Deep Learning

TensorFlow

PyTorch

Deep Learning

TensorFlow

PyTorch

+

Google Cloud

Azure

Why is this the case?

- Deep learning solves complicated problems…

Why is this the case?

- Deep learning solves complicated problems...

TEXT DESCRIPTION

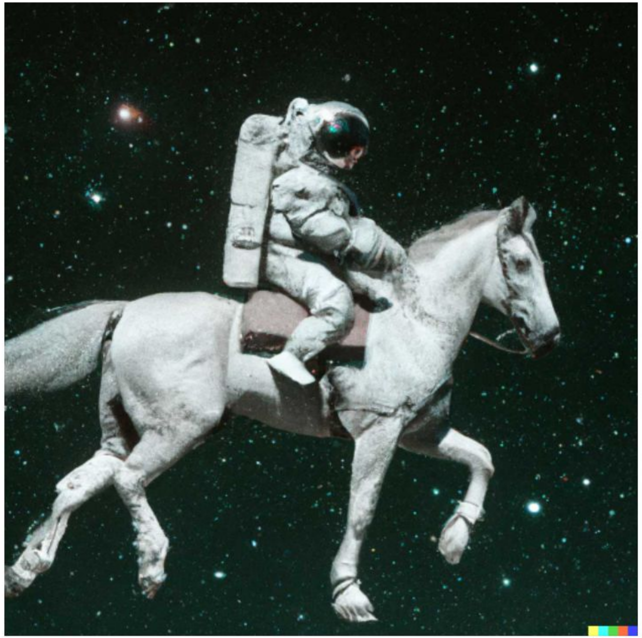
An astronaut Teddy bears A bowl of soup

riding a horse lounging in a tropical resort in space playing basketball with cats in space

in a photorealistic style in the style of Andy Warhol as a pencil drawing

→

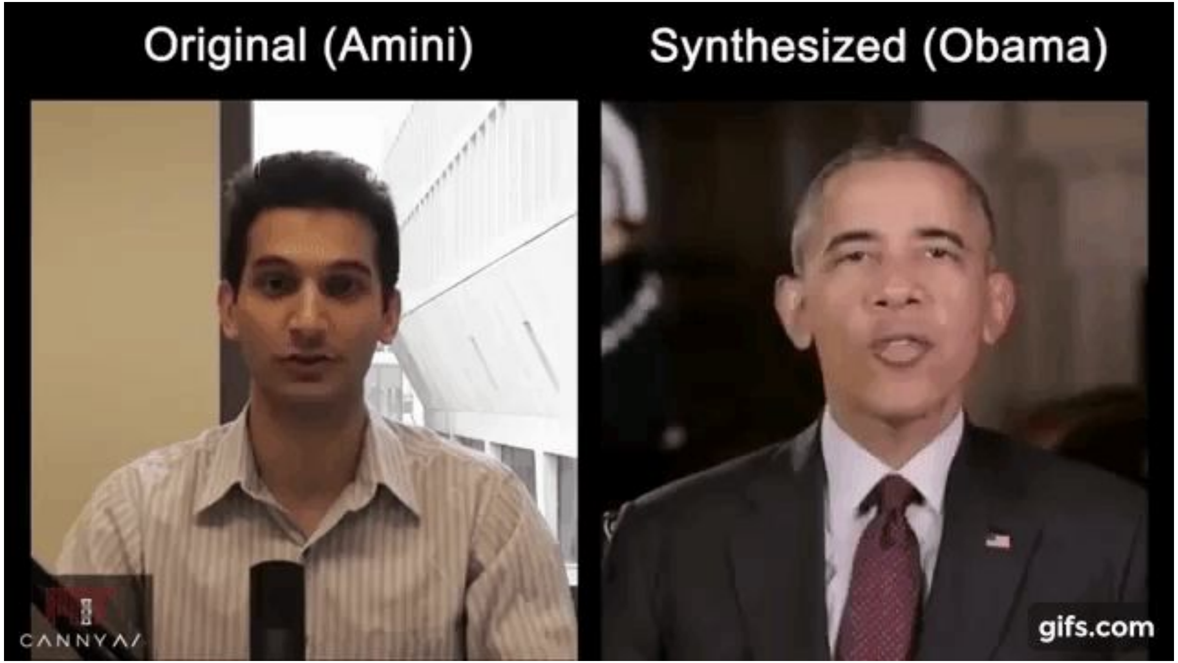
DALL·E 2



https://openai.com/dall-e-2/

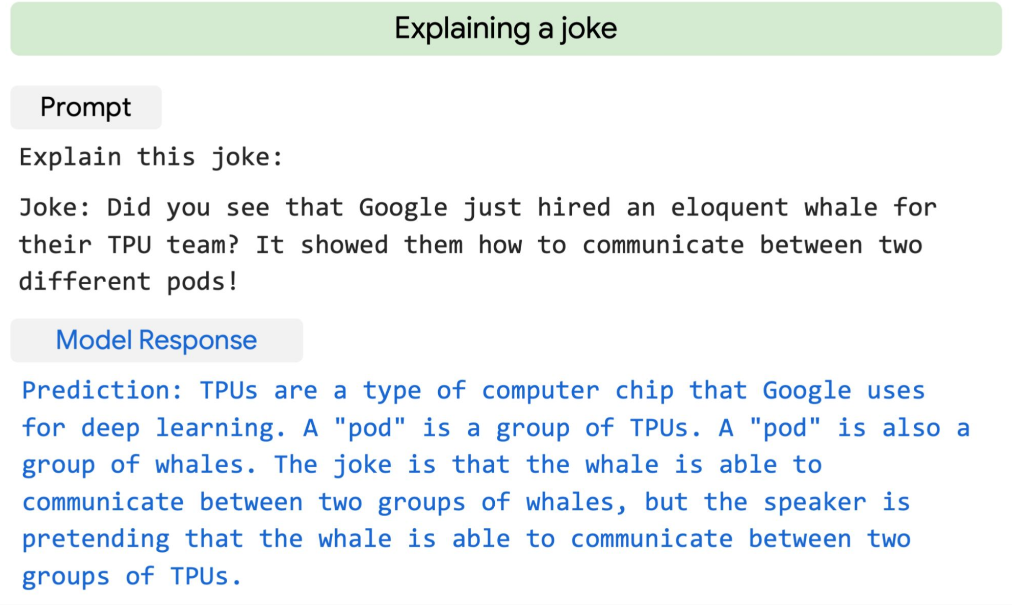
Why is this the case?

- Deep learning solves complicated problems…



Why is this the case?

- Deep learning solves complicated problems...



Explaining a joke

Prompt

Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

Model Response

Prediction: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

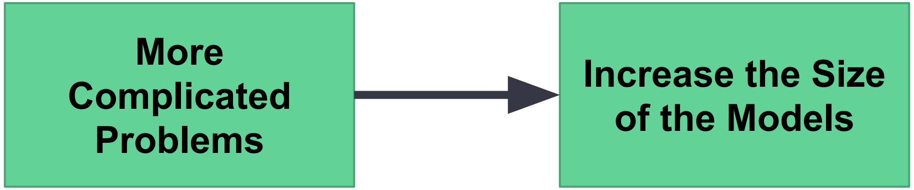
Why is this the case?

- Deep learning solves complicated problems...

More Complicated Problems

Why is this the case?

- Deep learning solves complicated problems…

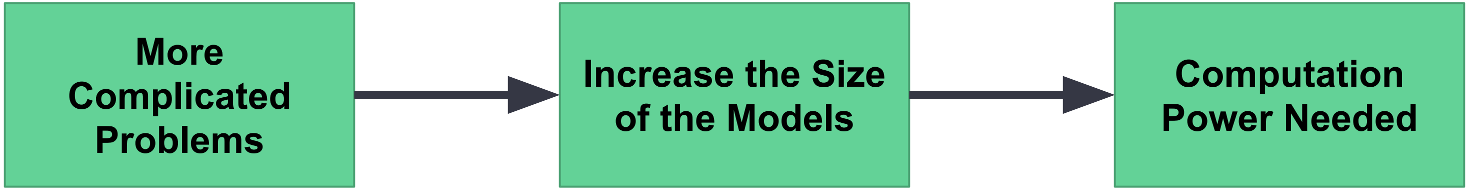


More Complicated Problems

Increase the Size of the Models

Why is this the case?

- Deep learning solves complicated problems…



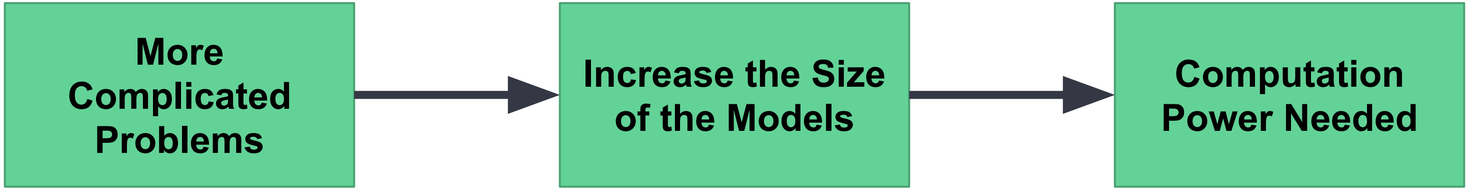
More Complicated Problems

Increase the Size of the Models

Computation Power Needed

Why is this the case?

- Deep learning solves complicated problems…



More Complicated Problems

Increase the Size of the Models

Computation Power Needed

… which require more resources forcing us to use supercomputers / clusters!

Problem:

Is there a way that we can reduce the resources necessary to train / run deep learning models?

Utilizing Lookup Tables for Deep Learning

Tamique de Brito and Noah Faro

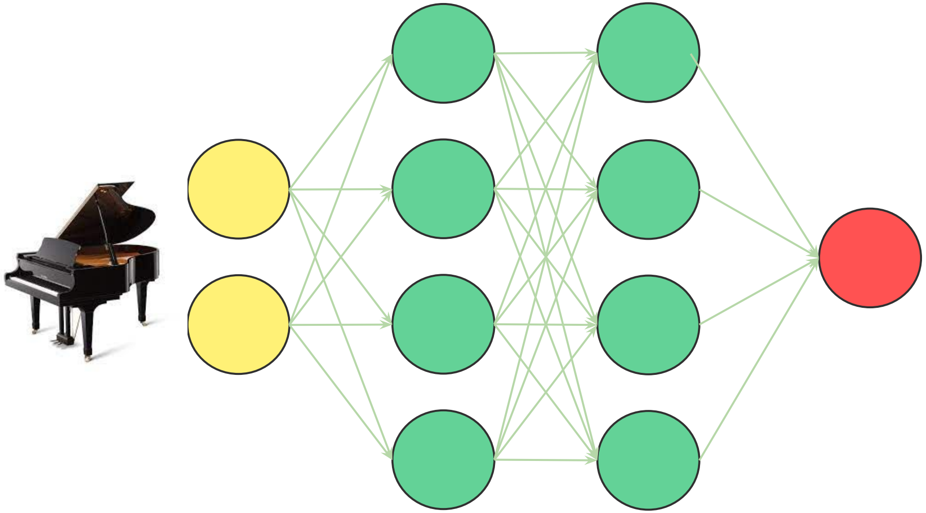
What we are going to cover:

Background

Previous Work

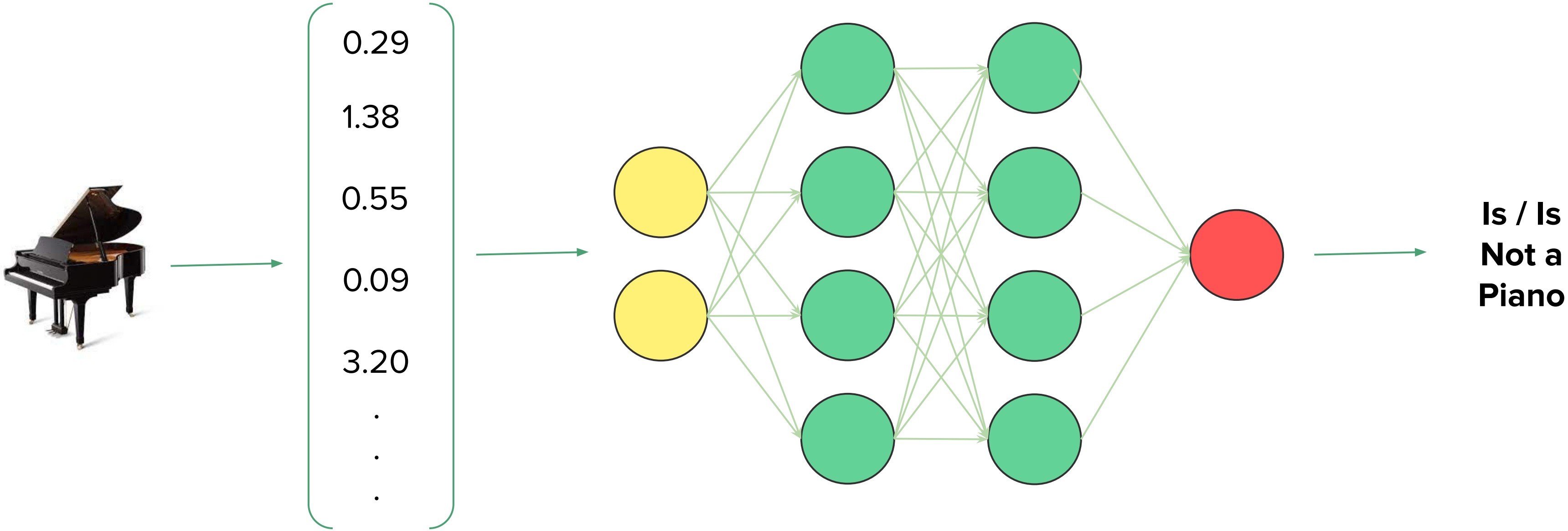
Proposed Solution

Deep Learning Model



Is / Is
Not a
Piano

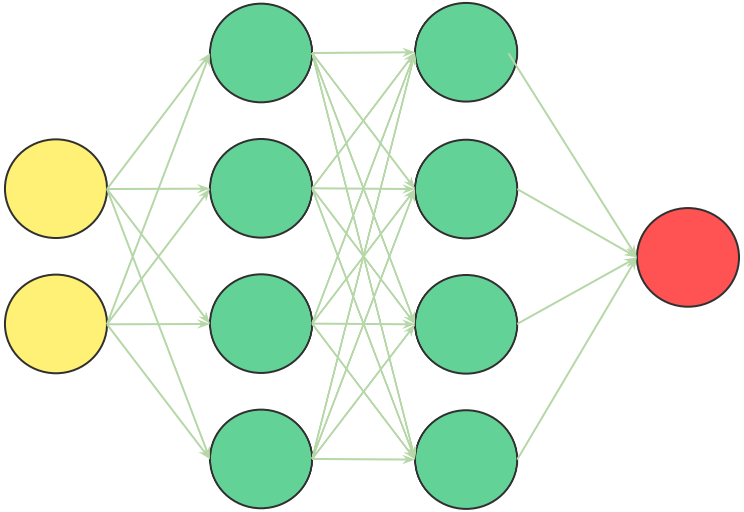
Deep Learning Model



0.29
1.38
0.55
0.09
3.20
.
.
.

Is / Is Not a Piano

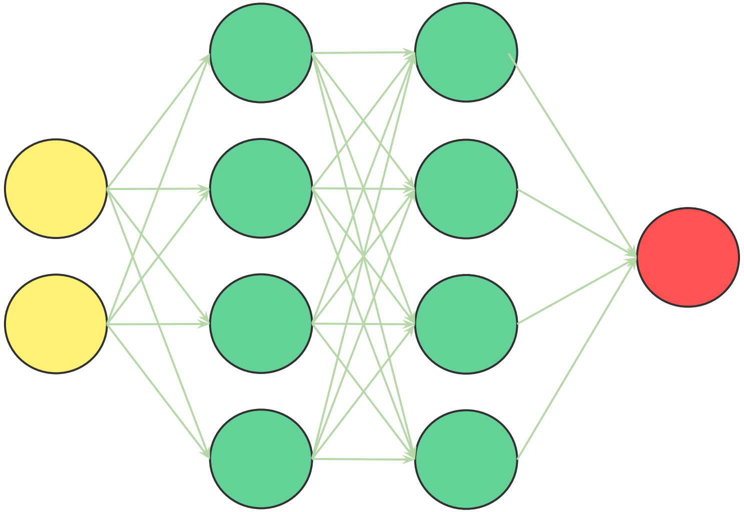
Deep Learning Model



Key:

Symbol	Name	Value
🟡	Inputs	Vector / Matrix
🟢	Computations	$f(x_i, x_j, x_k\ldots)$ -> Vector / Matrix
🔴	Outputs	Vector / Matrix / Boolean

Deep Learning Model



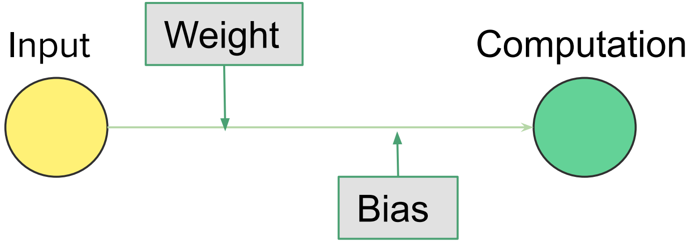
Each arrow represents the addition of some sort of *weight* and *bias*, which is basically a **series of matrix / vector operations**

$$Y = \sum (weight * input) + bias$$

Matrix Operations

Each arrow represents the addition of some sort of *weight* and *bias*, which is basically a **series of matrix / vector operations**

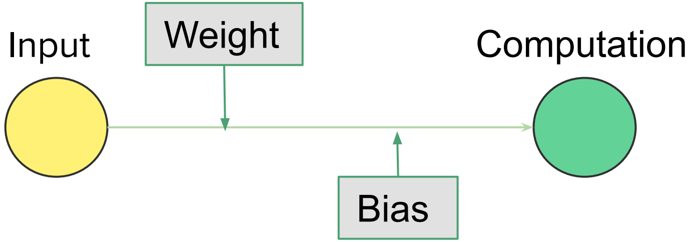
$$Y = \sum (weight * input) + bias$$



Matrix Operations

Each arrow represents the addition of some sort of *weight* and *bias*, which is basically a **series of matrix / vector operations**

$$Y = \sum (weight * input) + bias$$



Input:

1	1
1	1

Weight:

2	2
2	2

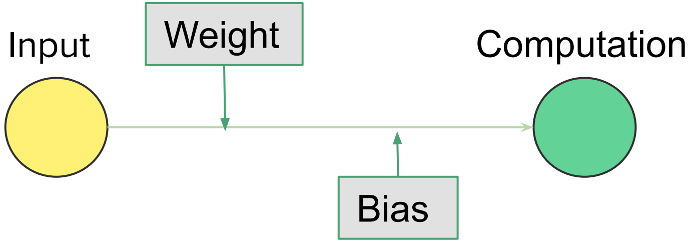
Bias:

3	3
3	3

Matrix Operations

Each arrow represents the addition of some sort of *weight* and *bias*, which is basically a **series of matrix / vector operations**

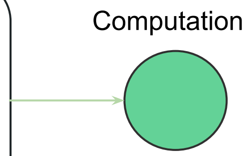
$$Y = \sum (weight * input) + bias$$



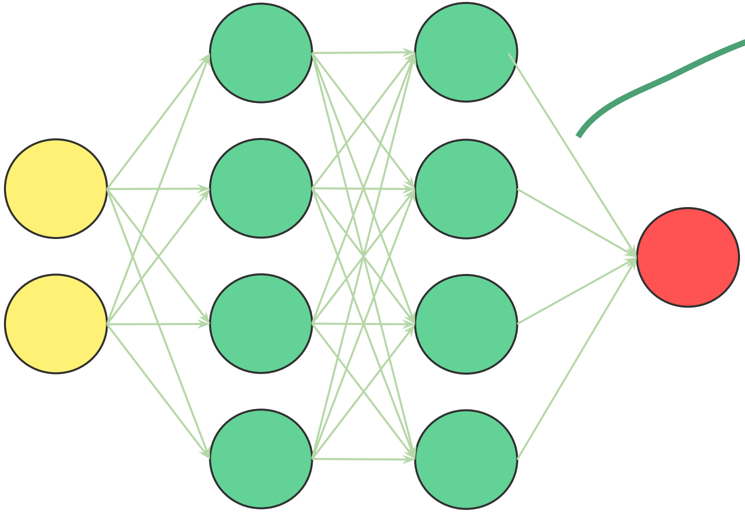
Input: $\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ Weight: $\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix}$ Bias: $\begin{bmatrix}3 & 3\\ 3 & 3\end{bmatrix}$

Computation done by this arrow:

$$\left(\left(\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} \times \begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix}\right) + \begin{bmatrix}3 & 3\\ 3 & 3\end{bmatrix}\right)$$

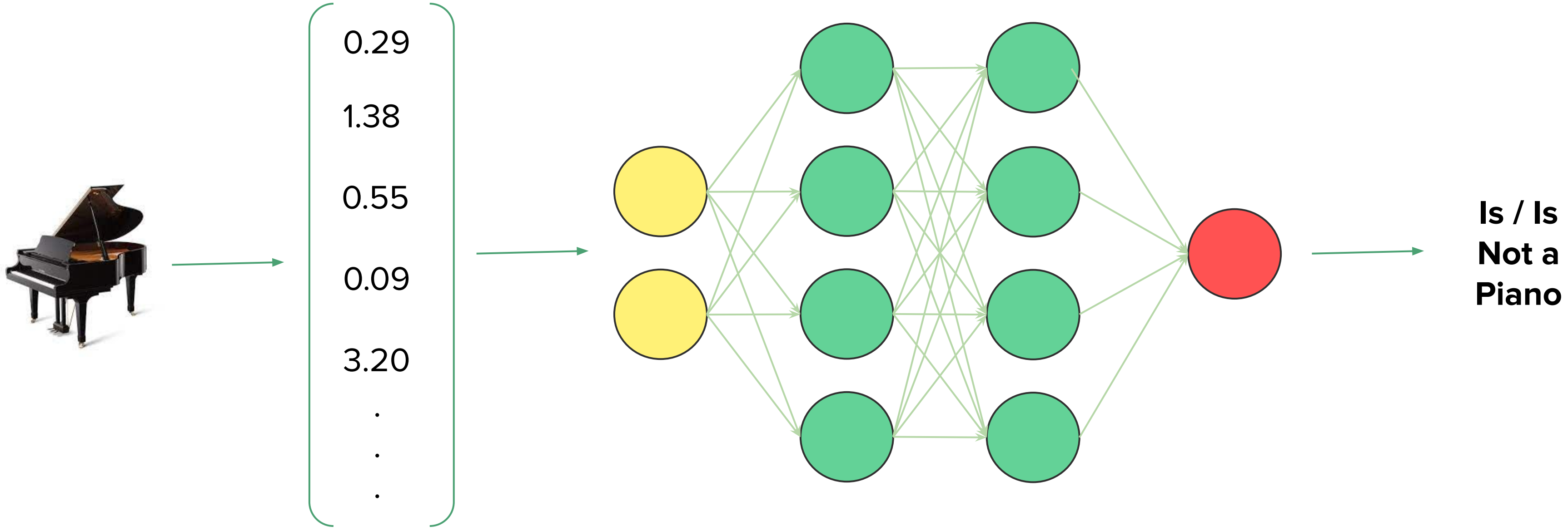


Deep Learning Model

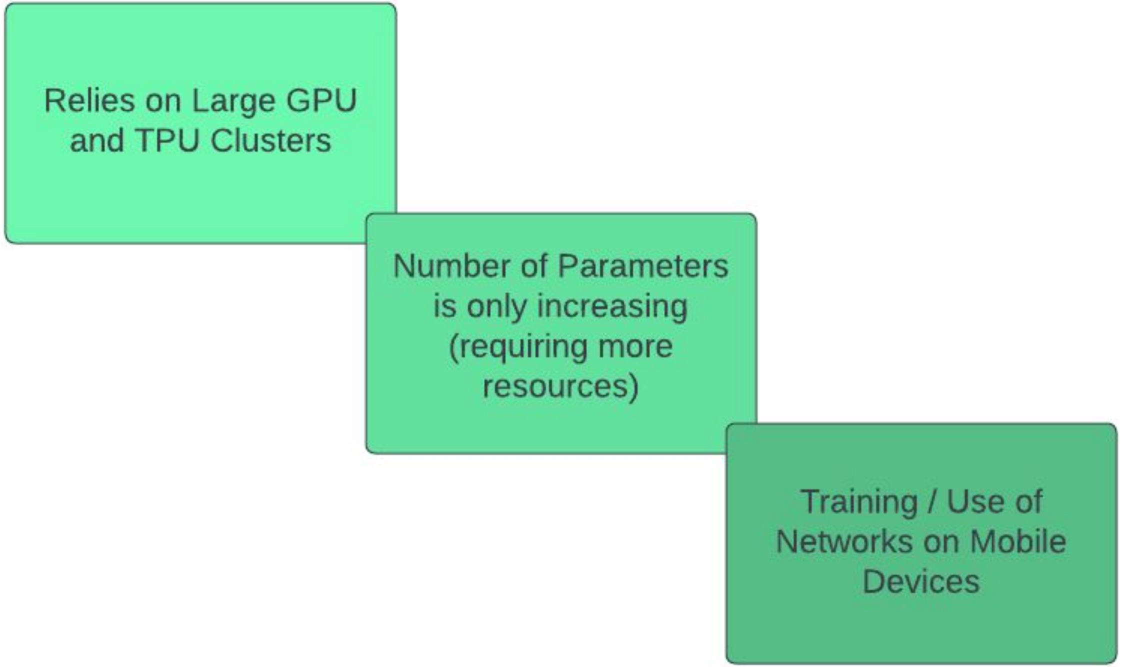


The operations with the **weights** and **biases** are what make the neural network *learn*.

Deep Learning Model



Current Problems With Deep Learning



Relies on Large GPU and TPU Clusters

Number of Parameters is only increasing (requiring more resources)

Training / Use of Networks on Mobile Devices

Current Approach—Value Quantization

(2016) Itay Hubara, Matthieu Courbariaux, et al.

Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations

Current Approach—Value Quantization

(2016) Itay Hubara, Matthieu Courbariaux, et al.

Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations

Goal:

To reduce computational resources while maintaining high accuracy

Current Approach—Value Quantization

(2016) Itay Hubara, Matthieu Courbariaux, et al.

Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations

Goal:	**Results:**
To reduce computational resources while maintaining high accuracy	Achieved *7x* faster training while maintaining accuracy compared to control (+/- 0.5%)

Current Approach—Value Quantization

(2016) Itay Hubara, Matthieu Courbariaux, et al.

Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations

Goal:	**Results:**
To reduce computational resources while maintaining high accuracy	Achieved *7x* faster training while maintaining accuracy compared to control (+/- 0.5%)

They did this with quantization!

Current Approach—Value Quantization

$$\boxed{3.2479573 * 10^{12} \quad * \quad 8.1491483 * 10^{4} \quad = \quad 2.6468086 * 10^{17}}$$

Current Approach—Value Quantization

$$3.2479573 * 10^{12} \quad * \quad 8.1491483 * 10^{4} \quad = \quad 2.6468086 * 10^{17}$$

$$3 * 10^{12} \quad * \quad 8 * 10^{4} \quad = \quad 2 * 10^{17}$$

Current Approach—Value Quantization

$$\begin{bmatrix} 1.23 & 1.23 \\ 1.23 & 1.23 \end{bmatrix} \times \begin{bmatrix} 2.34 & 2.34 \\ 2.34 & 2.34 \end{bmatrix} = \begin{bmatrix} 5.7564 & 5.7564 \\ 5.7564 & 5.7564 \end{bmatrix}$$

Current Approach—Value Quantization

$$\begin{bmatrix} 1.23 & 1.23 \\ 1.23 & 1.23 \end{bmatrix} \times \begin{bmatrix} 2.34 & 2.34 \\ 2.34 & 2.34 \end{bmatrix} = \begin{bmatrix} 5.7564 & 5.7564 \\ 5.7564 & 5.7564 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}$$

Current Approach—Value Quantization

$$\begin{bmatrix} 1.23 & 1.23 \\ 1.23 & 1.23 \end{bmatrix} \times \begin{bmatrix} 2.34 & 2.34 \\ 2.34 & 2.34 \end{bmatrix} = \begin{bmatrix} 5.7564 & 5.7564 \\ 5.7564 & 5.7564 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$$

Current Approach—Value Quantization

1.23	1.23
1.23	1.23

×

[illegible]	[illegible]

5.7564	5.7564
564	5.7564

… with +/- 0.5% difference in accuracy!

-1	-1
-1	-1

×

1	1
1	1

=

-1	-1
-1	-1

Current Approach—Value Quantization

Still directly computes matrix multiplies, which are time and resource expensive!

Our proposed improvement on quantization...

Our proposed improvement on quantization...

Space Quantization...

Our proposed improvement on quantization…

Space Quantization…

eliminates the need for matrix multiplies while training.

Lookup Table

Vectors

Matrices

	v_1	v_2	v_3
m_1	v_3	v_2	v_1
m_2	v_3	v_1	v_2

Lookup Table

Vectors

Matrices

	v_1	v_2	v_3
m_1	v_3	v_2	v_1
m_2	v_3	v_1	v_2

What is $\boldsymbol{m_2 * v_2}$?

Lookup Table

Vectors

		v_1	v_2	v_3
Matrices	m_1	v_3	v_2	v_1
	m_2	v_3	v_1	v_2

What is $\boldsymbol{m_2} * \boldsymbol{v_2}$?

Lookup Table

Vectors

Matrices

	v_1	v_2	v_3
m_1	v_3	v_2	v_1
m_2	v_3	v_1	v_2

What is $\boldsymbol{m_2} * \boldsymbol{v_2}$? → $\boldsymbol{v_1}$

Potential Improvement on Hubara/Courbariaux’s Study

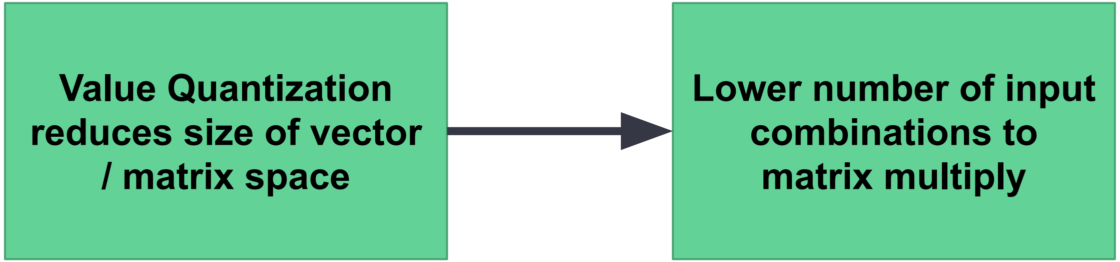
Value Quantization reduces size of vector / matrix space

Potential Improvement on Hubara/Courbariaux’s Study

Value Quantization reduces size of vector / matrix space

(by a factor of $1{,}000{,}000{,}000^d$)

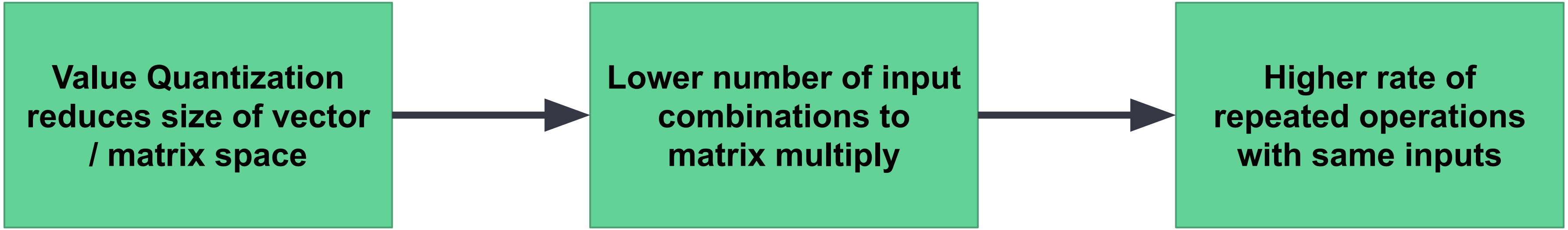
Potential Improvement on Hubara/Courbariaux’s Study



Value Quantization reduces size of vector / matrix space

Lower number of input combinations to matrix multiply

Potential Improvement on Hubara/Courbariaux's Study



Value Quantization reduces size of vector / matrix space

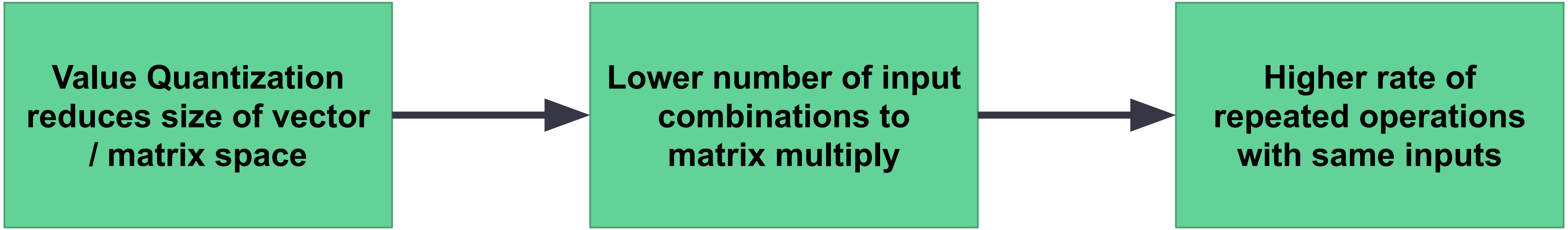
→

Lower number of input combinations to matrix multiply

→

Higher rate of repeated operations with same inputs

Potential Improvement on Hubara/Courbariaux's Study



Value Quantization reduces size of vector / matrix space

Lower number of input combinations to matrix multiply

Higher rate of repeated operations with same inputs

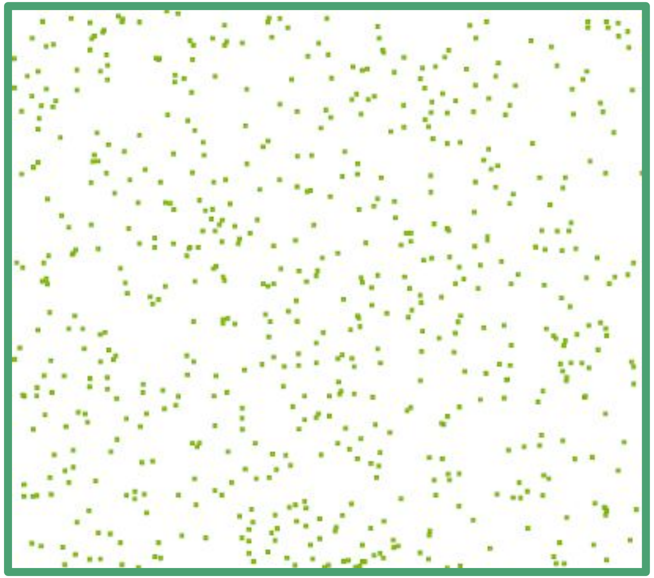
Can we make a lookup table for quantized space?

Quantizing Vector Space

Quantizing each entry still results in exponential space—too big!

Quantizing Vector Space

Can instead generate a fixed number of vectors and matrices with *randomization*



Quantizing Vector Space

Prior to running / training a deep learning model…

Quantizing Vector Space

Prior to running / training a deep learning model...

1. Define:
    - d : Dimensionality of vectors and matrices
    - N_V : Number of vectors to generate
    - N_M : Number of matrices to generate

Quantizing Vector Space

Prior to running / training a deep learning model…

1. Define:
    - d : Dimensionality of vectors and matrices
    - N_V : Number of vectors to generate
    - N_M : Number of matrices to generate
2. Generate N_v vectors of dimension d and N_M matrices of size d x d
    - Exact randomization scheme will be a component of our research
    - Example: generate entries via unit normal distribution

Quantizing Vector Space

Prior to running / training a deep learning model…

1. Define:
    - d : Dimensionality of vectors and matrices
    - N_V : Number of vectors to generate
    - N_M : Number of matrices to generate
2. Generate N_v vectors of dimension d and N_M matrices of size d x d
    - Exact randomization scheme will be a component of our research
    - Example: generate entries via unit normal distribution
3. Precompute pair-wise multiplications of each vector and matrix
    - This will result in $N_v * N_m$ multiplications

Quantizing Vector Space

Prior to running / training a deep learning model…

1. Define:
    - d : Dimensionality of vectors and matrices
    - N_V : Number of vectors to generate
    - N_M : Number of matrices to generate
2. Generate N_v vectors of dimension d and N_M matrices of size d x d
    - Exact randomization scheme will be a component of our research
    - Example: generate entries via unit normal distribution
3. Precompute pair-wise multiplications of each vector and matrix
    - This will result in $N_v * N_m$ multiplications
4. Store results in lookup table
    - Each result is mapped to its closest vector / matrix in the already generated quantized space

Example

- $d = 2$
- $N_V = 3$
- $N_M = 2$

Example

$$v_1 = \begin{bmatrix} 1.23 \\ 4.20 \end{bmatrix} \quad v_2 = \begin{bmatrix} 1.01 \\ 0.21 \end{bmatrix} \quad v_3 = \begin{bmatrix} 6.66 \\ 0.69 \end{bmatrix}$$

$$m_1 = \begin{bmatrix} 4.32 & 3.21 \\ 2.10 & 1.00 \end{bmatrix} \quad m_2 = \begin{bmatrix} 0.8 & 0.6 \\ 1.2 & 5.6 \end{bmatrix}$$

Example

$$v_1 = \begin{bmatrix} 1.23 \\ 4.20 \end{bmatrix} \quad v_2 = \begin{bmatrix} 1.01 \\ 0.21 \end{bmatrix} \quad v_3 = \begin{bmatrix} 6.66 \\ 0.69 \end{bmatrix}$$

$$m_2 \times v_2 = \ ?$$

$$m_1 = \begin{bmatrix} 4.32 & 3.21 \\ 2.10 & 1.00 \end{bmatrix} \quad m_2 = \begin{bmatrix} 0.8 & 0.6 \\ 1.2 & 5.6 \end{bmatrix}$$

Example

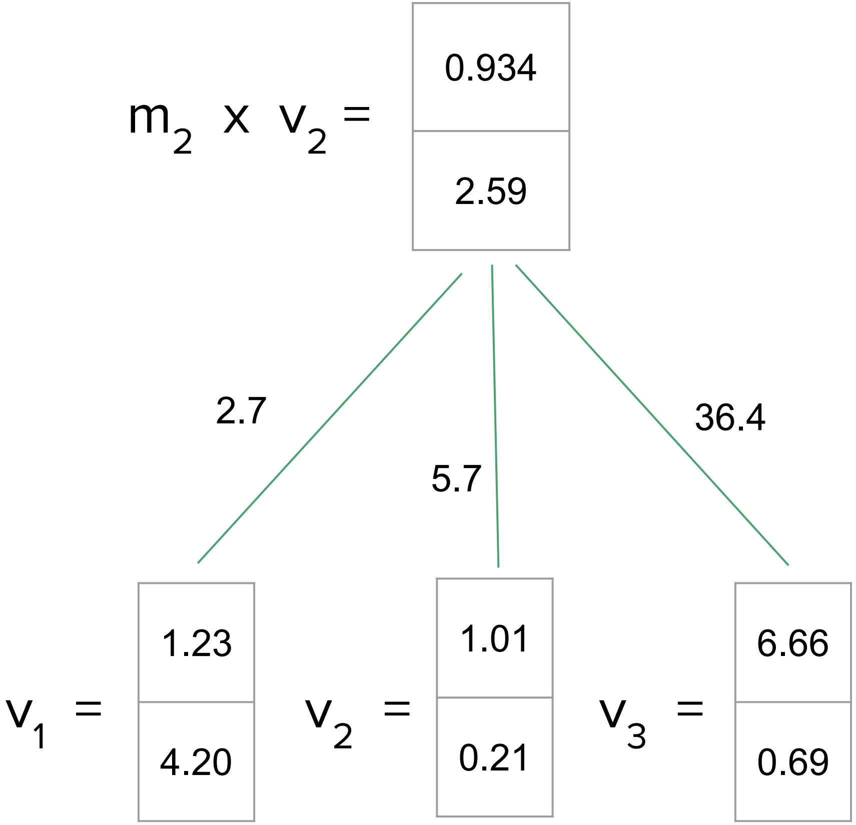
$$v_1 = \begin{bmatrix}1.23\\4.20\end{bmatrix} \quad v_2 = \begin{bmatrix}1.01\\0.21\end{bmatrix} \quad v_3 = \begin{bmatrix}6.66\\0.69\end{bmatrix}$$

$$m_1 = \begin{bmatrix}4.32 & 3.21\\2.10 & 1.00\end{bmatrix} \quad m_2 = \begin{bmatrix}0.8 & 0.6\\1.2 & 5.6\end{bmatrix}$$

$$m_2 \times v_2 = ?$$

$$\begin{bmatrix}0.8 & 0.6\\1.2 & 5.6\end{bmatrix} \times \begin{bmatrix}1.01\\0.21\end{bmatrix} = \begin{bmatrix}0.934\\2.59\end{bmatrix}$$

Example



$m_2 \text{ x } v_2 = \begin{bmatrix} 0.934 \\ 2.59 \end{bmatrix}$

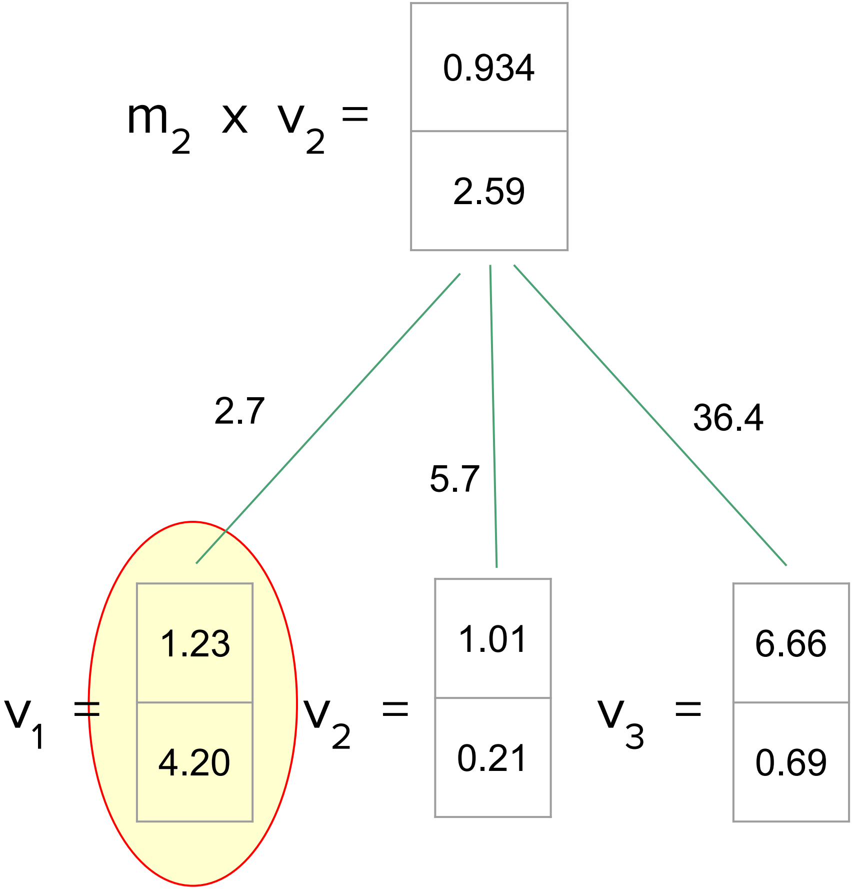
2.7

5.7

36.4

$v_1 = \begin{bmatrix} 1.23 \\ 4.20 \end{bmatrix}$ $v_2 = \begin{bmatrix} 1.01 \\ 0.21 \end{bmatrix}$ $v_3 = \begin{bmatrix} 6.66 \\ 0.69 \end{bmatrix}$

Example



$$m_2 \times v_2 = \begin{bmatrix} 0.934 \\ 2.59 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 1.23 \\ 4.20 \end{bmatrix} \quad v_2 = \begin{bmatrix} 1.01 \\ 0.21 \end{bmatrix} \quad v_3 = \begin{bmatrix} 6.66 \\ 0.69 \end{bmatrix}$$

Lookup Table

Vectors

Matrices

	v_1	v_2	v_3
m_1			
m_2		v_1	

$$m_2 * v_2 \longrightarrow v_1$$

Potential Improvements by Space Quantization

Potential Improvements by Space Quantization

Space Quantization may provide a 19x improvement in FLOPS!

Potential Improvements by Space Quantization

- Reference: A Tesla V100 GPU can provide 14 TFLOPS

Potential Improvements by Space Quantization

- Reference: A Tesla V100 GPU can provide 14 TFLOPS
- Set d=256, N_V=10K, N_M=100K, this gives the equivalent of 263 TFLOPS

Potential Improvements by Space Quantization

- Reference: A Tesla V100 GPU can provide 14 TFLOPS
- Set d=256, N_V=10K, N_M=100K, this gives the equivalent of 263 TFLOPS
- 263 TFLOPS is about **19x** 14 TFLOPS

Potential Improvements by Space Quantization

- Reference: A Tesla V100 GPU can provide 14 TFLOPS
- Set d=256, N_V=10K, N_M=100K, this gives the equivalent of 263 TFLOPS
- 263 TFLOPS is about **19x** 14 TFLOPS
- A V100 costs at least $6000

Potential Improvements by Space Quantization

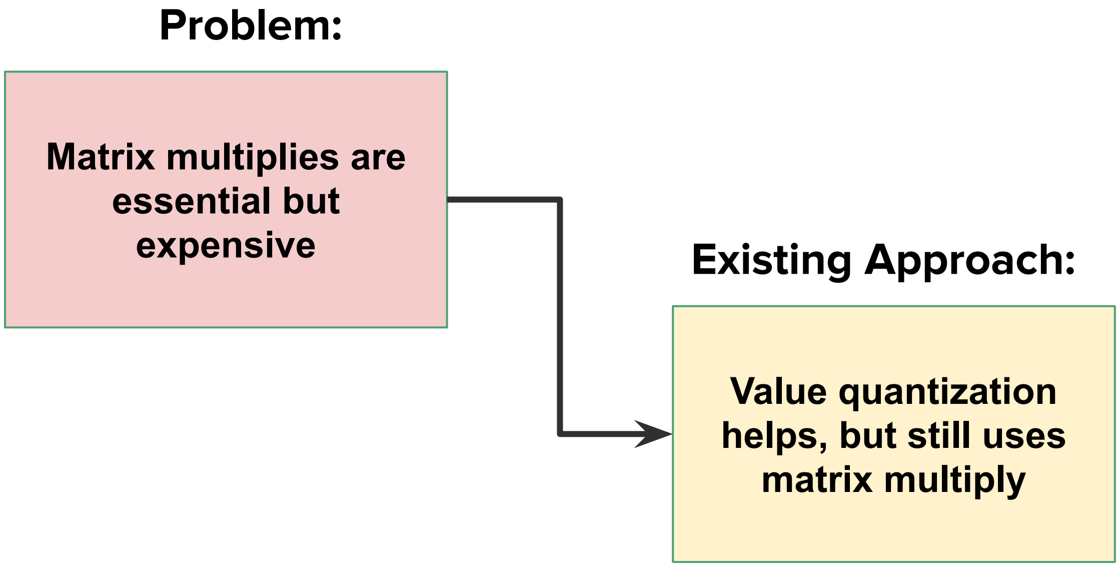
- Reference: A Tesla V100 GPU can provide 14 TFLOPS
- Set d=256, N_V=10K, N_M=100K, this gives the equivalent of 263 TFLOPS
- 263 TFLOPS is about **19x** 14 TFLOPS
- A V100 costs at least $6000
- This method could allow an old 2010 laptop to out-compute a $60K GPU cluster

Summary of Solution

Problem:

Matrix multiplies are essential but expensive

Summary of Solution



Problem:

Matrix multiplies are essential but expensive

Existing Approach:

Value quantization helps, but still uses matrix multiply

Summary of Solution

Problem:

Matrix multiplies are essential but expensive

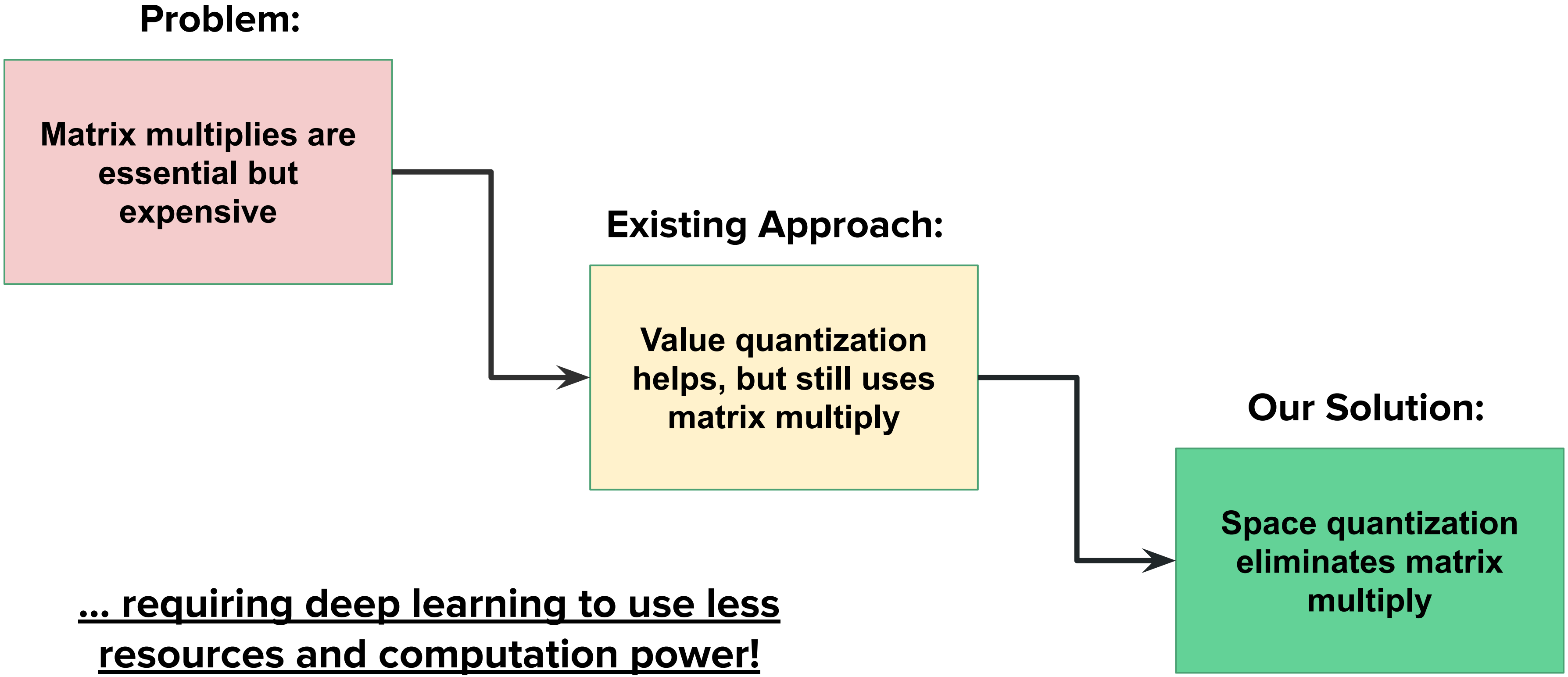
Existing Approach:

Value quantization helps, but still uses matrix multiply

Our Solution:

Space quantization eliminates matrix multiply

Summary of Solution



Problem:

Matrix multiplies are essential but expensive

Existing Approach:

Value quantization helps, but still uses matrix multiply

Our Solution:

Space quantization eliminates matrix multiply

<u>… requiring deep learning to use less resources and computation power!</u>

Proposed Timeline

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Implement Lookup Table Computation	⬛						
Change Learning Alg to Use Lookup Tables		⬛					
Revise Lookup Table Parameters and Generation				🟪	🟪	🟪	🟪
Design Synthetic Datasets			🟥	🟥			
Evaluate on Synthetic Datasets				🟥			
Evaluate on MNIST					🟨		
Evaluate on CIFAR10						🟦	🟦

Challenges to Space Quantization

Scaling up to more complex tasks	High cost to precomputation of lookup tables	Determining whether parameters need readjustment or method itself is infeasible
Go back to implementation of the framework, trying different quantization/generation schemes	Buy cloud computing power. Be more frugal with exploring hyperparameter space.	Design a task that's more complex than something already solved but less complex than what wasn't able to be solved

Thank you!

Potential Improvements by Space Quantization

- Set d=256, N_V=10K, N_M=100K, this gives the equivalent of 263 TFLOPS
    - Lookup tables size will be on the order of $10^4 * 10^5$ = 4GB
    - A single matrix multiply is 256 * 256 + 256 = 65792 FLOPs
    - On a 4GHz CPU, can do $4 * 10^9$ accesses per second
    - This corresponds to $65792 * 4 * 10^9 = 2.63 * 10^{14}$
    - This is equivalent to 263 TFLOPS for vector-matrix multiplication

Potential Extension: Genetic Algorithm

- Maybe matrix-multiply table is not optimal
- Can apply genetic algorithms to search for a better lookup table
- Genetic algorithm:
    - Fitness Function
    - Genetic Operator