

# Designing approximated Machine Learning models in Python for Homomorphic evaluation.

**Sarah Hartinger** Aikata  
Bachelor Thesis

[www.tugraz.at](http://www.tugraz.at)





# Motivation

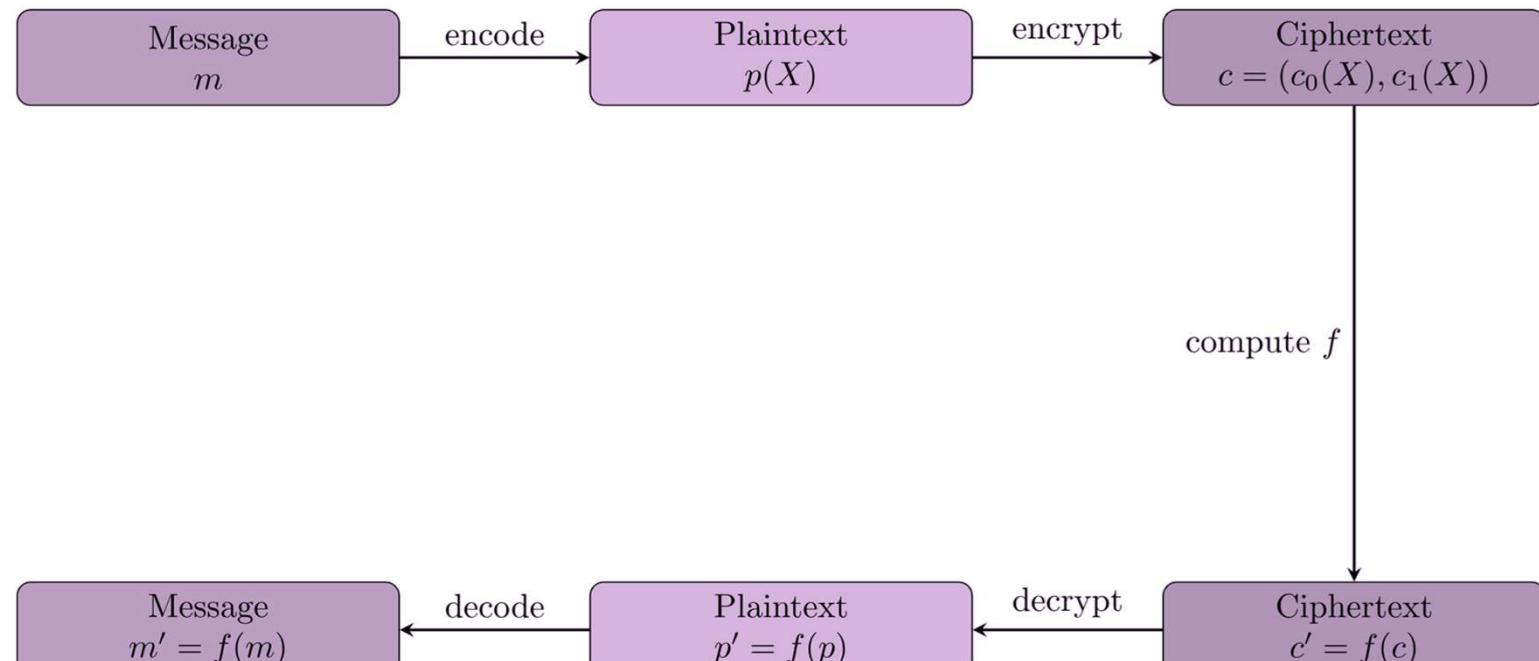
---

- Privacy-Preserving Machine Learning
- Homomorphic Encryption
- Addition, Multiplication, Rotation
- Softmax function

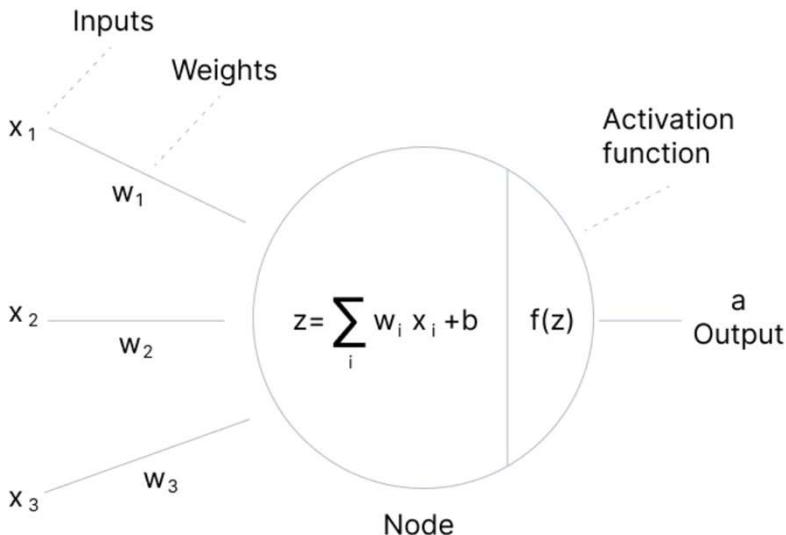
Implement an approximation of a vision transformer by Tin Nguyen, trained on the CIFAR 10 dataset.

Main Goal

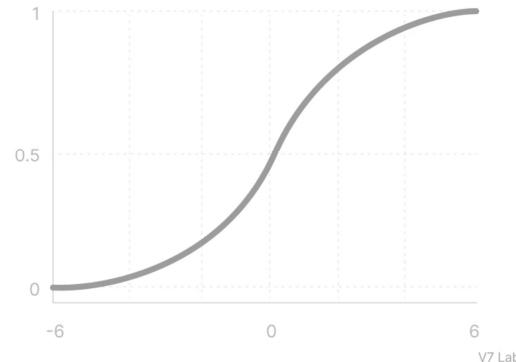
# Cheon-Kim-Kim-Song Scheme



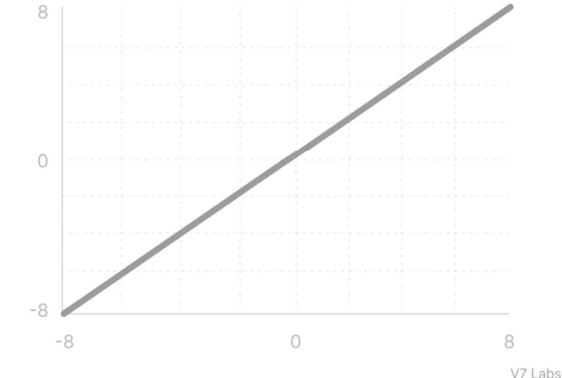
# Artificial Neurons/Nodes



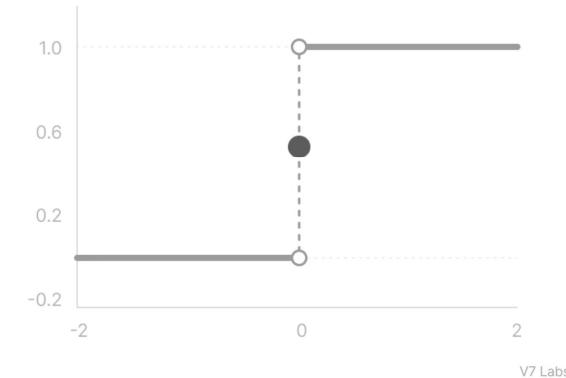
Sigmoid / Logistic



Linear Activation Function

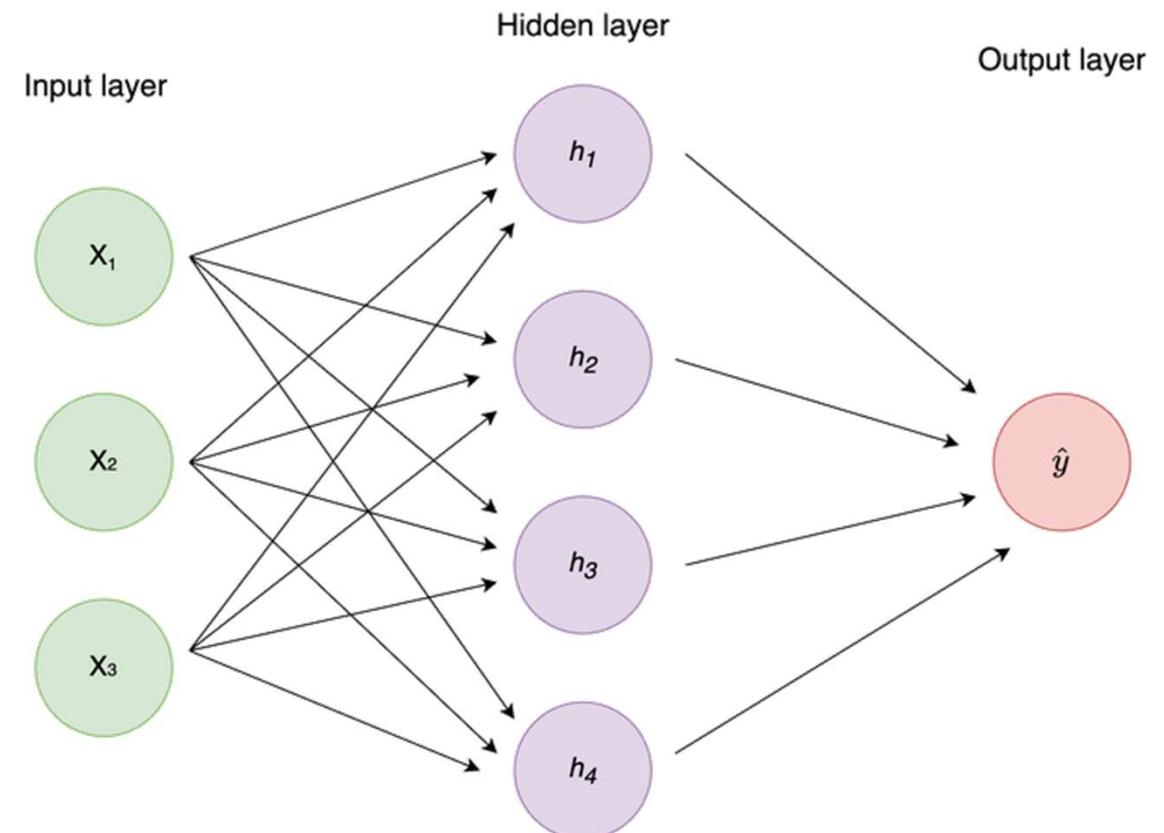


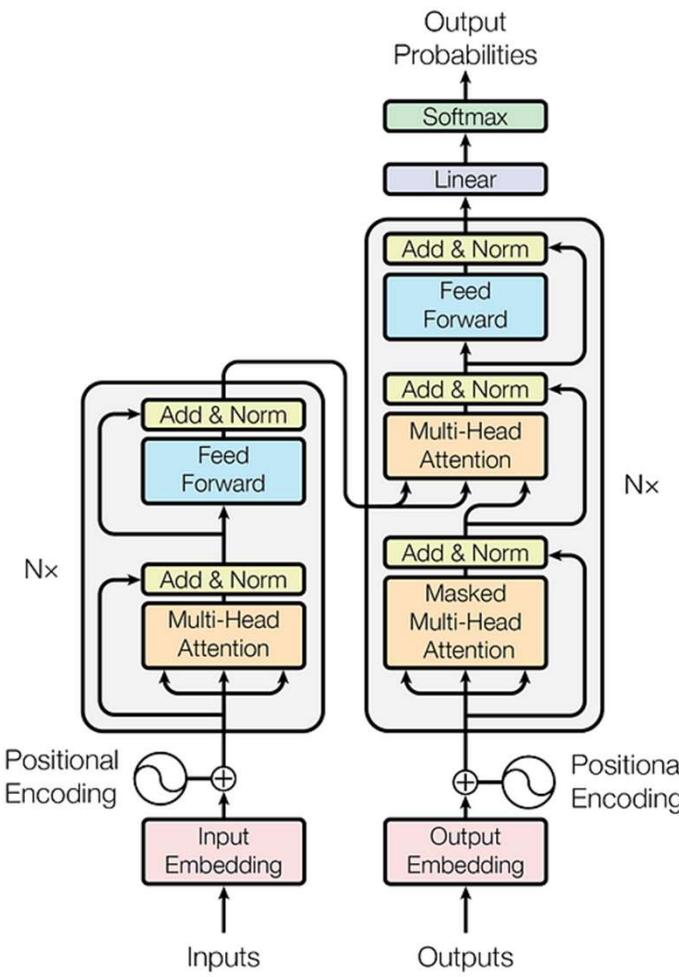
Binary Step Function



# Artificial Neuronal Network

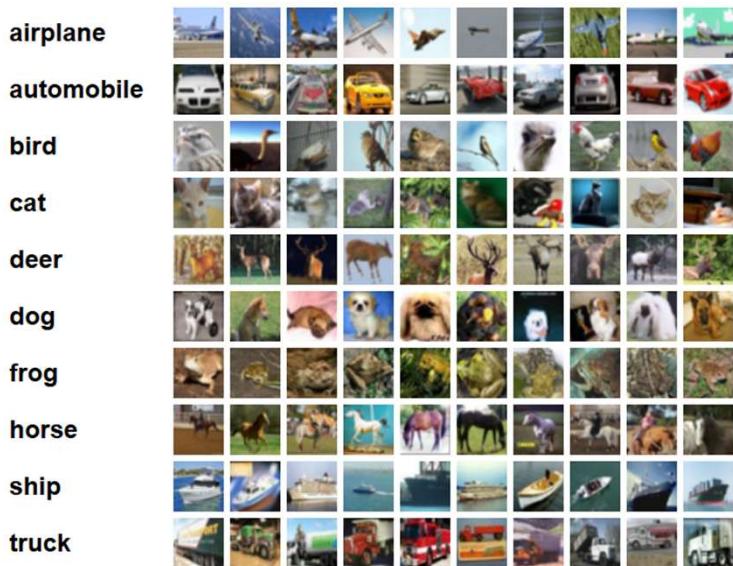
- Feed forward
- Recurrent





# Transformer Model

- Attention is all you need - Google 2017
- Feedforward
- Attention

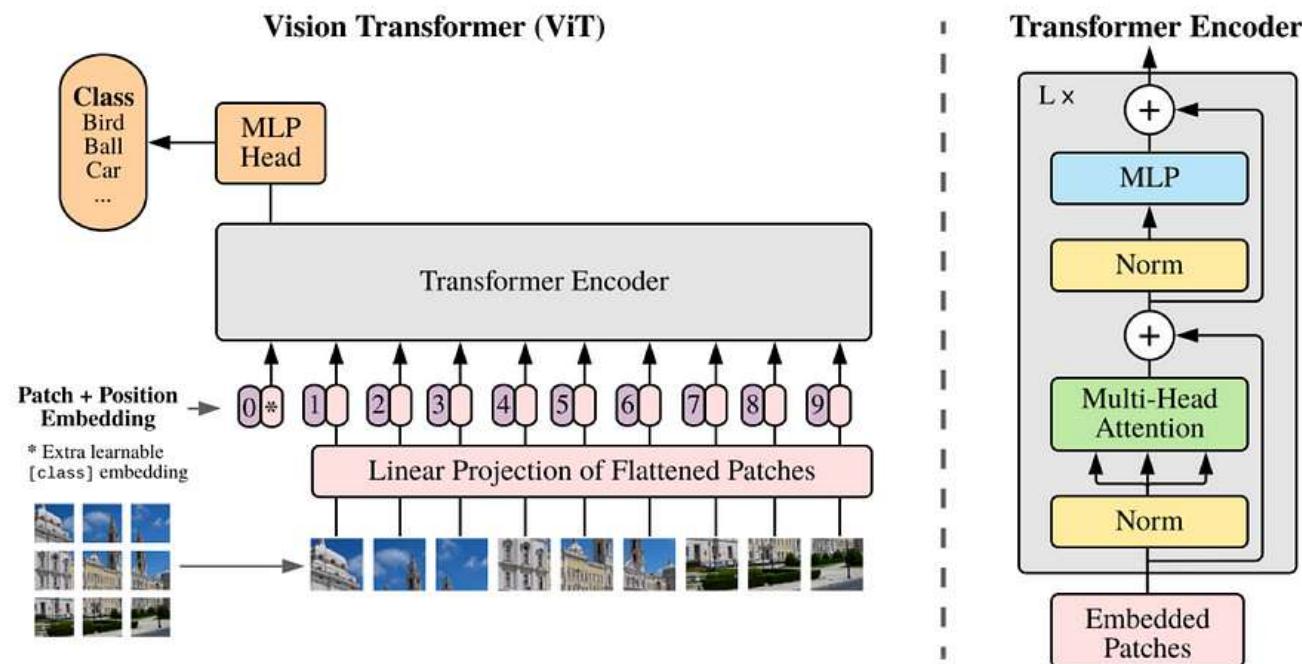


# Vision Transformer

---

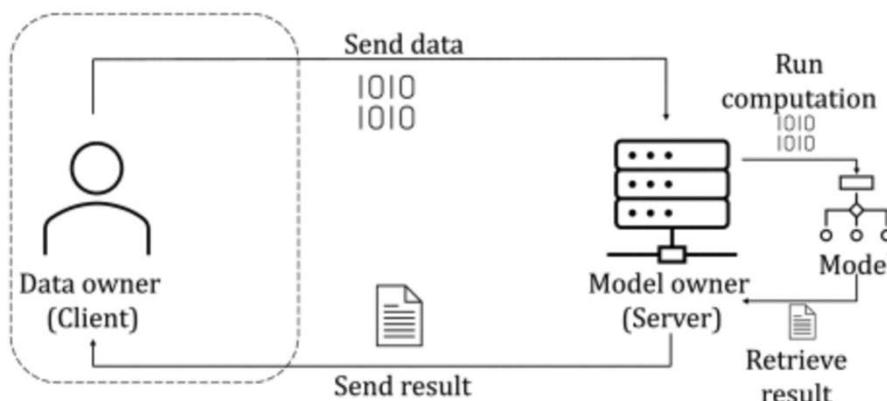
- Tin Nguyen - „Vision Transformer form Scratch“
- Inspired by “An Image is worth 16x16 Words: Transformers for Image Recognition at Scale”
- CIFAR-10 dataset

# Vision Transformer Model



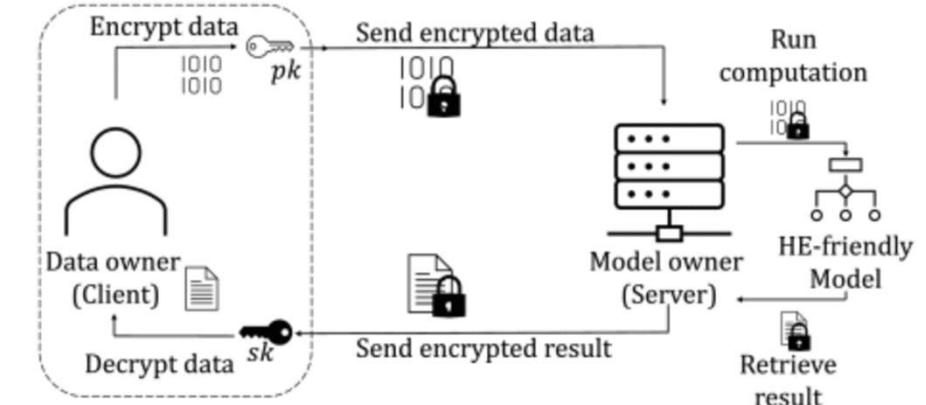
# Machine Learning

Data owner's domain of control



(a) Without privacy preservation

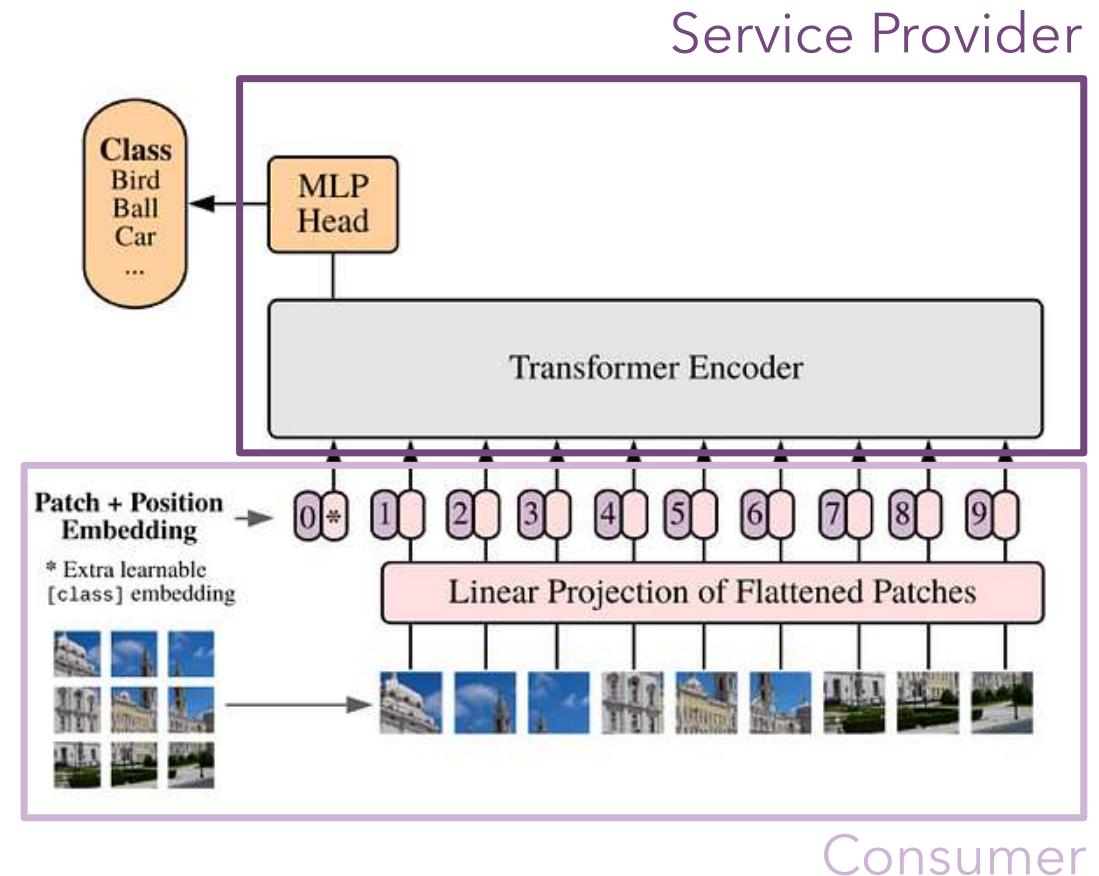
Data owner's domain of control



(b) With privacy preservation

# HE Vit Setting

- **Input:** Encrypted Input Embeddings
- **Output:** Encrypted CLS Token



# Multiplication

Expanded Vector x Row-major Matrix

$$\begin{array}{c}
 \left( \begin{array}{cccc} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{array} \right) \\
 \downarrow \\
 \left( \begin{array}{cccc} a_1 & a_2 & \dots & a_n \end{array} \right) \\
 \xrightarrow{\quad} \\
 \left( \begin{array}{ccccc} a_1 & a_1 & \dots & a_1 & a_2 & a_2 & \dots & a_2 & \dots \\ w_{1,1} & w_{1,2} & \dots & w_{1,n} & w_{2,1} & w_{2,2} & \dots & w_{2,n} & \dots \end{array} \right) \times \cdot \\
 \xrightarrow{\quad} \\
 \left( \begin{array}{cccc} \Sigma & \Sigma & \dots & \Sigma & \dots & \dots & \dots & \dots \end{array} \right)
 \end{array}$$

Repeated Vector x Column-major Matrix

$$\begin{array}{c}
 \left( \begin{array}{cccc} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \dots & w_{n,n} \end{array} \right) \\
 \downarrow \\
 \left( \begin{array}{cccc} a_1 & a_2 & \dots & a_n \end{array} \right) \\
 \xrightarrow{\quad} \\
 \left( \begin{array}{ccccc} a_1 & a_2 & \dots & a_n & a_1 & a_2 & \dots & a_n & \dots \end{array} \right) \times \cdot \\
 \xrightarrow{\quad} \\
 \left( \begin{array}{ccccc} w_{1,1} & w_{2,1} & \dots & w_{n,1} & w_{1,2} & w_{2,2} & \dots & w_{n,2} & \dots \end{array} \right) = \cdot \\
 \xrightarrow{\quad} \\
 \left( \begin{array}{cccc} \Sigma & \dots & \dots & \Sigma & \dots & \dots & \dots & \dots \end{array} \right)
 \end{array}$$

# Wrap Up

## Wrap Up Expanded

$$\begin{aligned} a &: \left( \begin{array}{|c|c|c|} \hline a_1 & a_1 & a_1 \\ \hline a_2 & a_2 & a_2 \\ \hline a_3 & a_3 & a_3 \\ \hline \end{array} \right) \\ b &: \left( \begin{array}{|c|c|c|} \hline b_1 & b_1 & b_1 \\ \hline b_2 & b_2 & b_2 \\ \hline b_3 & b_3 & b_3 \\ \hline \end{array} \right) \\ c &: \left( \begin{array}{|c|c|c|} \hline c_1 & c_1 & \boxed{c_1} \\ \hline c_2 & c_2 & \boxed{c_2} \\ \hline c_3 & c_3 & \boxed{c_3} \\ \hline \end{array} \right) \\ M &: \left( \begin{array}{|c|c|c| |c|c|c| |c|c|c|} \hline a_1 & b_1 & c_1 & a_2 & b_2 & c_2 & a_3 & b_3 & c_3 \\ \hline \end{array} \right) \end{aligned}$$

## Wrap Up Repeated

$$\begin{aligned} a &: \left( \begin{array}{|c|c|c|} \hline a_1 & a_2 & a_3 \\ \hline a_1 & a_2 & a_3 \\ \hline a_1 & a_2 & a_3 \\ \hline \end{array} \right) \\ b &: \left( \begin{array}{|c|c|c|} \hline b_1 & b_2 & b_3 \\ \hline b_1 & b_2 & b_3 \\ \hline b_1 & b_2 & b_3 \\ \hline \end{array} \right) \\ c &: \left( \begin{array}{|c|c|c|} \hline c_1 & c_2 & c_3 \\ \hline c_1 & c_2 & c_3 \\ \hline c_1 & c_2 & c_3 \\ \hline \end{array} \right) \\ M &: \left( \begin{array}{|c|c|c| |c|c|c| |c|c|c|} \hline a_1 & a_2 & a_3 & b_1 & b_2 & b_3 & c_1 & c_2 & c_3 \\ \hline \end{array} \right) \end{aligned}$$

# Input

$$\left( \begin{array}{ccccccc} l_{1,1} & l_{1,1} & l_{1,1} & \dots & l_{1,1} & l_{1,2} & l_{1,2} \\ & & & & l_{1,2} & l_{1,2} & l_{1,2} \\ & & & \dots & & & \dots \\ & & & & l_{1,2} & l_{1,2} & l_{1,2} \\ & & & & \dots & l_{1,64} & l_{1,64} \\ & & & & & l_{1,64} & l_{1,64} \\ & & & & & \dots & \dots \\ & & & & & l_{1,64} & l_{1,64} \end{array} \right)$$

$$\left( \begin{array}{ccccccc} l_{2,1} & l_{2,1} & l_{2,1} & \dots & l_{2,1} & l_{2,2} & l_{2,2} \\ & & & & l_{2,2} & l_{2,2} & l_{2,2} \\ & & & \dots & & & \dots \\ & & & & l_{2,2} & l_{2,2} & l_{2,2} \\ & & & & \dots & l_{2,64} & l_{2,64} \\ & & & & & l_{2,64} & l_{2,64} \\ & & & & & \dots & \dots \\ & & & & & l_{2,64} & l_{2,64} \end{array} \right)$$

⋮

$$\left( \begin{array}{ccccccc} l_{64,1} & l_{64,1} & l_{64,1} & \dots & l_{64,1} & l_{64,2} & l_{64,2} \\ & & & & l_{64,2} & l_{64,2} & l_{64,2} \\ & & & \dots & & & \dots \\ & & & & l_{64,2} & l_{64,2} & l_{64,2} \\ & & & & \dots & l_{64,64} & l_{64,64} \\ & & & & & l_{64,64} & l_{64,64} \\ & & & & & \dots & \dots \\ & & & & & l_{64,64} & l_{64,64} \end{array} \right)$$

# Norm

$$y = \alpha * \left( \frac{x - mean}{\sqrt{variance + \epsilon}} \right) + bias$$

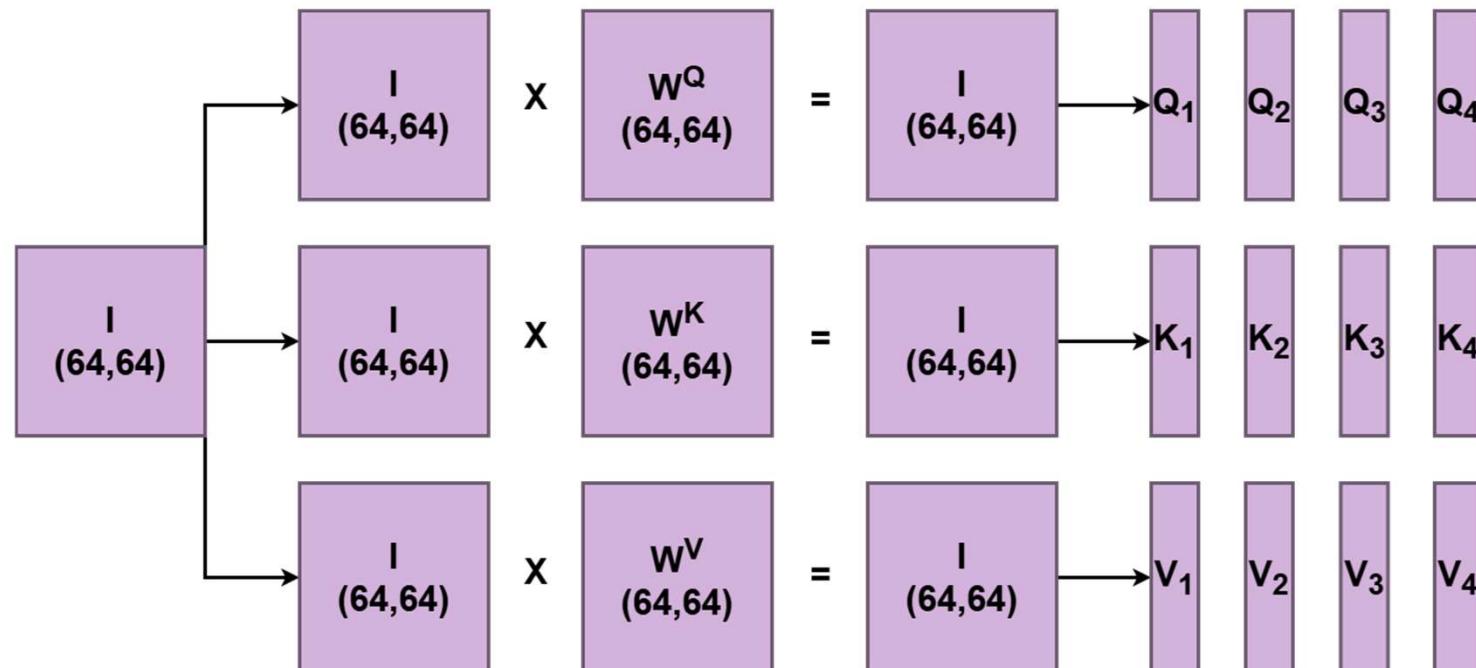
$$y' = (x_i - pre.mean) * (pre.std * \alpha) + b$$

# Attention

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{\text{features}}}\right) * V$$

$$\text{SoftMax}(x) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

# Multi Head Attention



# Multi Head Attention

$$Head(Q_i, K_i, V_i) = \text{SoftMax}\left(\frac{Q_i K_i^T}{\sqrt{\text{values per head}}}\right) * V_i$$

$$\begin{aligned} & MultiHead(Q, K, V) \\ &= \text{Concat}(head_1, head_2, \dots, head_n) * W^O + b^O \end{aligned}$$

# Multi Head Attention - Softmax

$$\begin{aligned} & \frac{1}{64} + \frac{63}{64^2} * x_j + \frac{63 * 62}{64^3} * x_j^2 - \frac{1}{64^2} * \sum_{i=1, i \neq j}^n x_i - \frac{62}{64^3} * \sum_{i=1, i \neq j}^n x_i^2 \\ & - \frac{2 * 62}{64^3} * x_j * \sum_{i=1, i \neq j}^n x_i + \frac{2}{64^3} * \sum_{i=1, i \neq k, i \neq j}^n x_i * \sum_{k=1, k \neq i, k \neq j}^n x_k \end{aligned}$$

# Multi Head Attention

$$Head(Q_i, K_i, V_i) = \text{SoftMax}\left(\frac{Q_i K_i^T}{\sqrt{\text{values per head}}}\right) * V_i$$

$$MultiHead(Q, K, V) = \text{Concat}(head_1, head_2, \dots, head_n) * W^O + b^O$$

# MLP - GELU

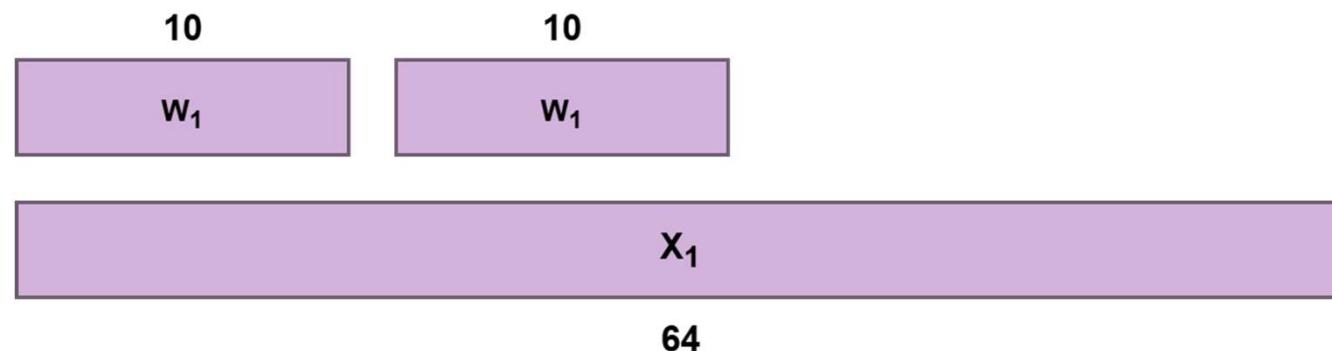
$$MLP(x) = \text{ApproxGELU}(x * W_1 + b_1) * W_2 + b_2$$

$$\begin{aligned} & \text{ApproxGELU}(x) \\ &= 0.5 * x (1 + \tanh(\sqrt{2/pi}) * (x + 0.044715 * x^3)) \end{aligned}$$

$$\begin{aligned} & \text{ApproxTanh}(x) \\ &= \tanh(a) + (1 - \tanh^2(a)) * (x - a) - (\tanh(a) - \tanh^3(a)) \\ &\quad * (x - a)^2 \end{aligned}$$

# Classifier

$$\mathbf{W}^T \begin{pmatrix} 64, 10 \end{pmatrix} \times \mathbf{x} \begin{pmatrix} 64 \end{pmatrix} + \mathbf{b} \begin{pmatrix} 10 \end{pmatrix} = \mathbf{R} \begin{pmatrix} 10 \end{pmatrix}$$



# Result

- OpenFHE library
- Ring Dimension:  $2^{16}$
- Multiplicative Depth: 25
- Accuracy compared to classifications: 43%
- Accuracy of original model: 46%
- Runtime: approx. 2 hours



# Conclusion

---

- Privacy Preserving Vision Transformer using Homomorphic Evaluation
- Enable the use of Machine Learning in sensitive sectors
- Currently focused on inference but might be expanded to training in the future