column is related to the other columns.

An Example of Multiple Samples.

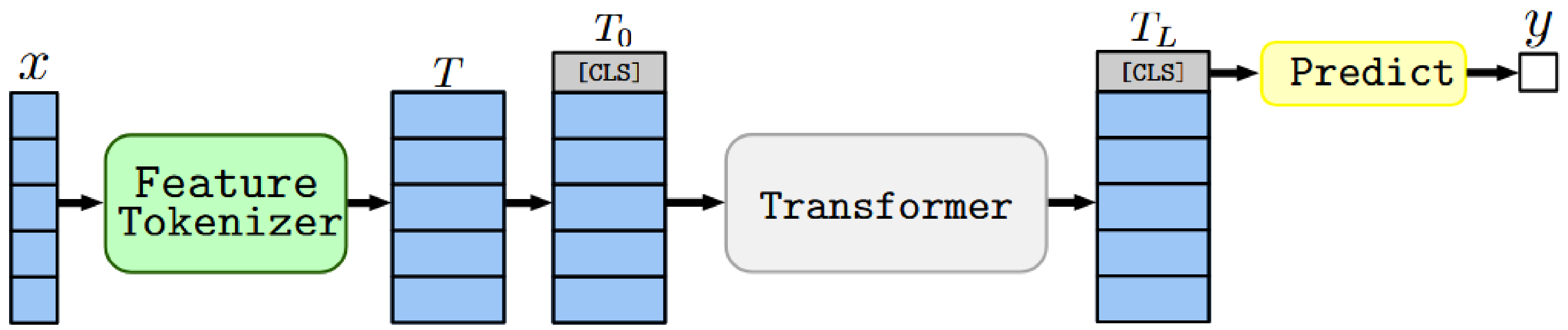


Student Table			
S_id	Name	Gender	Age
1	Tom	F	20
2	John	F	21
3	Lily	M	?

Input Table
Pre-encoder
Input Embeddings
$T \in \mathbb{R}^{b\times(k+1)\times d_e}$
$Q \in \mathbb{R}^{b\times k\times d}$
Project Weight
W_Q, W_K, W_V
$K^T \in \mathbb{R}^{b\times d\times k}$
$V \in \mathbb{R}^{b\times k\times d}$
Column Attention Matrix
$a \in \mathbb{R}^{b\times k\times k}$
Softmax & Scaled
$a_{weight} \in \mathbb{R}^{b\times k\times k}$
Output $Z \in \mathbb{R}^{b\times k\times d}$

Various Transformer-based Methods

- FT-Transformer: Feature Tokenizer + Transformer



- Other Transformer-based methods:
  - AutoInt
  - TabTransformer
  - Excelformer
  - …

Song W, Shi C, Xiao Z, et al. Autoint: Automatic feature interaction learning via self-attentive neural networks[C]//Proceedings of the 28th ACM international conference on information and knowledge management. 2019: 1161-1170.
Huang X, Khetan A, Cvitkovic M, et al. Tabtransformer: Tabular data modeling using contextual embeddings[J]. arXiv preprint arXiv:2012.06678, 2020.
Chen J, Yan J, Chen Q, et al. Excelformer: A neural network surpassing gbdts on tabular data[J]. arXiv preprint arXiv:2301.02819, 2023.

Regularization-based Structure

- Regularization methods are techniques used in machine learning to prevent overfitting and improve the model's generalization
- Essentially, limit the model's complexity.
- Main idea
  1. data augmentation
  2. network architecture choices
  3. penalty terms that explicitly influence parameter learning

TANGOS

- Based on regularizing neuron attributions.
- Attribution methods work by using gradient signals to evaluate the contributions of the input features.
- Two aspects:
  1. Specialization
  2. Orthogonalization

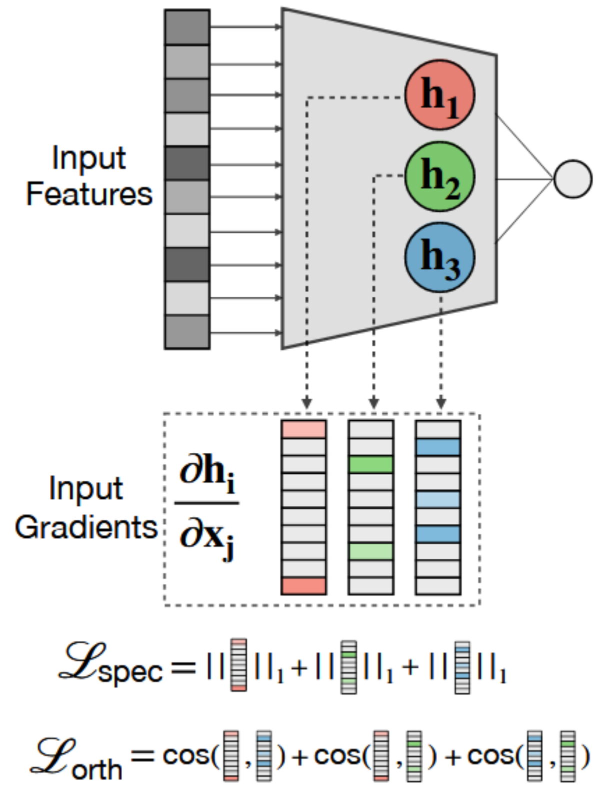

Input Features
h_1
h_2
h_3
Input Gradients
$\frac{\partial h_i}{\partial x_j}$
$\mathscr{L}_{spec} = || \; ||_1 + || \; ||_1 + || \; ||_1$
$\mathscr{L}_{orth} = \cos(\; , \;) + \cos(\; , \;) + \cos(\; , \;)$

Figure 2: **Method illustration.** TANGOS regularizes the gradients with respect to each of the latent units.

Jeffares A, Liu T, Crabbé J, et al. TANGOS: Regularizing tabular neural networks through gradien
orthogonalization and specialization[J]. arXiv preprint arXiv:2303.05506, 2023.

Neuron Attributions

- Predictive function f using function composition

$$f = l \circ g$$

  - $g : X \to H$ maps the input to a representation $h = g(x) \in H$
  - $H \subseteq \mathbb{R}^{d_H}$ is a d_H dimensional latent space.

- Use gradient signals to evaluate the contributions of the input features.

$$a_j^i(x) \equiv \frac{\partial h_i(x)}{\partial x_j}$$

  - $a_j^i(x) \in R$ to denote the attribution of the i-th neuron w.r.t. the feature x_j

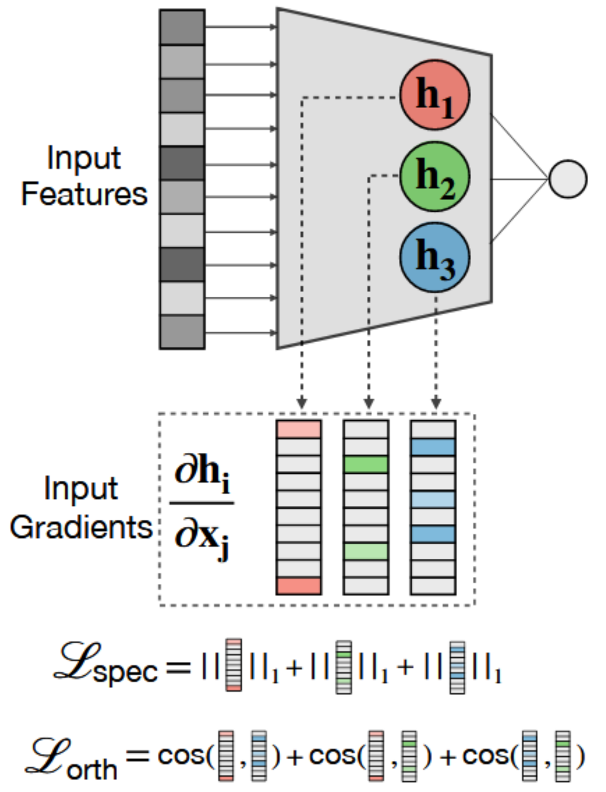


Figure 2: **Method illustration.** TANGOS regularizes the gradients with respect to each of the latent units.

Specialization

- The contribution of input features to the activation of a particular neuron should be sparse.
- a few features should account for a large percentage of total attributions.

$$\mathcal{L}_{\text{spec}}(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{d_H}\sum_{i=1}^{d_H}\|a^i(x_b)\|_1$$

$$||\mathbf{x}||_p = (\sum_{i=1}^{N}|x_i|^p)^{\frac{1}{p}}$$

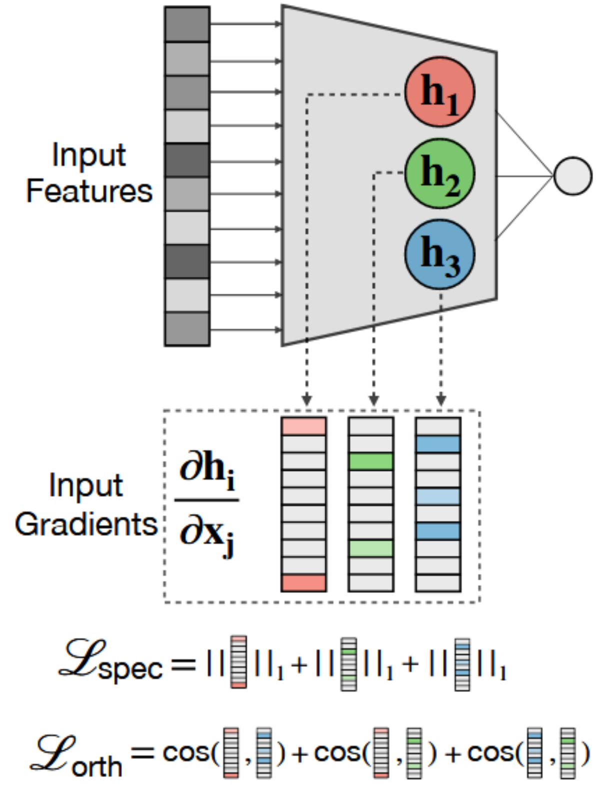

Input Features
h_1
h_2
h_3
Input Gradients
$\frac{\partial h_i}{\partial x_j}$
$\mathcal{L}_{\text{spec}} = ||\ ||_1 + ||\ ||_1 + ||\ ||_1$
$\mathcal{L}_{\text{orth}} = \cos(\ ,\ ) + \cos(\ ,\ ) + \cos(\ ,\ )$

Figure 2: **Method illustration.** TANGOS regularizes the gradients with respect to each of the latent units.

Orthogonalization

- Different neurons should attend to non-overlapping subsets of input features given a particular input sample.
- Penalize the correlation between neuron attributions.

$$\mathcal{L}_{\mathrm{orth}}(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{C}\sum_{i=2}^{d_H}\sum_{j=1}^{i-1}\rho\left[a^i(x_b), a^j(x_b)\right]$$

$$C = \frac{d_H\cdot(d_H-1)}{2}$$

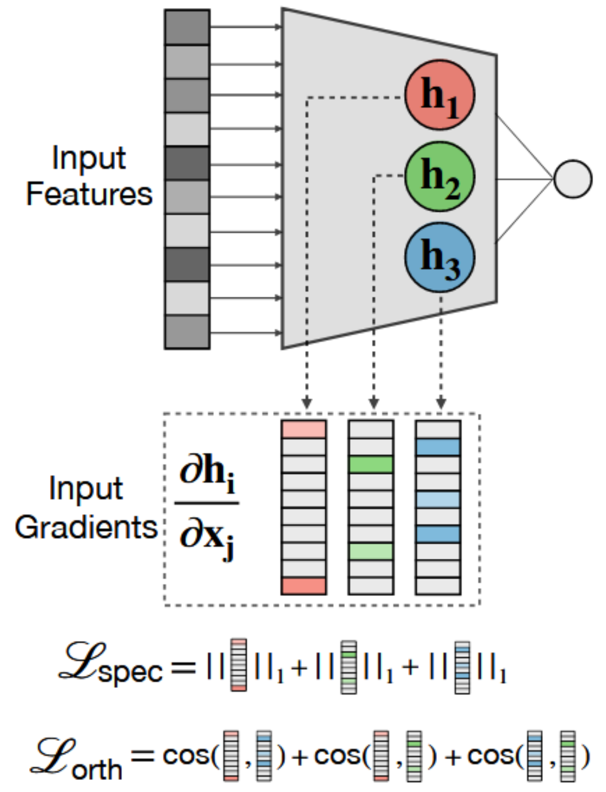

Input Features
h_1
h_2
h_3
Input Gradients
$\frac{\partial h_i}{\partial x_j}$
$\mathcal{L}_{\mathrm{spec}} = \|\ \|_1 + \|\ \|_1 + \|\ \|_1$
$\mathcal{L}_{\mathrm{orth}} = \cos(\ ,\ ) + \cos(\ ,\ ) + \cos(\ ,\ )$

Figure 2: **Method illustration.** TANGOS regularizes the gradients with respect to each of the latent units.

Jeffares A, Liu T, Crabbé J, et al. TANGOS: Regularizing tabular neural networks through gradien
orthogonalization and specialization[J]. arXiv preprint arXiv:2303.05506, 2023.

TANGOS Regularizer

$$\mathcal{R}_{\texttt{TANGOS}}(x) = \lambda_1 \mathcal{L}_{\text{spec}}(x) + \lambda_2 \mathcal{L}_{\text{orth}}(x)$$

$$\mathcal{L}_{\text{spec}}(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{d_H}\sum_{i=1}^{d_H}\|a^i(x_b)\|_1$$

$$\mathcal{L}_{\text{orth}}(x) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{C}\sum_{i=2}^{d_H}\sum_{j=1}^{i-1}\rho\left[a^i(x_b), a^j(x_b)\right]$$

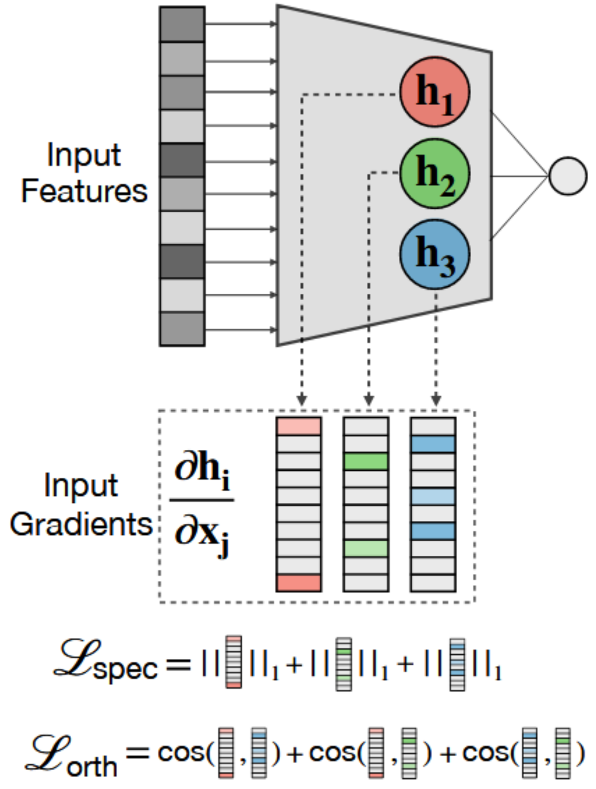

Input Features
$\mathbf{h_1}$
$\mathbf{h_2}$
$\mathbf{h_3}$
Input Gradients
$\frac{\partial \mathbf{h_i}}{\partial \mathbf{x_j}}$
$\mathscr{L}_{\text{spec}} = \|\;\|_1 + \|\;\|_1 + \|\;\|_1$
$\mathscr{L}_{\text{orth}} = \cos(\;,\;) + \cos(\;,\;) + \cos(\;,\;)$

Figure 2: **Method illustration.** TANGOS regularizes the gradients with respect to each of the latent units.

Jeffares A, Liu T, Crabbé J, et al. TANGOS: Regularizing tabular neural networks through gradien
orthogonalization and specialization[J]. arXiv preprint arXiv:2303.05506, 2023.

Thanks for your time.

QA.