Well no, not these kind of transformers…

Transformers

Professor Marie Roch

for details on transformers, see chapter 10 in draft:

Jurafsky, D., and Martin, J. H. (2023). *Speech and Language Processing (Pearson Prentice Hall, Upper Saddle River, NJ)*

Transformers

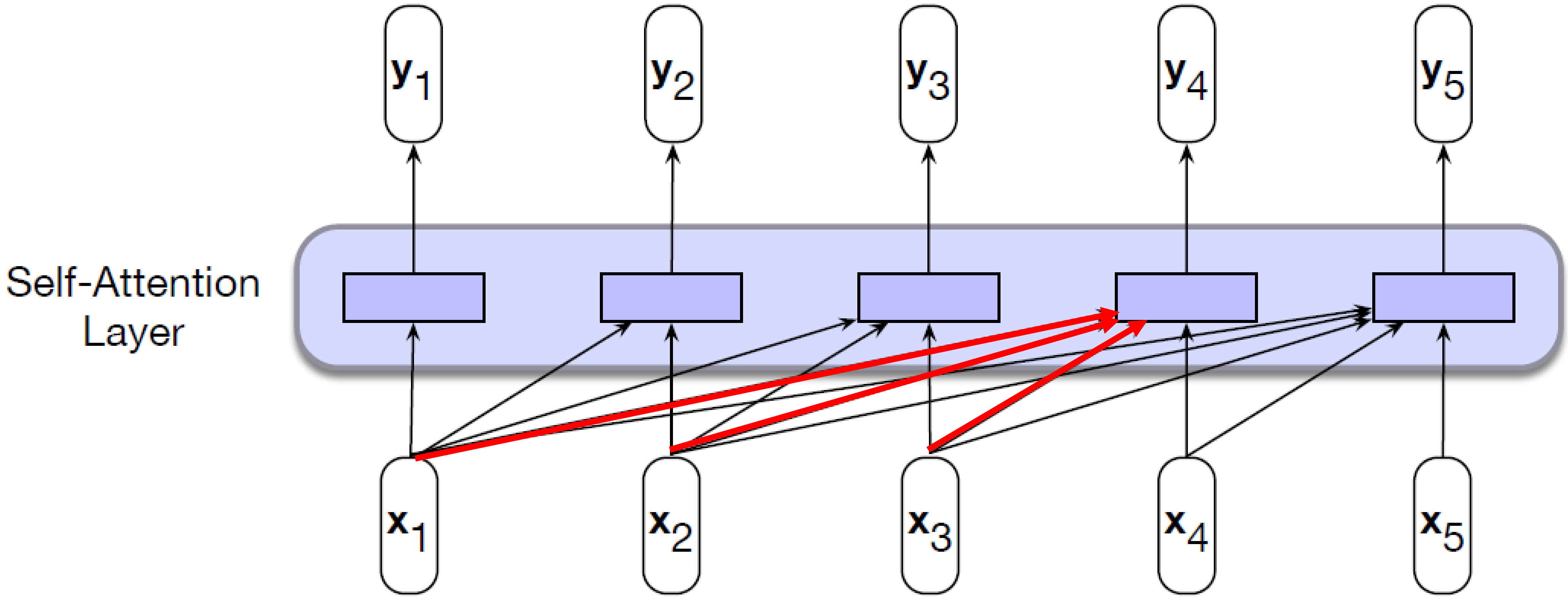
- Architecture which processes blocks of inputs:

$$[x_1, x_2, \ldots, x_n] \to [y_1, y_2, \ldots, y_n]$$

- Each mapping, $x_i \to y_i$, is
  - computed independently of $x_j \to y_j$ $(j \neq i)$
  - has access to other inputs x_j
  - is said to be *causal* if $x_i \to y_i$ only has access to inputs $\mathrm{x_j}: j \leq i$ (*non-causal* if we can access future inputs)
- The access to other inputs is used in a self-attention mechanism (self-attention, as the attention is within the current block)

SAN DIEGO STATE UNIVERSITY

Causal self-attention layer



$x_4 \rightarrow y_4$ can attend to the values of x_1, x_2, x_3

J&M 2023, Fig 10.1

San Diego State University

How do we attend to other inputs?

- Simplest attention mechanism is the dot product
  - $score(x_i, x_j) = x_i \cdot x_j$
  - remember: $x_i \cdot x_j = |x_i||x_j|\cos(\theta)$ where $\theta = \measuredangle x_i x_j$,
    ∴ larger score implies vector similarity
- We can transform the scores to a distribution

$$\alpha_{i,j} = \mathrm{softmax}\left(\mathrm{score}(x_i, x_j)\right)$$

- Output in this simple mechanism (we will do more)

$$y_i = \sum_{j=1}^{N} \alpha_{i,j} x_j \text{ or if causal } y_i = \sum_{j=1}^{i} \alpha_{i,j} x_j$$

San Diego State University

Key ideas so far

- For each input, we estimate a distribution indicating the relevance of neighboring inputs including the current one.
- The output is a linear combination of the inputs scaled by their importance.
- Attention can be used anywhere in the network, so the inputs are likely to be some type of feature representation.

- From here on, we will use causal or non-causal examples with the understanding that the other can be easily derived.

Building on this idea

Input $x_1, x_2, \dots, x_N$ will play a variety of roles in the prediction of y_i

- Value $x_{1 \le j \le N}$: will weight these values to compute y_i.
- Query and Key provide the importance of each Value
  - Query x_i: focus of attention mechanism.
  - Key $x_{1 \le j \le N}$: vector to which we compare the query

$$y_i = \sum_{j=1}^{N} \mathrm{softmax}(x_i \cdot x_j) x_j$$

query

value

Note: We are showing roles here, there is one more step before we compute actual key, query, value

Query, key, and value

- Weight vectors learn to appropriately transform inputs for their role
  - q_i, k_i, v_i: $\quad q_i = W^Q x_i \quad k_i = W^K x_i \quad v_i = W^V x_i$
  - Learned by standard backpropagation

- Dimension of W matrices
  - For now, we define $d_k = d_v = d$ each W is $d \times d$ where d is the dimension of the observation x_i:
    $$(d \times d)(d \times 1) \rightarrow (d \times 1)$$
  - Later, we will introduce multi-headed attention
    - Will let us learn multiple attention representations.
    - Will permit query & key vectors of length d_k and value of length d_v where $d_k \neq d_v \neq d$

SAN DIEGO STATE UNIVERSITY

Query, key, and value

- Scores can now be thought of as the dot product of a query and key:

$$\alpha_{i,j} = score\left(x_i, x_j\right) = (W^Q x_i) \cdot \left(W^K x_j\right) = q_i \cdot k_j$$

as these can be quite large, we normalize by the input dimension

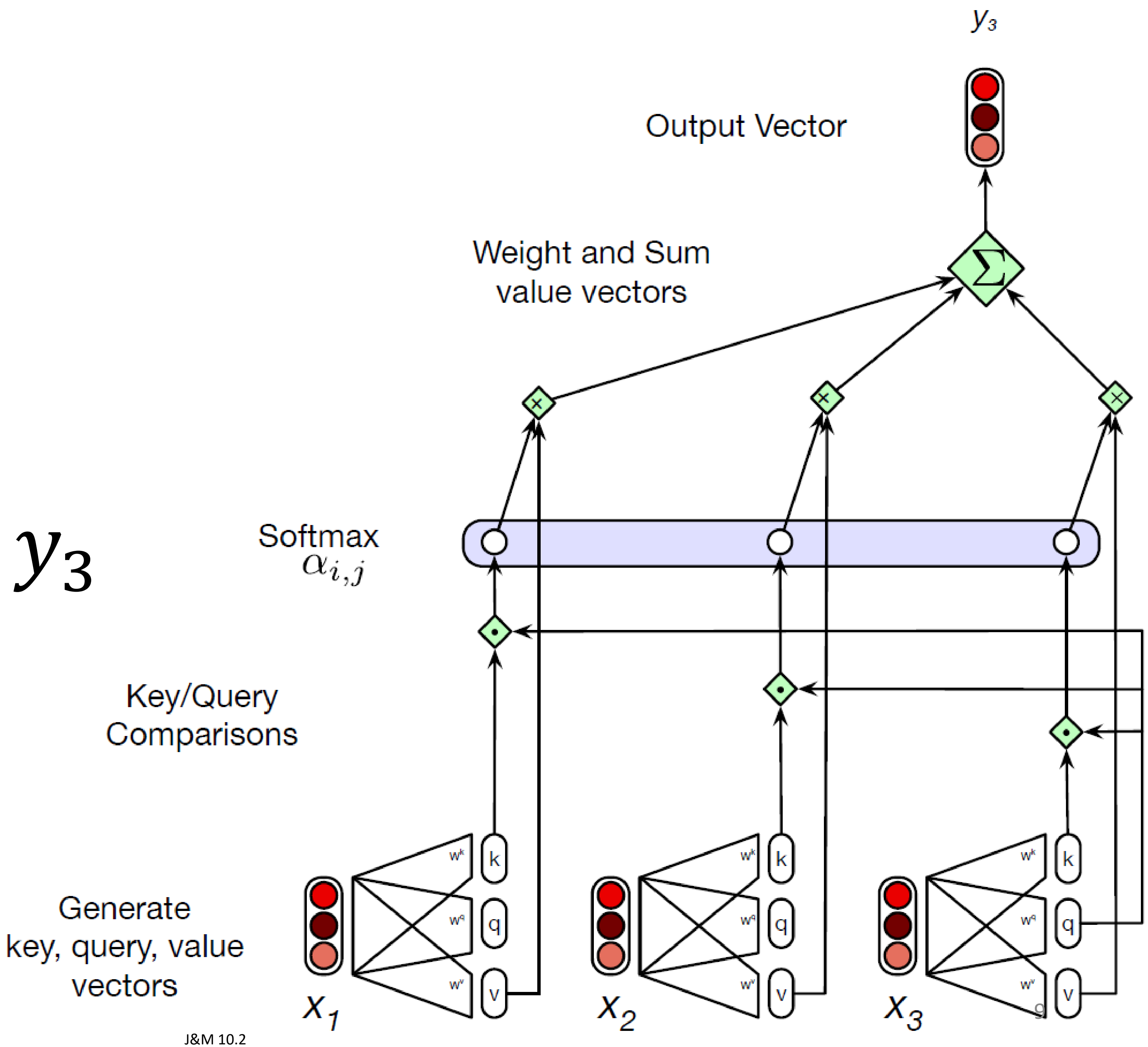
$$\alpha_{i,j} = score\left(x_i, x_j\right) = \mathrm{softmax}\left(\frac{q_i \cdot k_j}{\sqrt{d_k}}\right)$$

- Output

$$y_i = \sum_{j=1}^{N} \alpha_{i,j}\, v_j$$

Self attention example

Computing $x_3 \to y_3$



J&M 10.2

Efficiency

- We can take advantage of highly optimized parallel matrix libraries
- Pack all x_i into $N \times d$ matrix X.
- Three multiplications resulting in $N \times d_k$ matrices:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

SAN DIEGO STATE UNIVERSITY

Efficiency

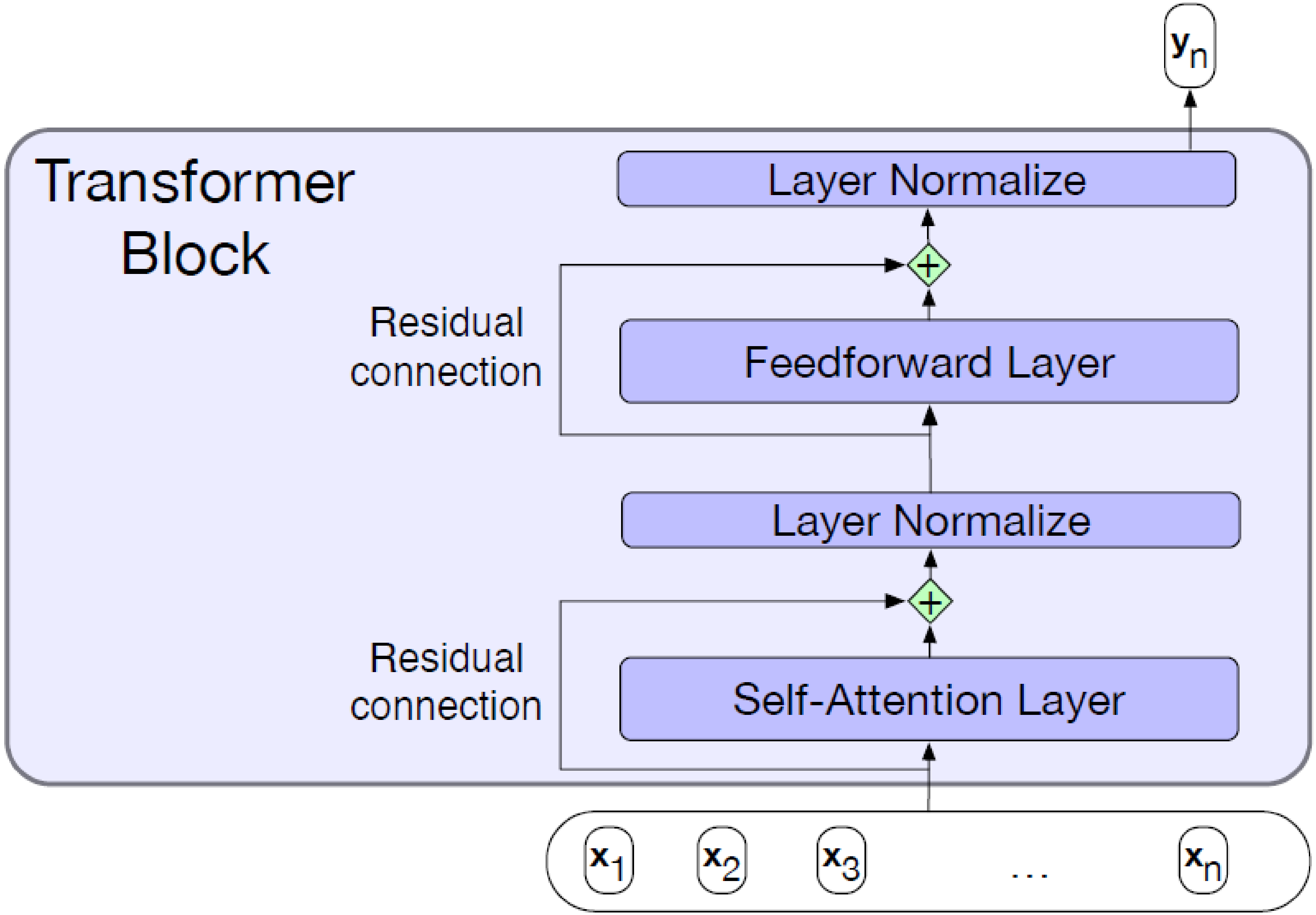
- The score required many multiplications between queries and keys
- We can do this once: QK^T
- Matrix on the right show sample QK^T
  - evident that attention is quadratic with respect to input length
  - causal transformer example
    - upper triangle set to $-\infty$ in postprocessing
    - why?

	q1•k1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
	q2•k1	q2•k2	$-\infty$	$-\infty$	$-\infty$
N	q3•k1	q3•k2	q3•k3	$-\infty$	$-\infty$
	q4•k1	q4•k2	q4•k3	q4•k4	$-\infty$
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

J&M 10.3

The transformer block

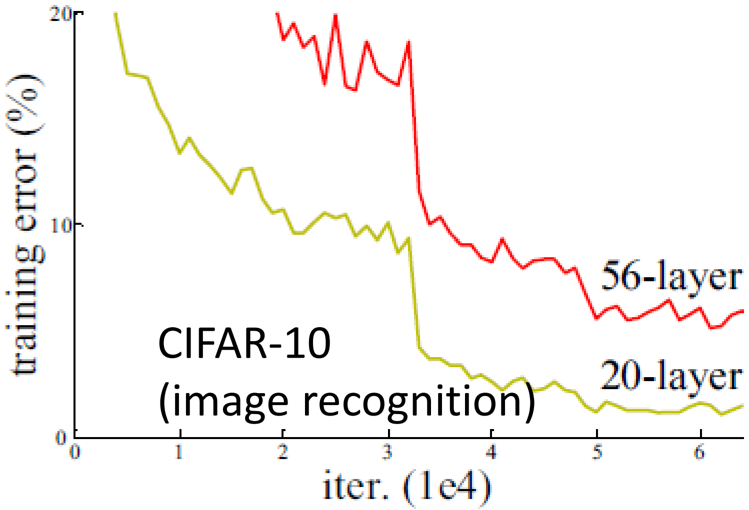


J&M 10.4

British Postal Service, Graham Baker-Smith 2015

Residual layer (He et al. 2016, CVPR)

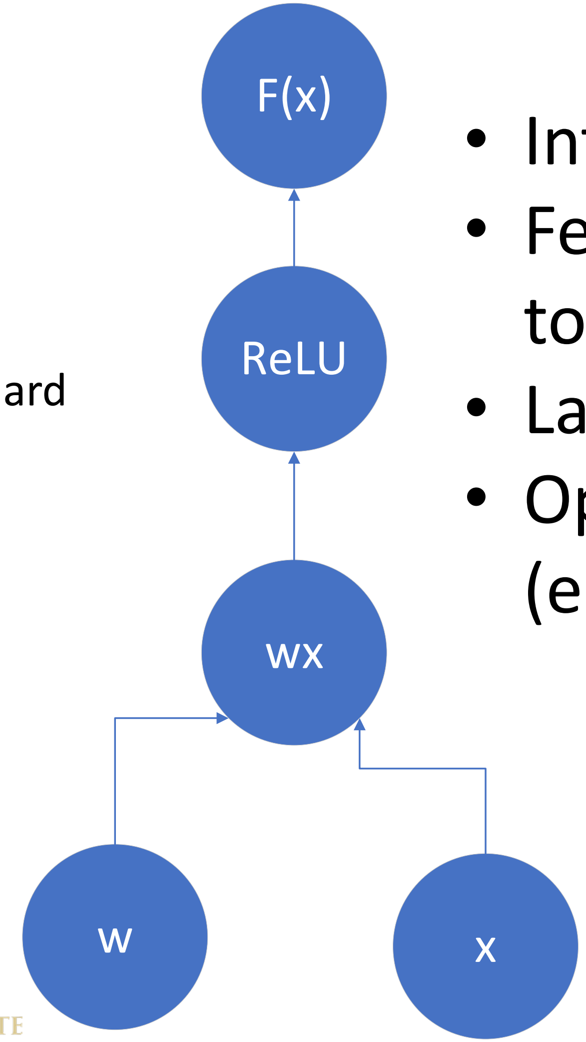
- He et. al. asked: Are deeper networks better?
  - normalization of starting values and intermediate layers (e.g. batchnorm) helps with vanishing/exploding gradients, yet …
  - deeper networks can start to converge and then saturate or degrade
- Insight: provide skip connections that carry the input forward along with what we learn each layer



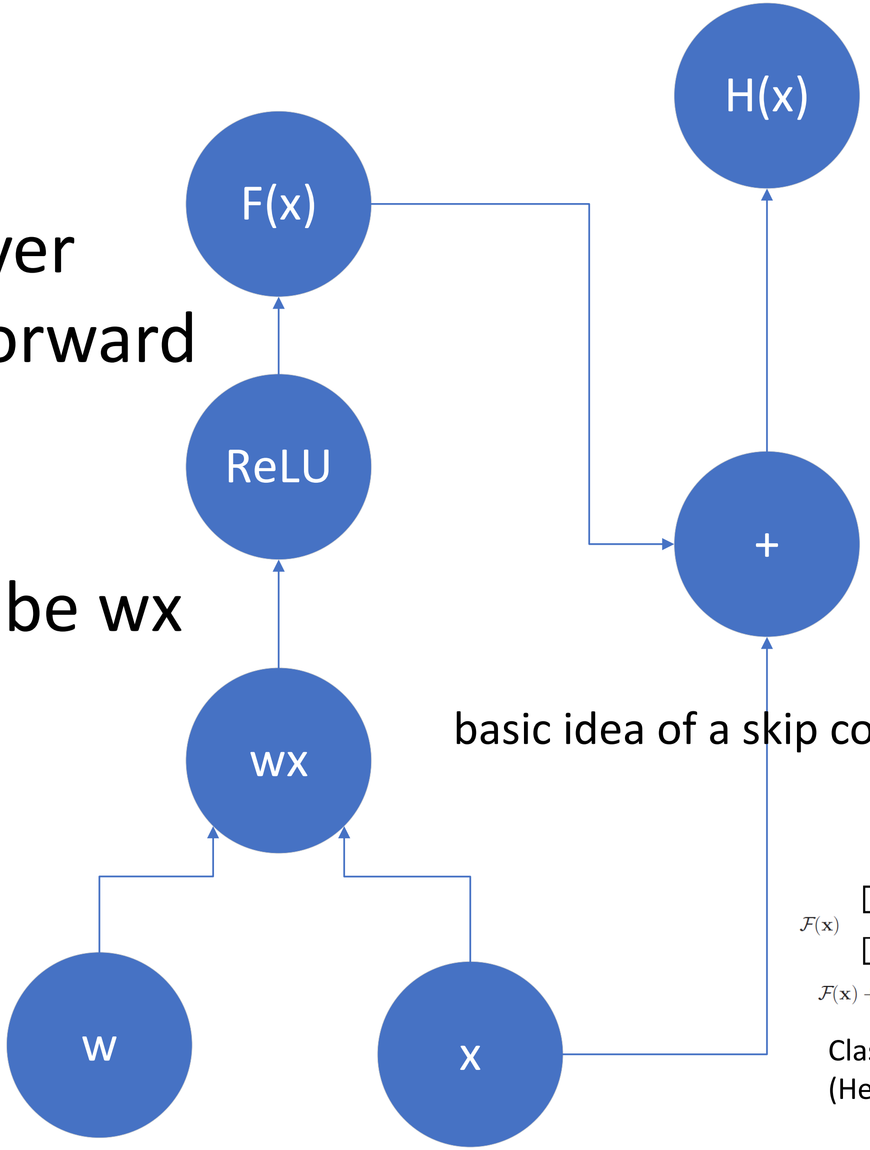
He at al., 2016

Residual Layer (ResNet)

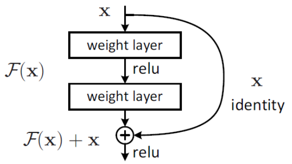
standard



- Introduces “skip” layer
- Feeds information forward to deeper layers
- Layer learns H(X)-x
- Operation need not be wx (e.g., convolution)

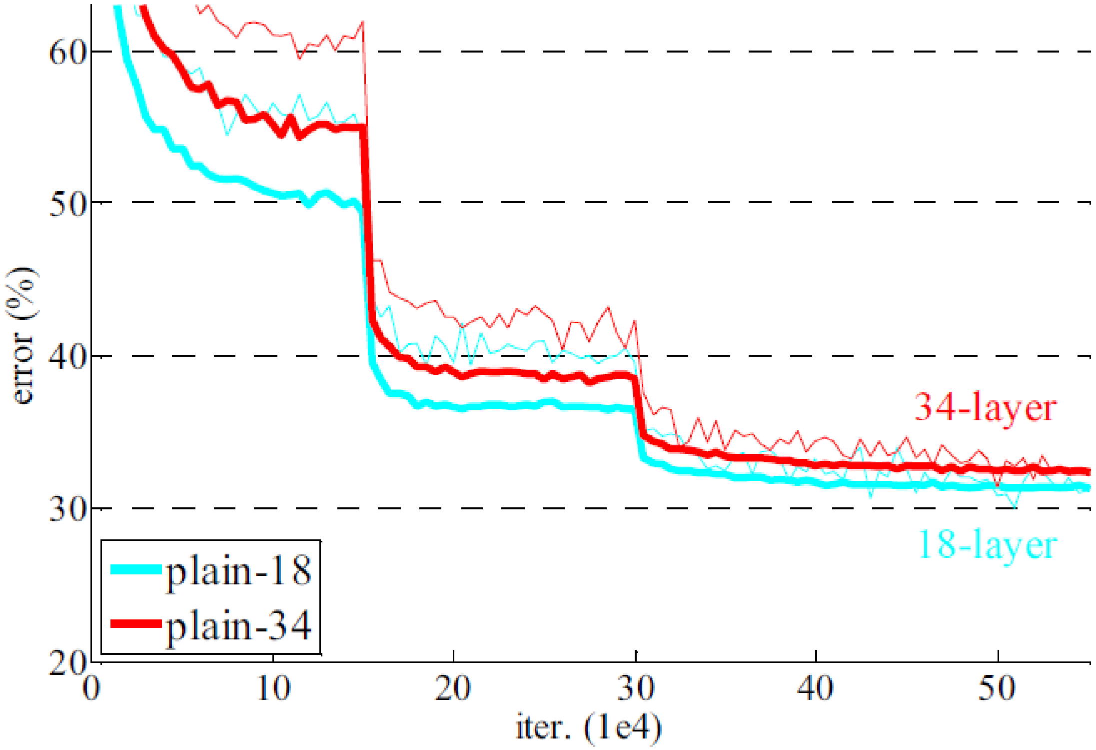


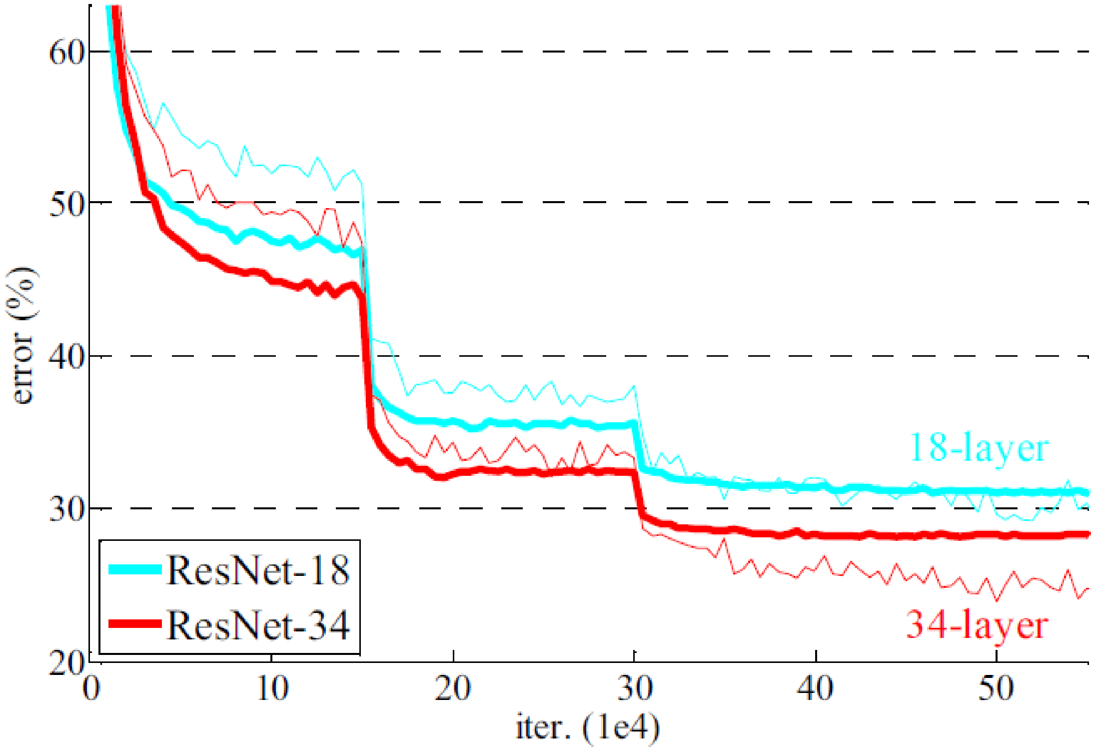
basic idea of a skip connection



Classic ResNet block (He et al. 2016)

ResNet helps learn deeper networks





Training (thin lines) and validation (thick lines) curves for CNN (left) vs CNN ResNet (right) on ImageNet training data with 18 and 34 layers.

San Diego State University

Layer normalization

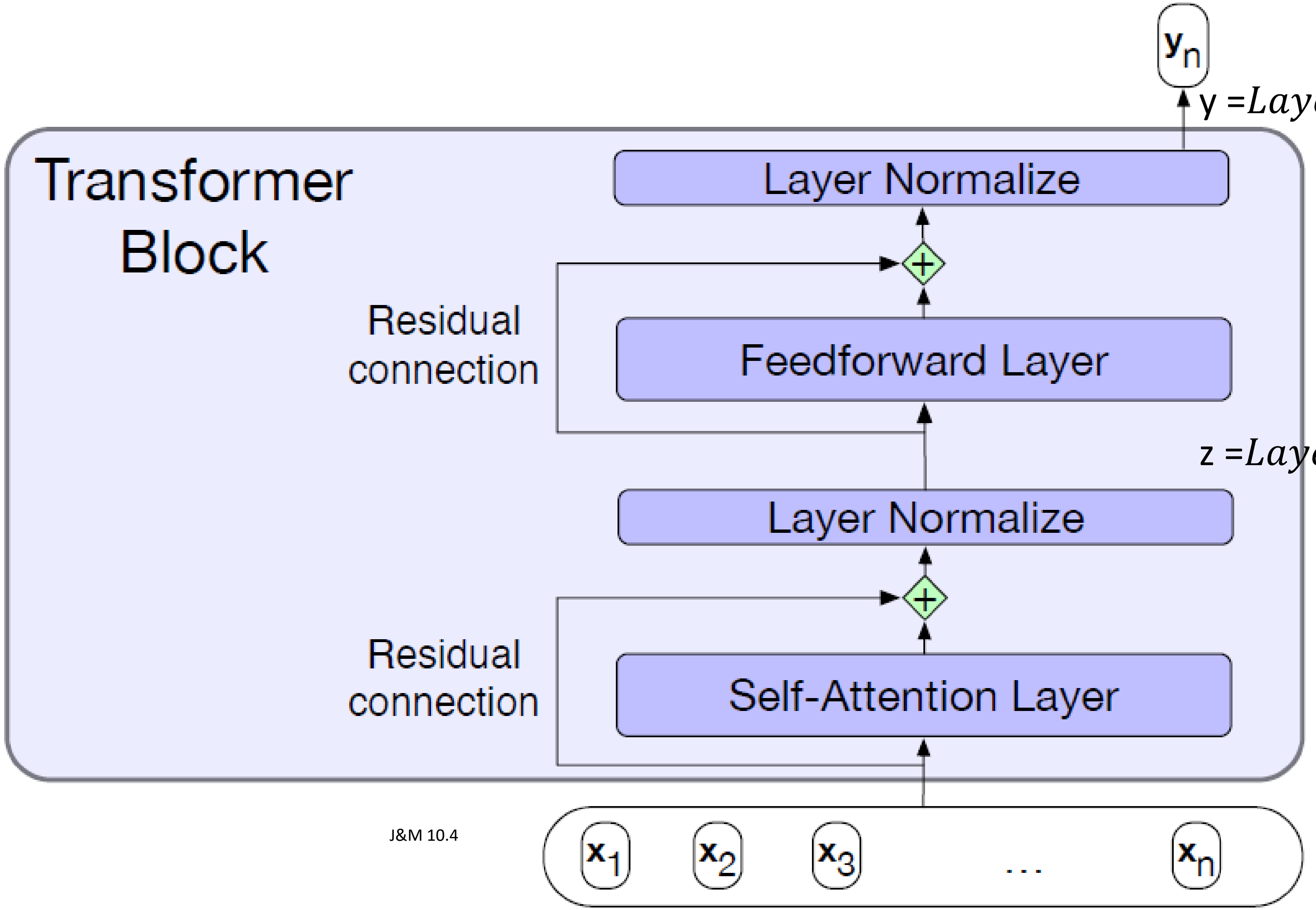
- Similar to Z-score normalization
  - Remember, if $x \sim n(\mu, \sigma^2)$, then $\frac{x-\mu}{\sigma} \sim n(0,1)$
  - adds learnable gain and offset
  - Given input vector *x*, we compute mean μ and standard deviation σ.

$$\hat{x} = \frac{x - \mu}{\sigma}$$

- Layer normalization computes $\gamma\ \hat{x} + \beta$ where γ and β are learnable.

These ideas are similar to what we have seen with batch normalization layers

The transformer block



y $= Layernorm(z + feedforward(z))$

z $= Layernorm(x + SelfAttention(x))$

J&M 10.4

Multihead attention

or multiple heads are better than one…

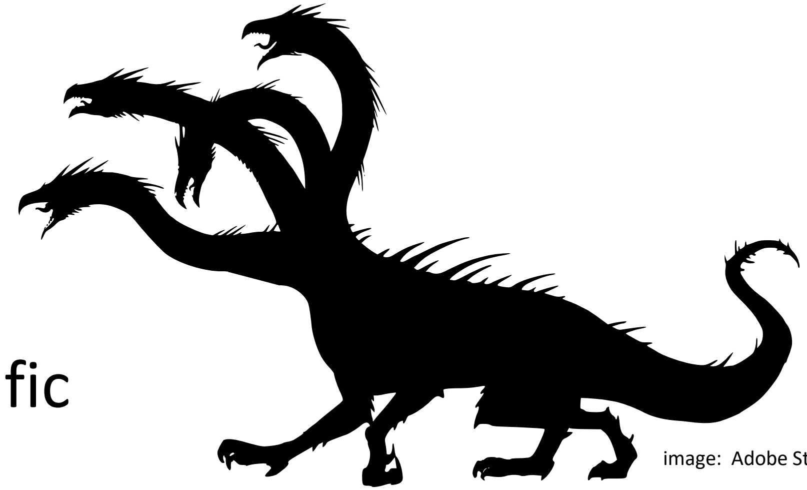
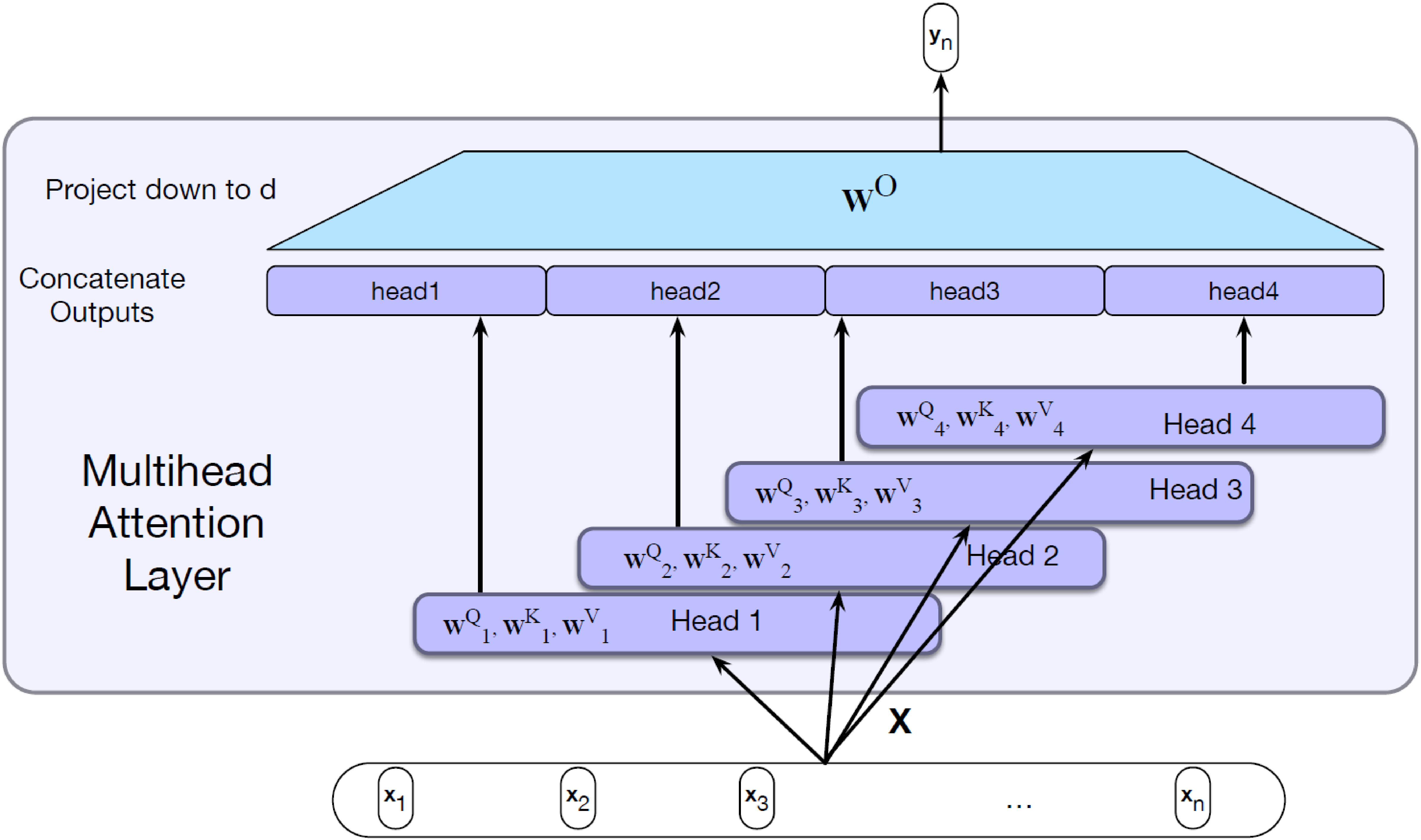


image: Adobe Stock

- The query and key matrices learn a specific type of relationship.
- It might be the case that there is more than one type of relationship to be learned…
- Let h = # heads. Learn *h* query, key, and value transforms
  - $W_1^Q, W_2^Q, \ldots, W_h^Q$
  - $W_1^K, W_2^K, \ldots, W_h^K$
  - $W_1^V, W_2^V, \ldots, W_h^V$
- We also relax the requirement that weight layers be square
  - W_i^Q, W_i^K can be $\mathrm{d} \times d_k$
  - W_i^V can be $d \times d_v$

San Diego State University

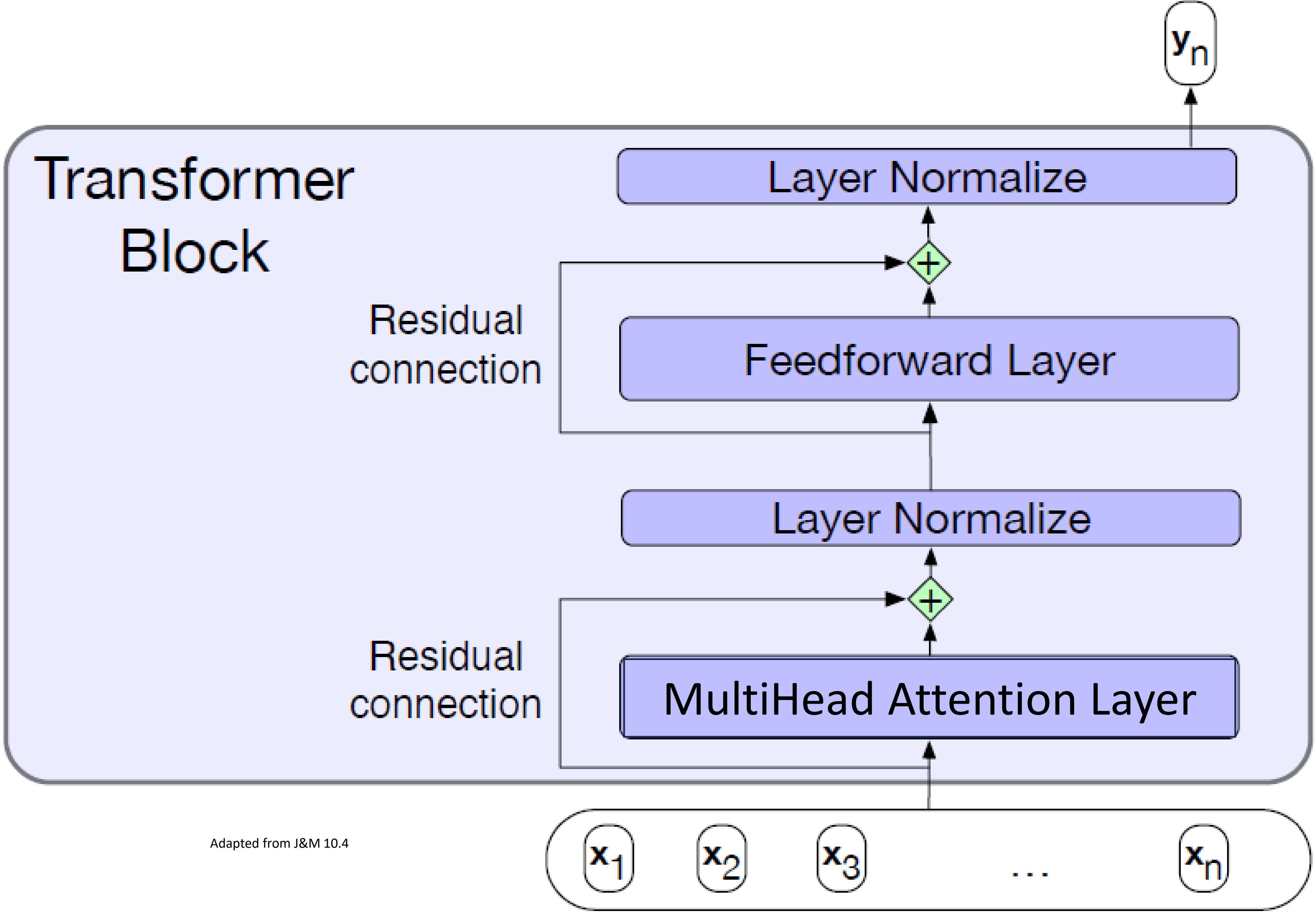


y_n
Project down to d
W^O
Concatenate Outputs
head1
head2
head3
head4
W^Q_4, W^K_4, W^V_4
Head 4
W^Q_3, W^K_3, W^V_3
Head 3
W^Q_2, W^K_2, W^V_2
Head 2
W^Q_1, W^K_1, W^V_1
Head 1
Multihead Attention Layer
X
x_1
x_2
x_3
…
x_n

Multihead attention layer

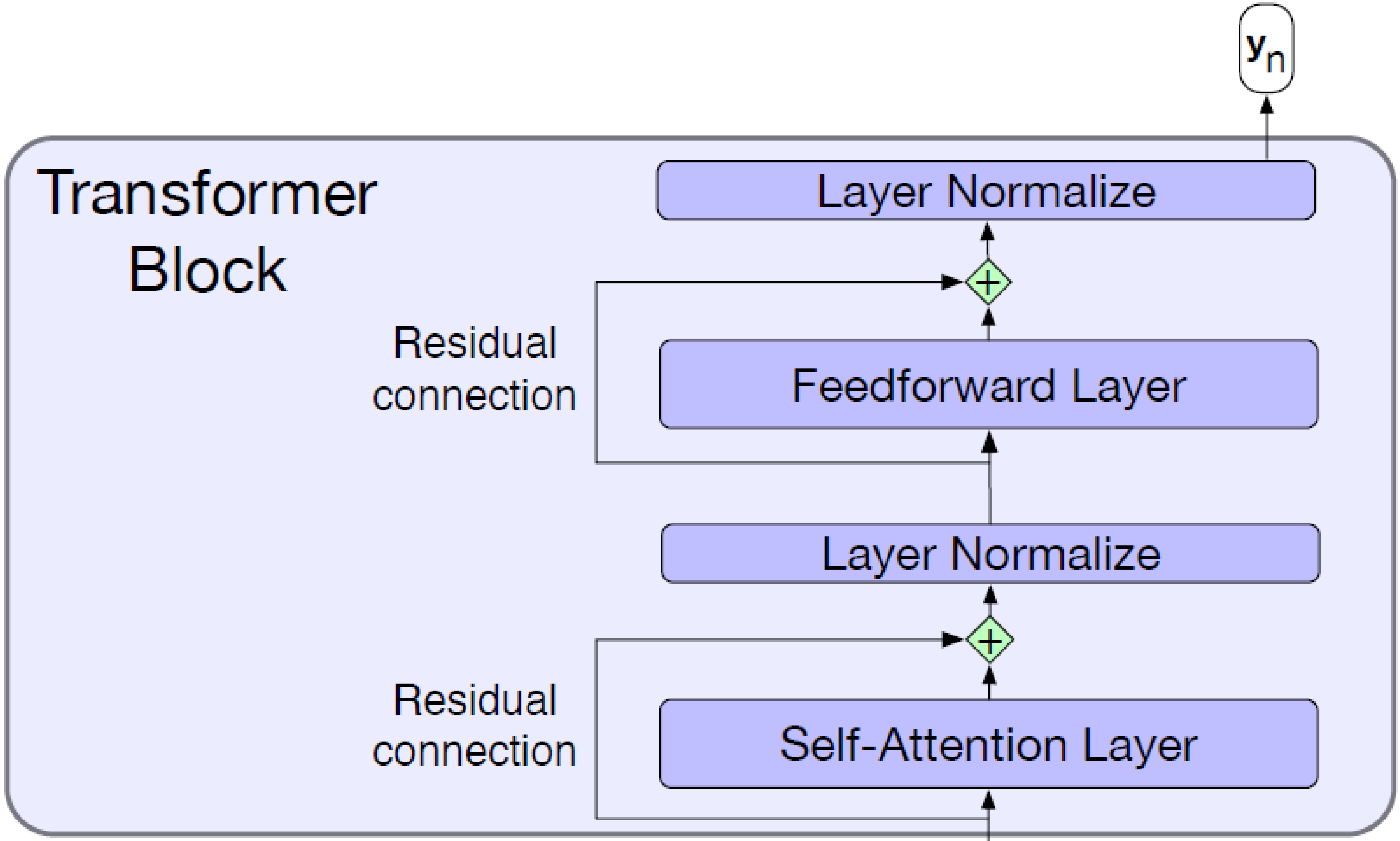
- Each head produces a vector of length d_v.
- When we concatenate the outputs of the N heads, we end up with a vector of size $1 \times \mathrm{h}d_v$: $[o_{h_1} \oplus o_{h_2} \oplus o_{h_3} \oplus \ldots \oplus o_{h_N}]$
- Project down to $1 \times d$
  - Learn a weight matrix w^O of size $\mathrm{hd_v} \times d$
  - So $[o_{h_1} \oplus o_{h_2} \oplus o_{h_3} \oplus \ldots \oplus o_{h_N}]\mathrm{w^o}$ is size: $(1 \times \mathrm{h}d_v)(\mathrm{hd_v} \times d) \rightarrow 1 \times d$.

San Diego State University

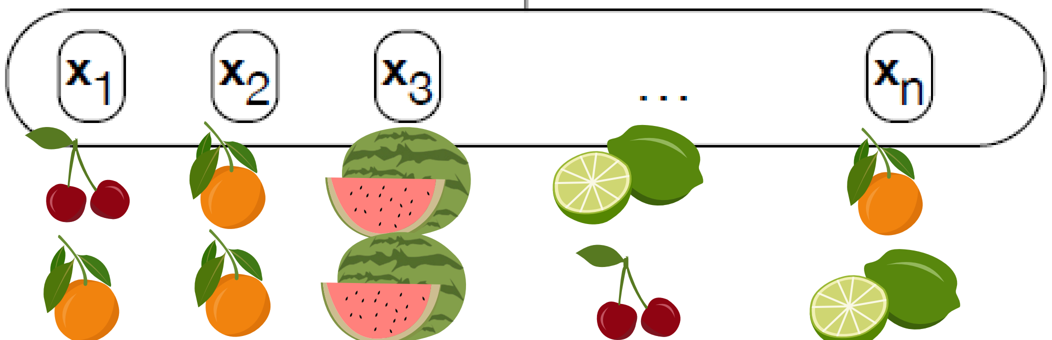


Adapted from J&M 10.4

Fruit for thought



Suppose we predict x3 with the following two sequences
Is one sequence likely to be more likely than the other?



sequence cherry

sequence orange

Adapted from J&M 10.4

San Diego State University

Positional encodings

- Supplement the input vector with something that lets us learn relative position
- Active area of research
- Common simple method: Use sinusoids of varying frequency

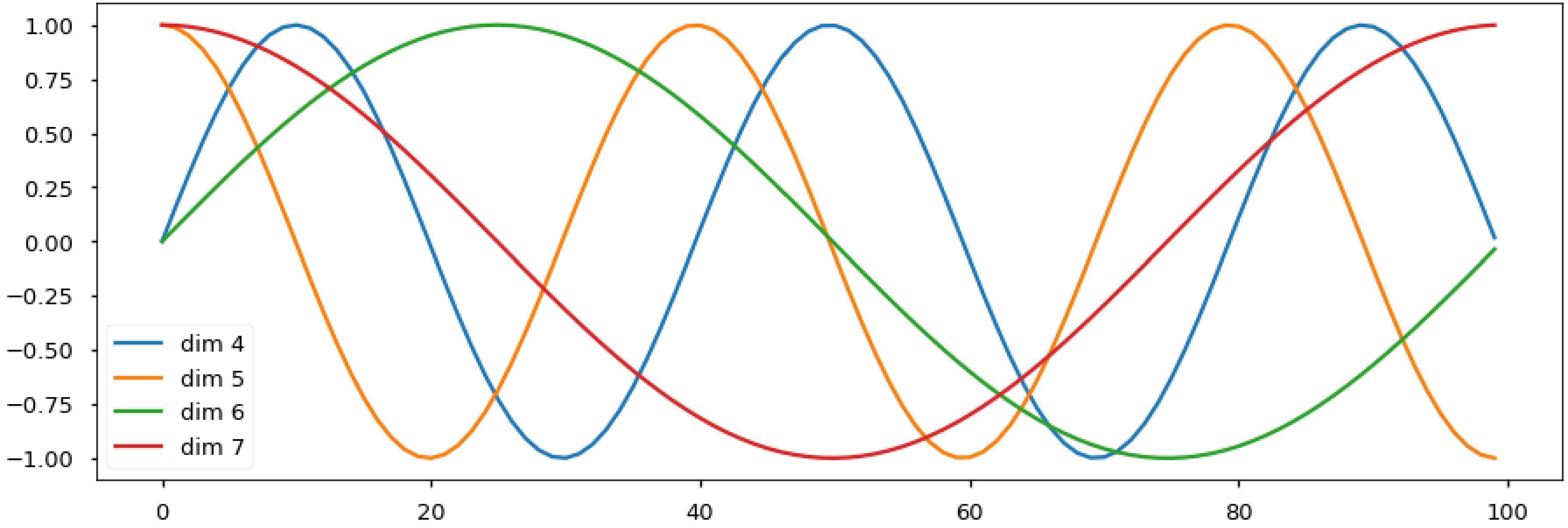


image: Rush 2018 https://aclanthology.org/W18-2509.pdf

Keras ≥2.9

- Module keras.layers.attention contains attention layers
  - attention – classic dot-product single-headed attention
  - multi_head_attention – Multi-head attention

  The module is flexible and can be used in ways that we have not talked about.

- Complete example of transformer-based speech recognizer at: https://keras.io/examples/audio/transformer_asr/