Google Developer Student Clubs

# Neural Network

## with TensorFlow

```
import tensorflow as tf
```

Yeow Zi Qin
Year 2 AI Student

Yap Yoong Siew
@Future_Rainbow

# Teaser

## What we are going to learn?

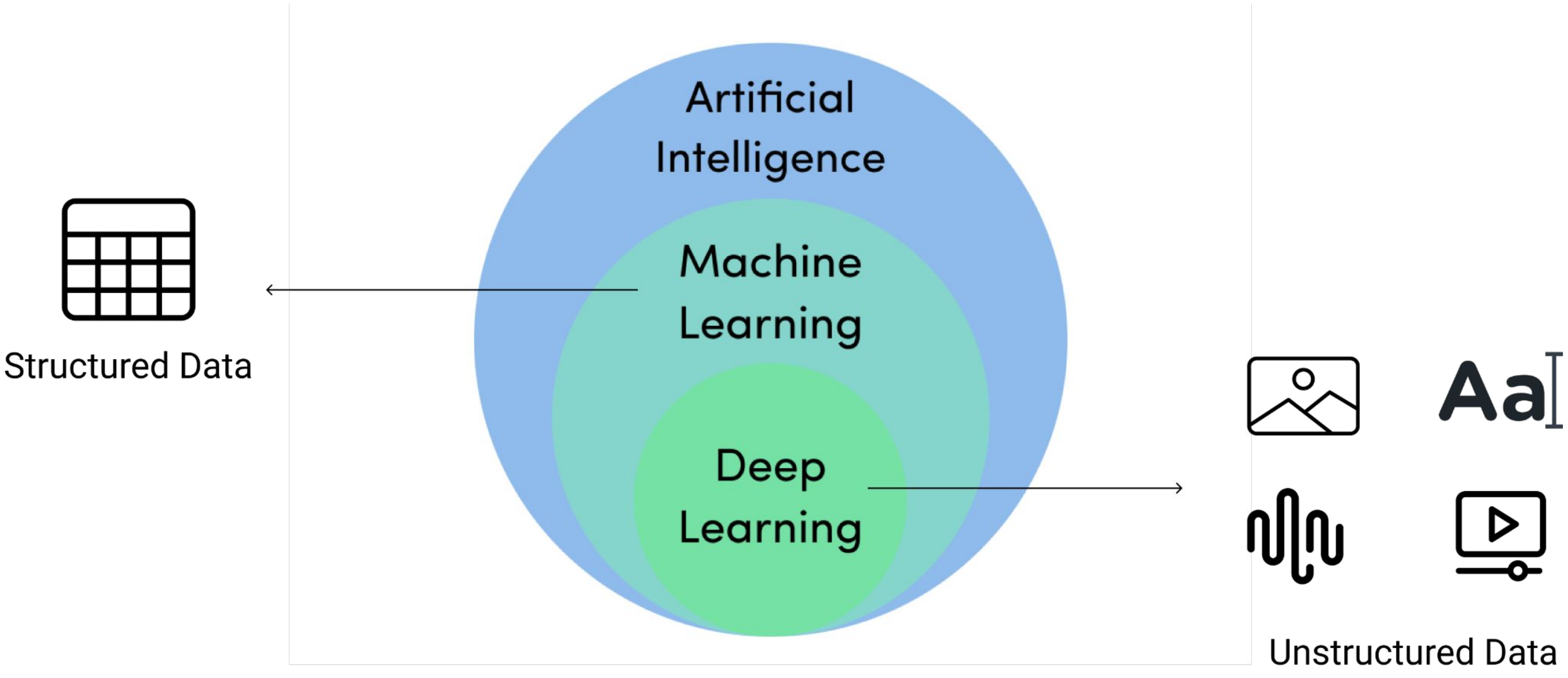
```
tensor = tf.constant(123)
```

- Introduction to **tensor**
- Preprocess and load data into tensors
- Build a neural network with Keras **Sequential**/**Functional** API
- Working on a small **regression** example
- Build a **multi-class classification** model
- Improve the model by **transfer learning** (EfficientNetB0)

```
model = tf.keras.Sequential([
  tf.keras.layers.Dense(1), # Input layer
  tf.keras.layers.Dense(10), # Hidden layer
  tf.keras.layers.Dense(100), # Hidden layer
  tf.keras.layers.Dense(1), # Output layer
])
```

# Before that

Let's revise something:



# Before that

Let's revise something: (commonly used algorithms)

## Machine Learning

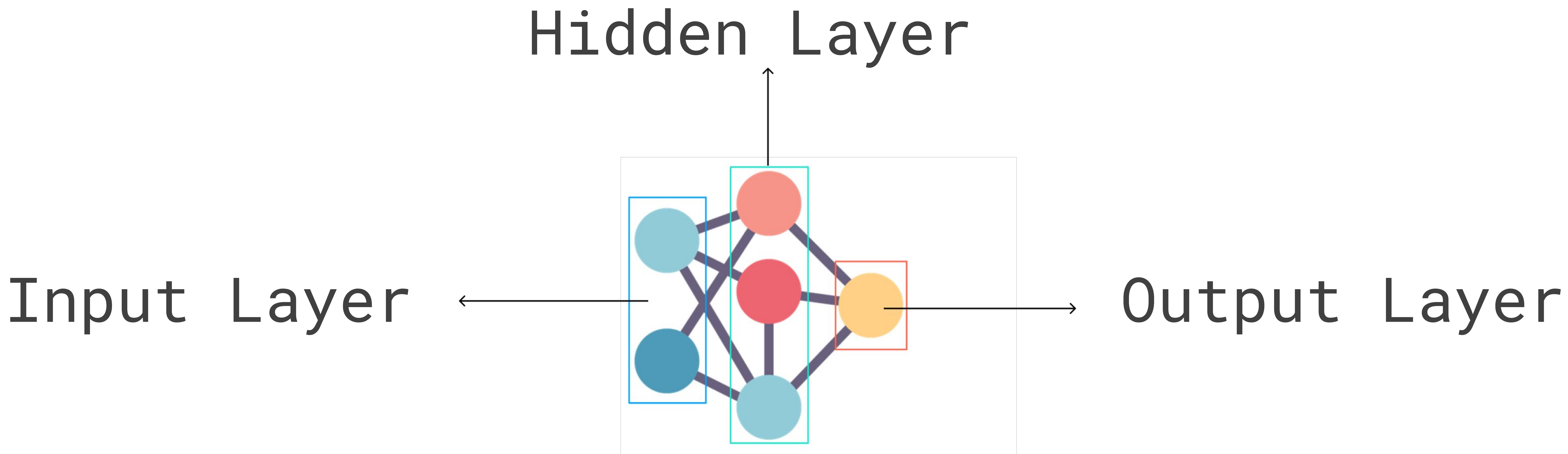
- Random forest
- Naive bayes
- Nearest neighbour
- SVM
- ...many more

## Deep Learning

- Neural networks
- Fully connected neural network
- Convolutional neural network
- Recurrent neural network
- ...many more

# Breaking down into a

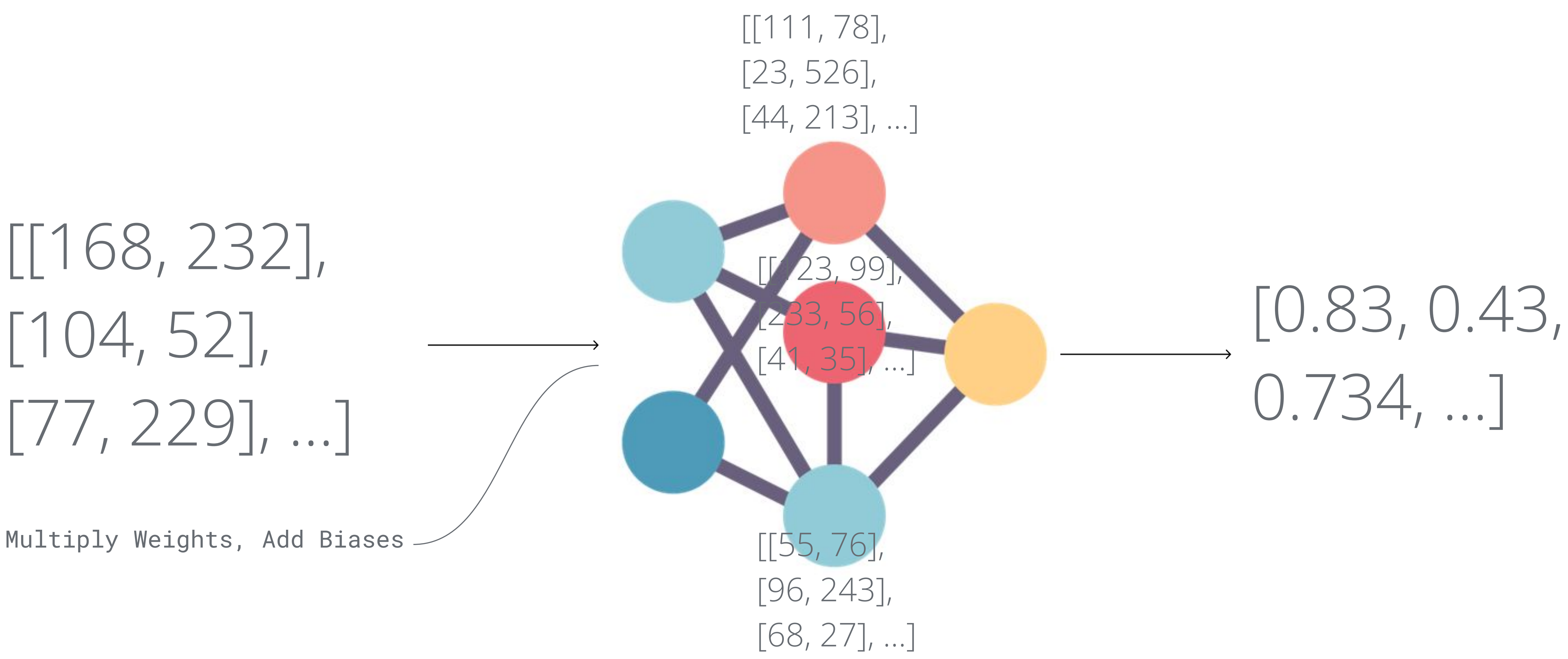
## Neural Network




```
model = tf.keras.Sequential([
  tf.keras.layers.Input(units=2), # Input layer
  tf.keras.layers.Dense(units=3), # Hidden layer
  tf.keras.layers.Dense(units=1), # Output layer
])
```

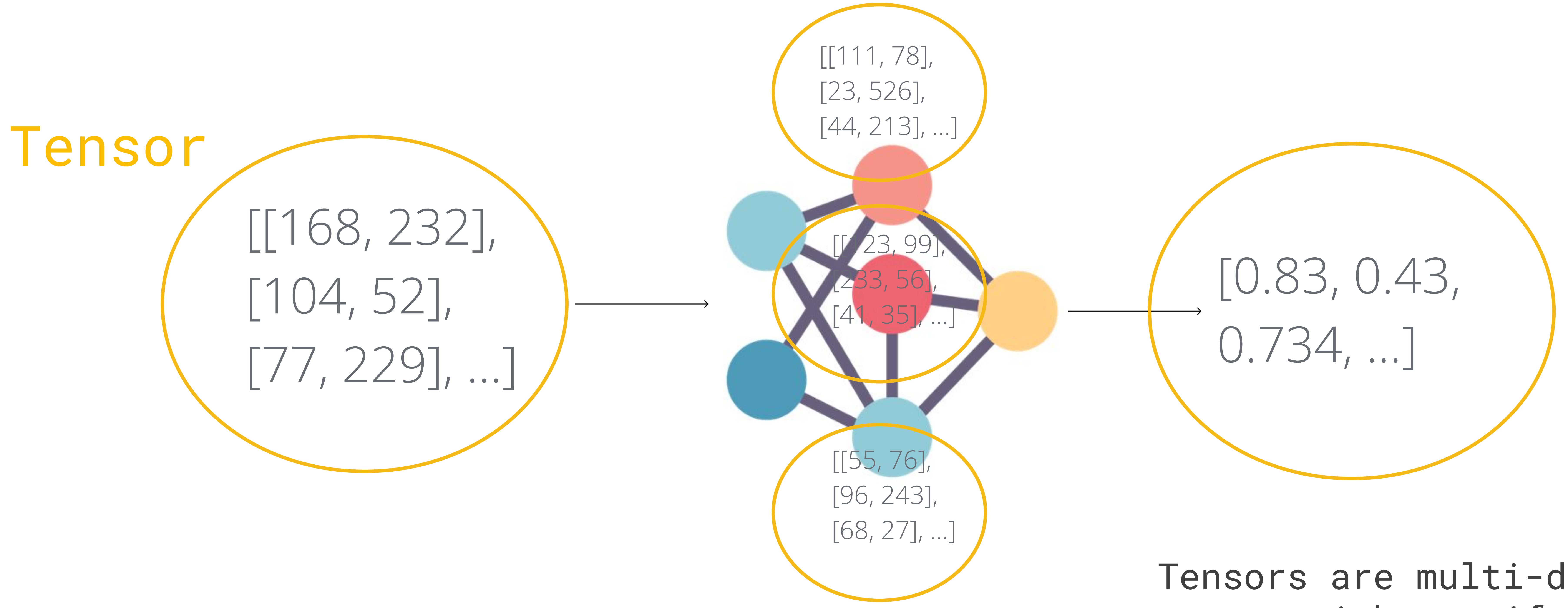
# Breaking down into a

## Neural Network



# Breaking down into a

## Neural Network




Tensors are multi-dimensional arrays with a uniform dtype

# Breaking down into a

## Neural Network

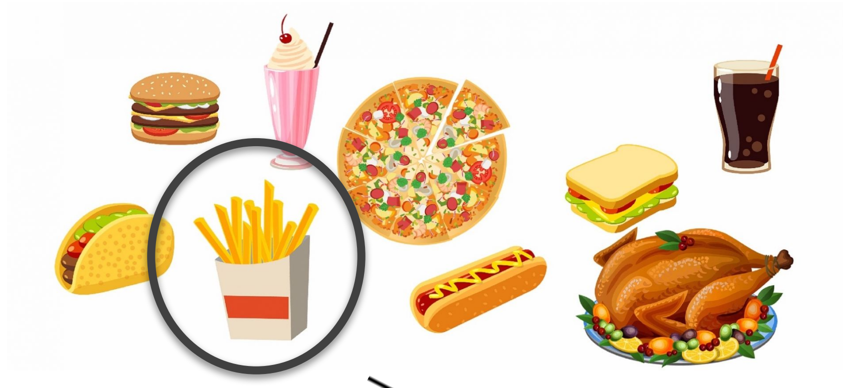
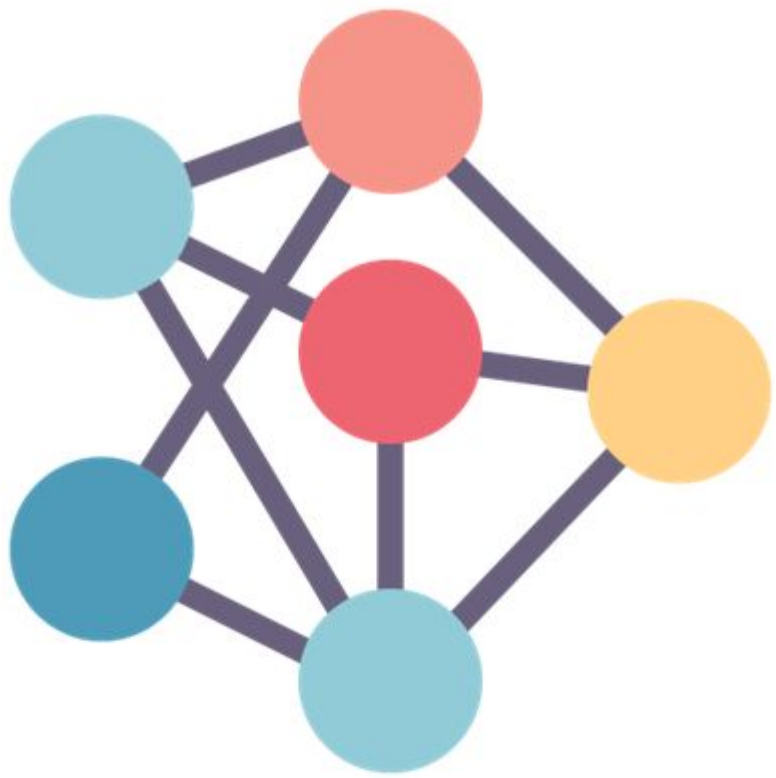


Image turns into tensors

[[168, 232],
[104, 52],
[77, 229], ...]

Fit



Output

[0.83, 0.43,
0.334, ...]

This image contain french fries with 83% confidence

# Breaking down into a

## Neural Network

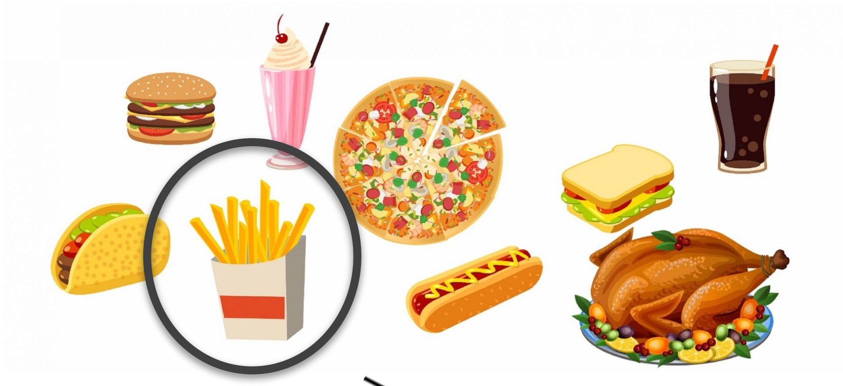


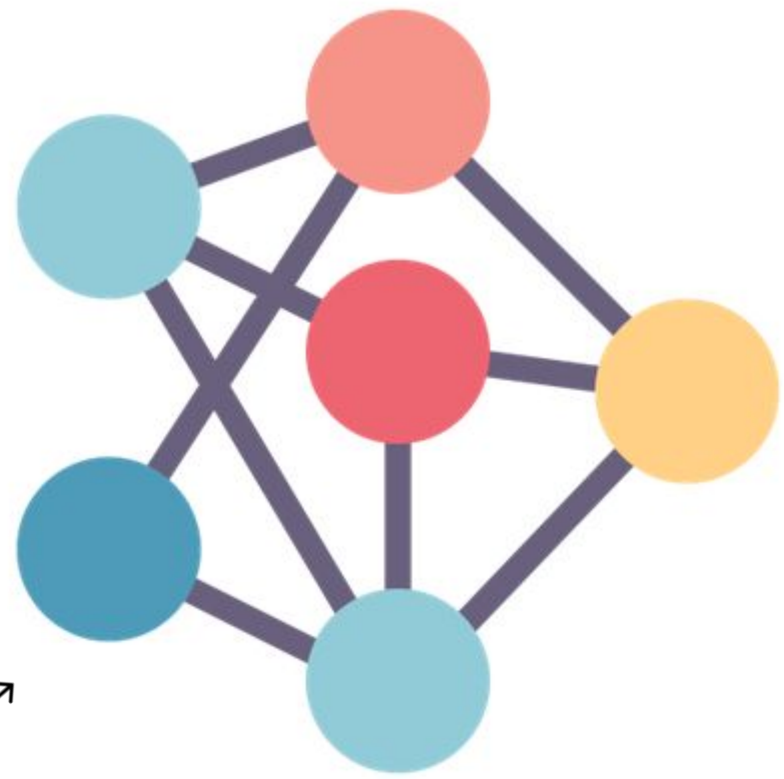
Image turns into tensors

[[168, 232],
[104, 52],
[77, 229], ...]

Normalization

[[0.3453, 0.5439],
[0.23, 0.1134],
[0.1498, 0.487], ...]

Fit



Output

[0.83, 0.43,
0.334, ...]

This image contain
french fries with
83% confidence

# Typical Workflow

`for creating a neural network`

1. Prepare the data (turn it into tensors)
2. Define the layers and units
3. Compile the model
4. Fit the data
5. Evaluate the model
6. Repeat by experimentation

# Typical Workflow

`for creating a neural network`

1. **Prepare the data (turn it into tensors)**
2. Define the layers and units
3. Compile the model
4. Fit the data
5. Evaluate the model
6. Repeat by experimentation

# Typical Workflow

## for creating a neural network

1. Prepare the data (turn it into tensors)
2. **Define the layers and units**
3. Compile the model
4. Fit the data
5. Evaluate the model
6. Repeat by experimentation

```
import tensorflow as tf

model = tf.keras.Sequential([
  tf.keras.layers.Input(3), # Input layer
  tf.keras.layers.Dense(10, activation='relu'), # Hidden layer
  tf.keras.layers.Dense(100, activation='relu'), # Hidden layer
  tf.keras.layers.Dense(2, activation='sigmoid') # Output layer
])
```

# Typical Workflow

## for creating a neural network

1. Prepare the data (turn it into tensors)
2. Define the layers and units
3. **Compile the model**
4. Fit the data
5. Evaluate the model
6. Repeat by experimentation

```
model.compile(
    loss=tf.keras.losses.binary_crossentropy,
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
)
```

# Typical Workflow

## `for creating a neural network`

1. Prepare the data (turn it into tensors)
2. Define the layers and units
3. Compile the model
4. **Fit the data**
5. Evaluate the model
6. Repeat by experimentation

```
model.fit(
  X_train,
  y_train,
  epochs=5,
  validation_data=(X_test, y_test)
)
```

# Typical Workflow

## for creating a neural network

1. Prepare the data (turn it into tensors)
2. Define the layers and units
3. Compile the model
4. Fit the data
5. **Evaluate the model**
6. Repeat by experimentation

```
model.evaluate(x_test, y_test)
```

# Typical Workflow

`for creating a neural network`

1. Prepare the data (turn it into tensors)
2. Define the layers and units
3. Compile the model
4. Fit the data
5. Evaluate the model
6. **Repeat by experimentation**

# Example

## Regression Problems

Build a neural network that can predict the **housing price** based on:

1. No. of bedroom
2. No. of bathroom
3. No. of parking slot

🛏️ x 4

🛁 x 4

🚗 x 2

```
# 1. Create a model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.Input(shape=(3,)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(1, activation=None)
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.mae,
              optimizer=tf.keras.optimizers.Adam(lr=0.0001),
              metrics=["mae"])

# 3. Fit the model
model.fit(X_train, y_train, epochs=100)
```

$ 956,000

Google Developer Student Clubs

# Cheatsheet

## Regression Problems

| Hyperparameter | Typical value |
|---|---|
| Input layer shape | Same shape as number of features (e.g. 3 for # bedrooms, # bathrooms, # car spaces in housing price prediction) |
| Hidden layer(s) | Problem specific, minimum = 1, maximum = unlimited |
| Neurons per hidden layer | Problem specific, generally 10 to 100 |
| Output layer shape | Same shape as desired prediction shape (e.g. 1 for house price) |
| Hidden activation | Usually ReLU (rectified linear unit) |
| Output activation | None, ReLU, logistic/tanh |
| Loss function | MSE (mean square error) or MAE (mean absolute error)/Huber (combination of MAE/MSE) if outliers |
| Optimizer | SGD (stochastic gradient descent), Adam |

**Source:** Adapted from page 293 of Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow Book by Aurélien Géron

🛏️ x 4

🛁 x 4

🚗 x 2

```
# 1. Create a model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.Input(shape=(3,)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(1, activation=None)
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.mae,
              optimizer=tf.keras.optimizers.Adam(lr=0.0001),
              metrics=["mae"])

# 3. Fit the model
model.fit(X_train, y_train, epochs=100)
```

$ 956,000

# Example

## Classification Problems

Build a neural network that classifies the images between:

1. Sushi
2. Steak
3. Pizza

Sushi

Steak

Pizza

```
# 1. Create a model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.Input(shape=(224, 224, 3)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(3, activation="softmax")
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])

# 3. Fit the model
model.fit(X_train, y_train, epochs=5)

# 4. Evaluate the model
model.evaluate(X_test, y_test)
```

Sushi

[[0.93, 0.13, 0.236],...]

# Example

## Classification Problems

Sushi

Steak

Pizza

| Hyperparameter | Binary Classification | Multiclass classification |
|---|---|---|
| Input layer shape | Same as number of features (e.g. 5 for age, sex, height, weight, smoking status in heart disease prediction) | Same as binary classification |
| Hidden layer(s) | Problem specific, minimum = 1, maximum = unlimited | Same as binary classification |
| Neurons per hidden layer | Problem specific, generally 10 to 100 | Same as binary classification |
| Output layer shape | 1 (one class or the other) | 1 per class (e.g. 3 for food, person or dog photo) |
| Hidden activation | Usually ReLU (rectified linear unit) | Same as binary classification |
| Output activation | Sigmoid | Softmax |
| Loss function | Cross entropy (`tf.keras.losses.BinaryCrossentropy` in TensorFlow) | Cross entropy (`tf.keras.losses.CategoricalCrossentropy` in TensorFlow) |
| Optimizer | SGD (stochastic gradient descent), Adam | Same as binary classification |

**Source:** Adapted from page 293 of Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow Book by Aurélien Géron

```
# 1. Create a model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.Input(shape=(224, 224, 3)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(3, activation="softmax")
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])

# 3. Fit the model
model.fit(X_train, y_train, epochs=5)

# 4. Evaluate the model
model.evaluate(X_test, y_test)
```

Sushi

[[0.93, 0.13, 0.236],...]

# Cheatsheet

## Classification Problems

| Hyperparameter | Typical value |
|---|---|
| Input layer shape | Same shape as number of features (e.g. 3 for # bedrooms, # bathrooms, # car spaces in housing price prediction) |
| Hidden layer(s) | Problem specific, minimum = 1, maximum = unlimited |
| Neurons per hidden layer | Problem specific, generally 10 to 100 |
| Output layer shape | Same shape as desired prediction shape (e.g. 1 for house price) |
| Hidden activation | Usually ReLU (rectified linear unit) |
| Output activation | None, ReLU, logistic/tanh |
| Loss function | MSE (mean square error) or MAE (mean absolute error)/Huber (combination of MAE/MSE) if outliers |
| Optimizer | SGD (stochastic gradient descent), Adam |

**Source:** Adapted from page 293 of Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow Book by Aurélien Géron

🛏️ x 4

🛁 x 4

🚗 x 2

```python
# 1. Create a model (specified to your problem)
model = tf.keras.Sequential([
  tf.keras.Input(shape=(3,)),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(100, activation="relu"),
  tf.keras.layers.Dense(1, activation=None)
])

# 2. Compile the model
model.compile(loss=tf.keras.losses.mae,
              optimizer=tf.keras.optimizers.Adam(lr=0.0001),
              metrics=["mae"])

# 3. Fit the model
model.fit(X_train, y_train, epochs=100)
```

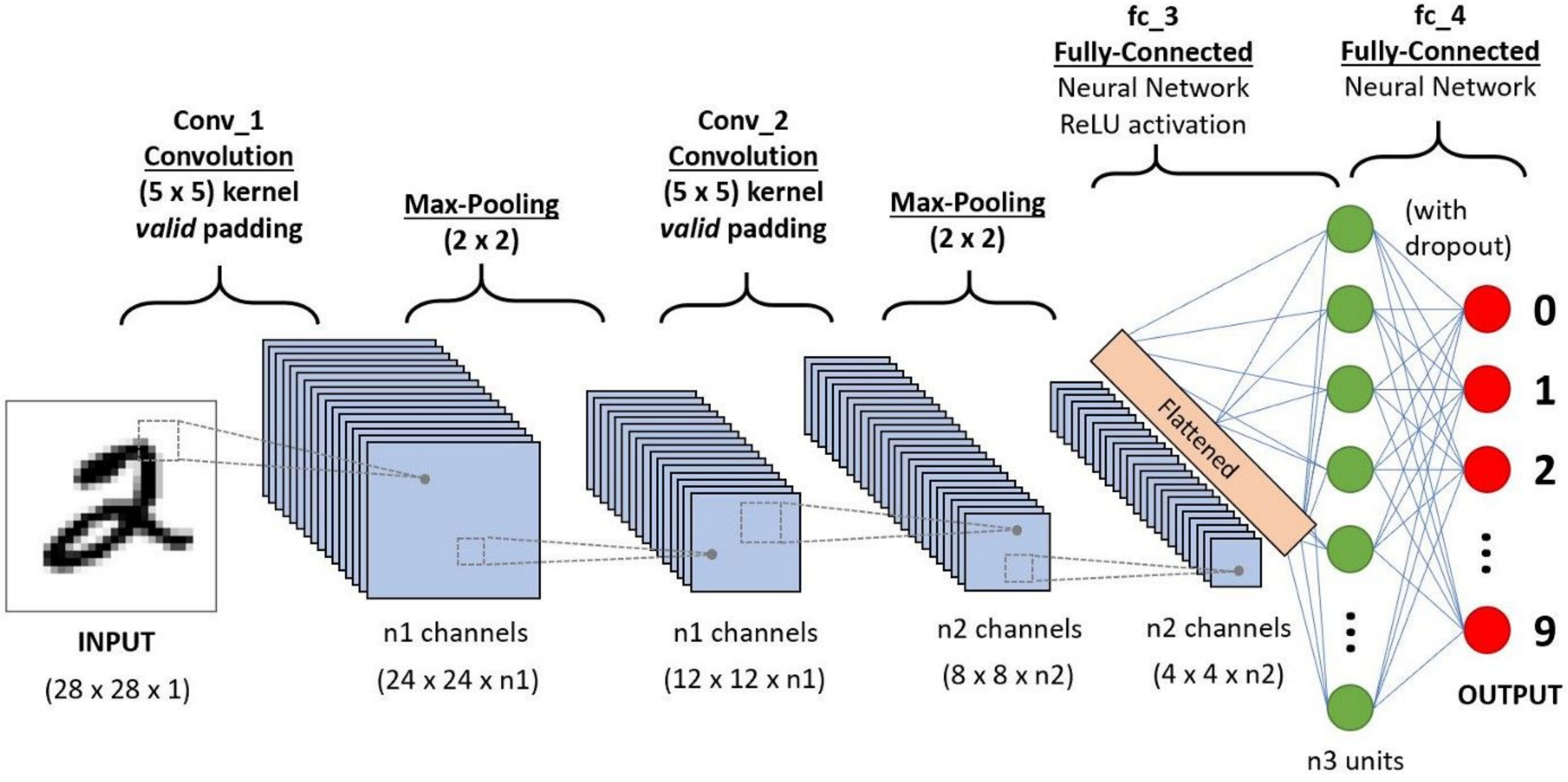
$ 956,000

# Desmos Graph Explanation

Visualize Activation Functions

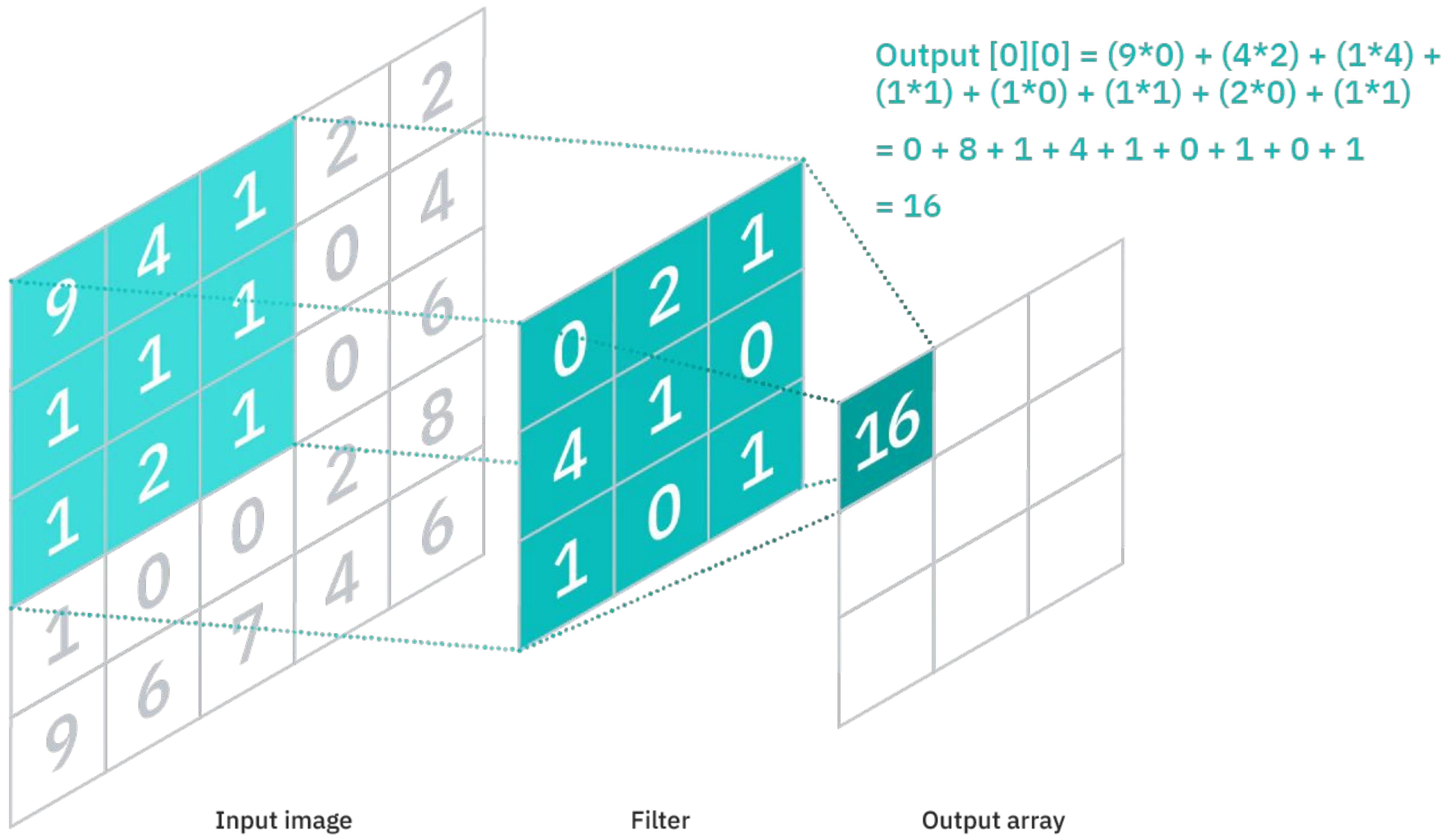
https://www.desmos.com/calculator/drqqhtb037

# Convolution Neural Network (CNN)



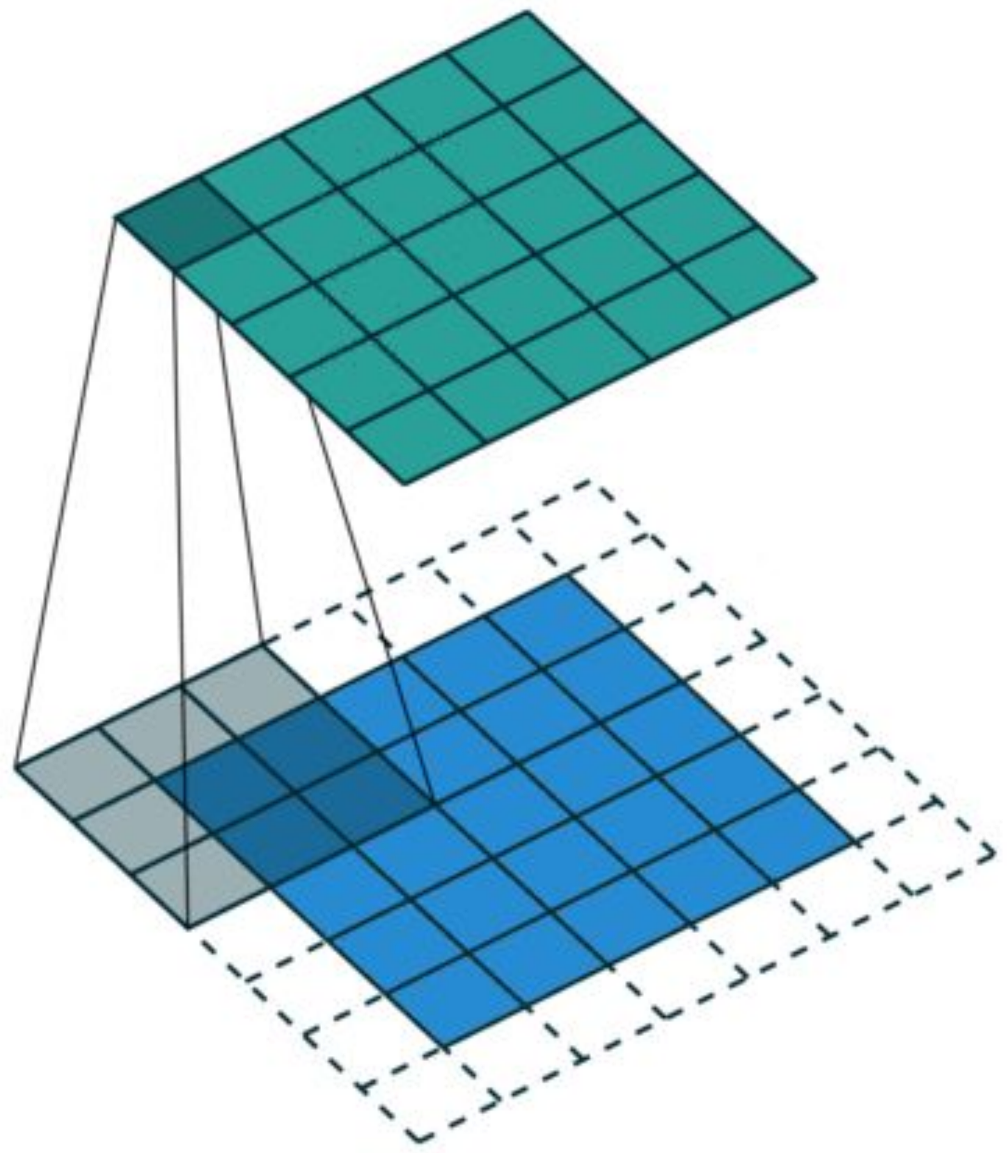
# Conv2D Layer

`tf.keras.layers.Conv2D` or `tf.keras.layers.Convolution2D`



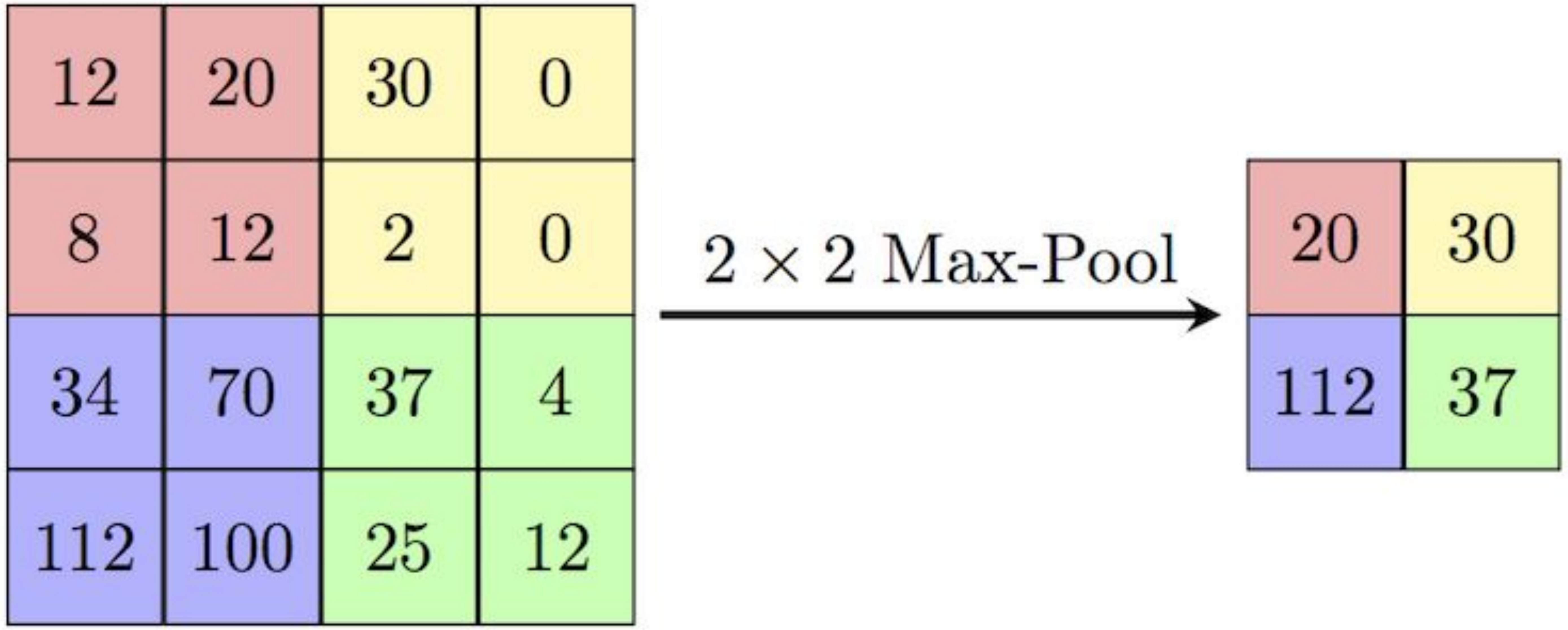
# Conv2D Layer

`tf.keras.layers.Conv2D` or `tf.keras.layers.Convolution2D`



# MaxPool2D Layer

`tf.keras.layers.MaxPool2D`

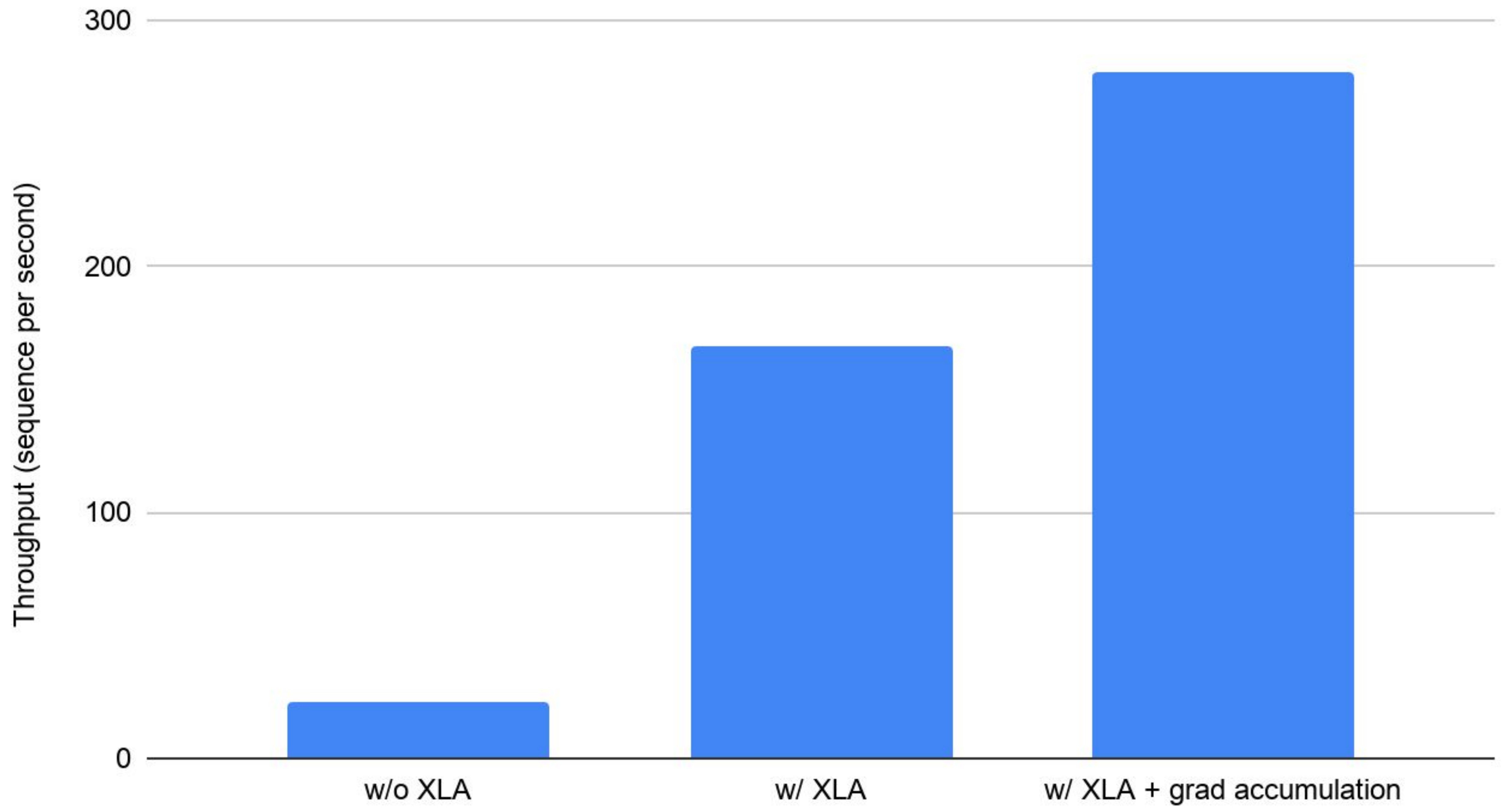


# Where to go from here?

Further ML topics to explore

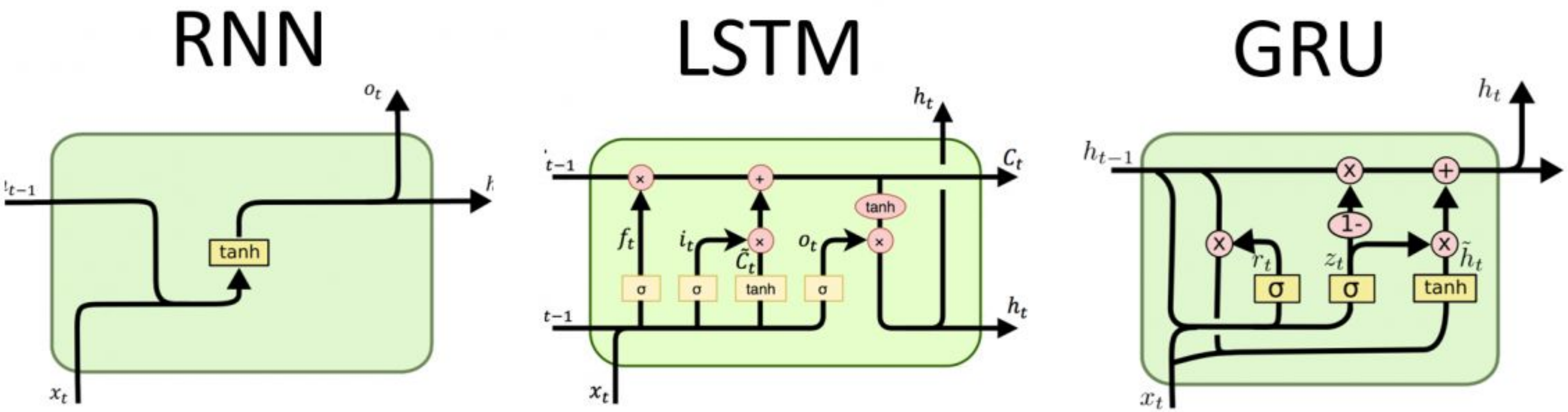
# XLA Tensorflow

## Accelerated Linear Algebra

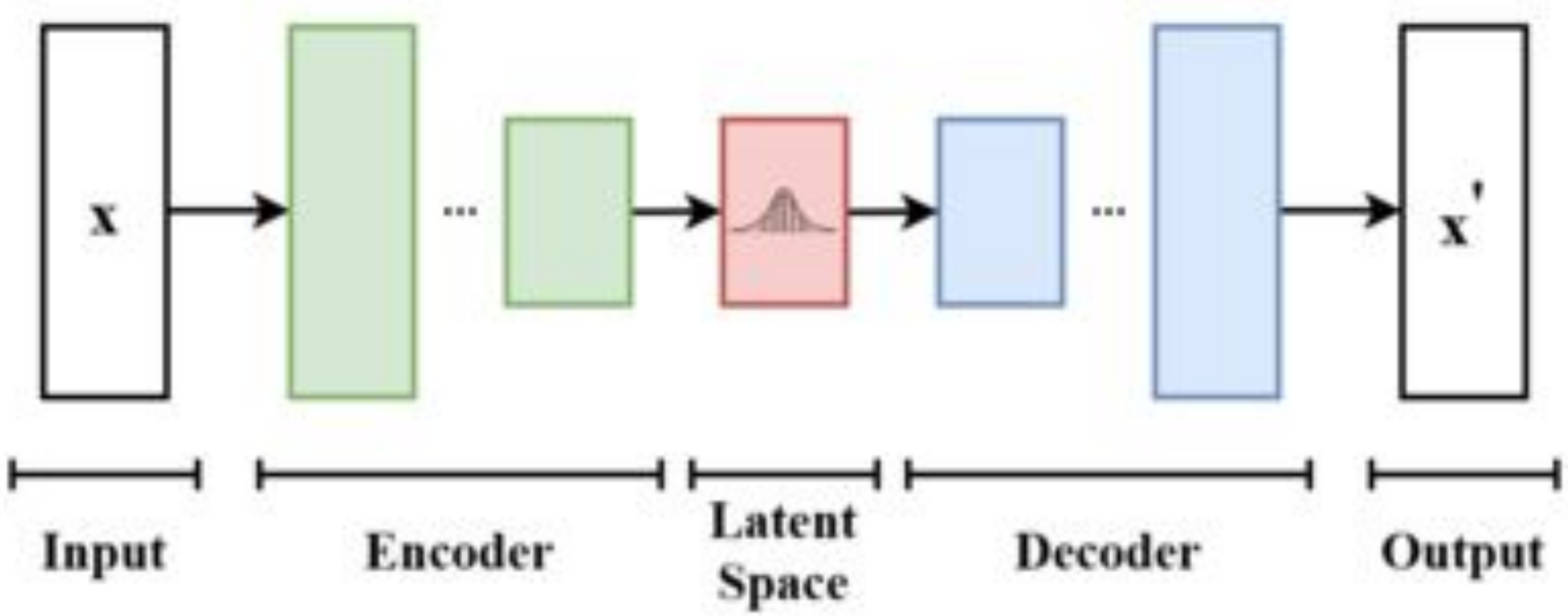


# RNN/LSTM/GRU

Different types of neural network architectures

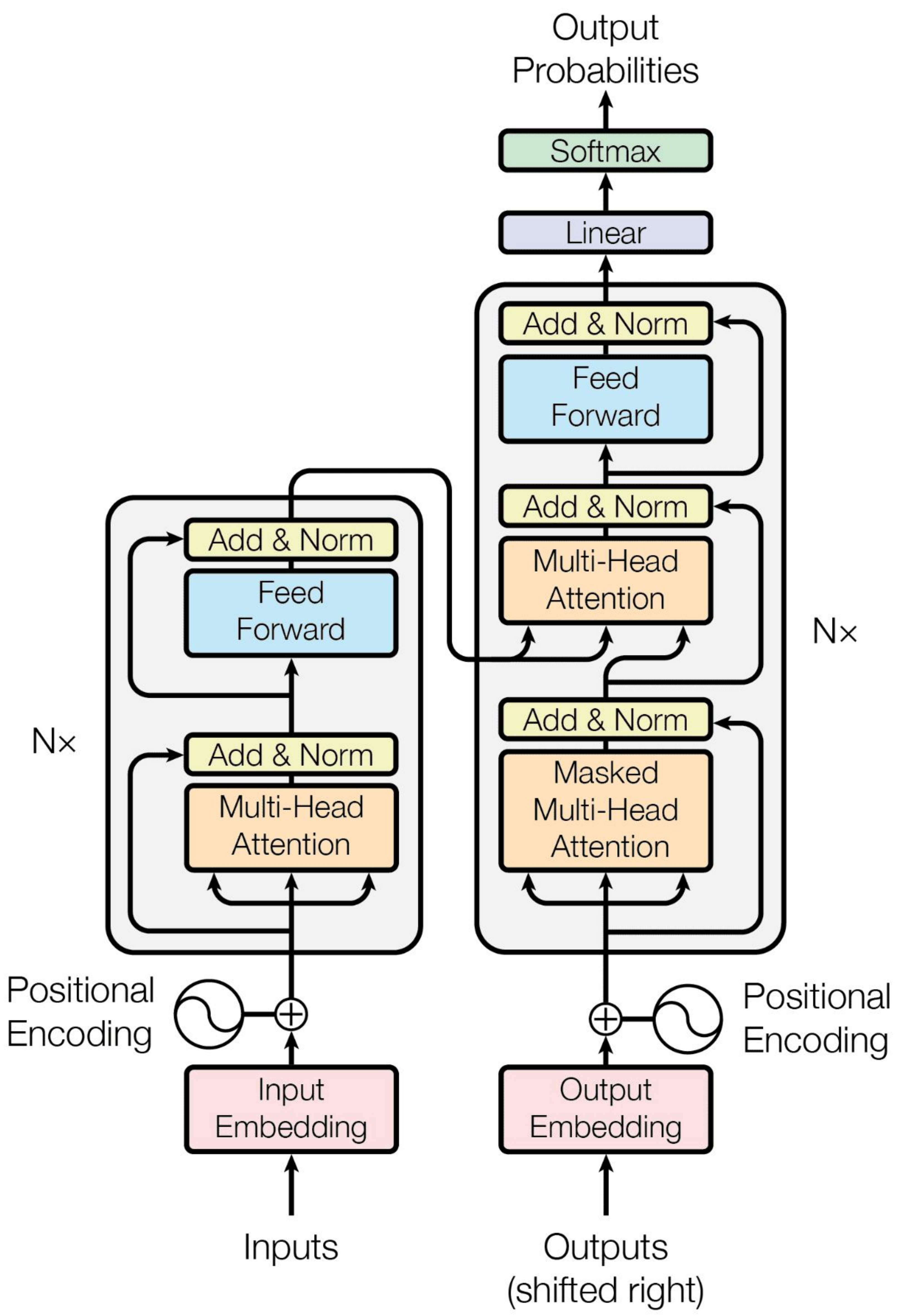


# Variational Autoencoder

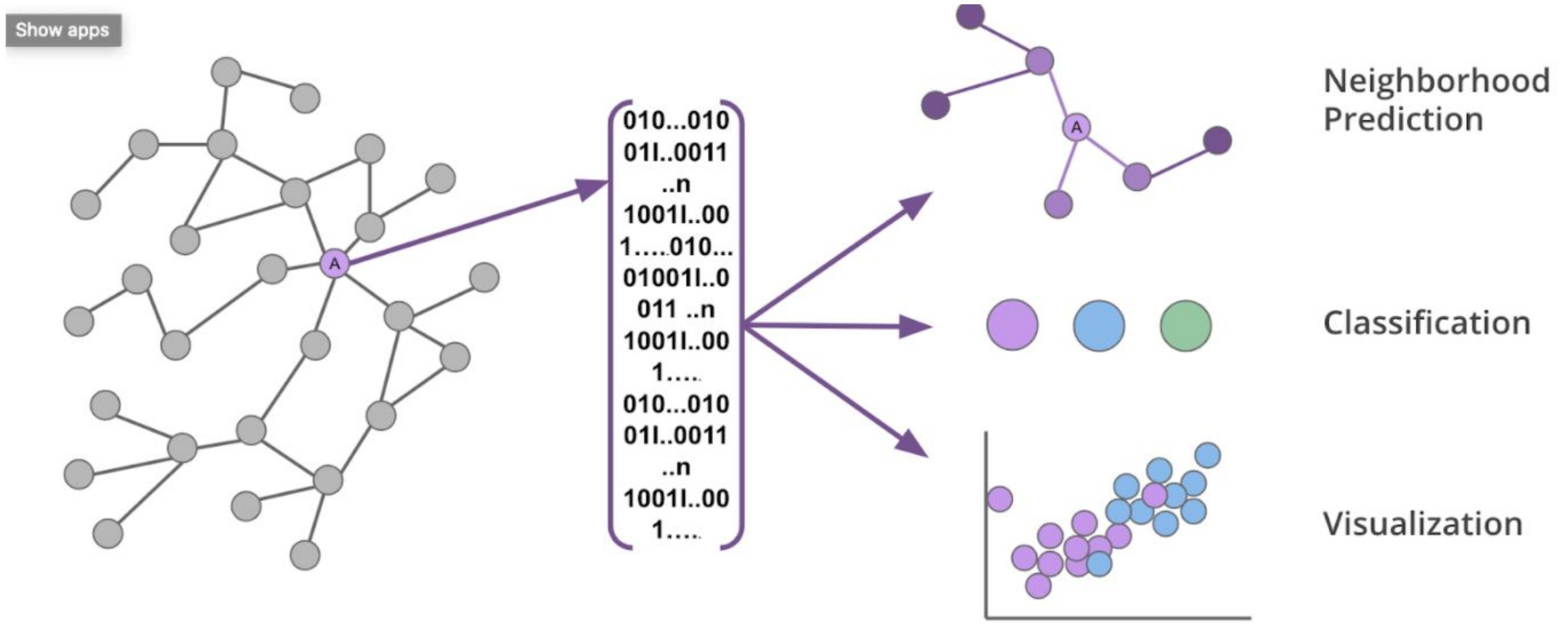


# Transformers

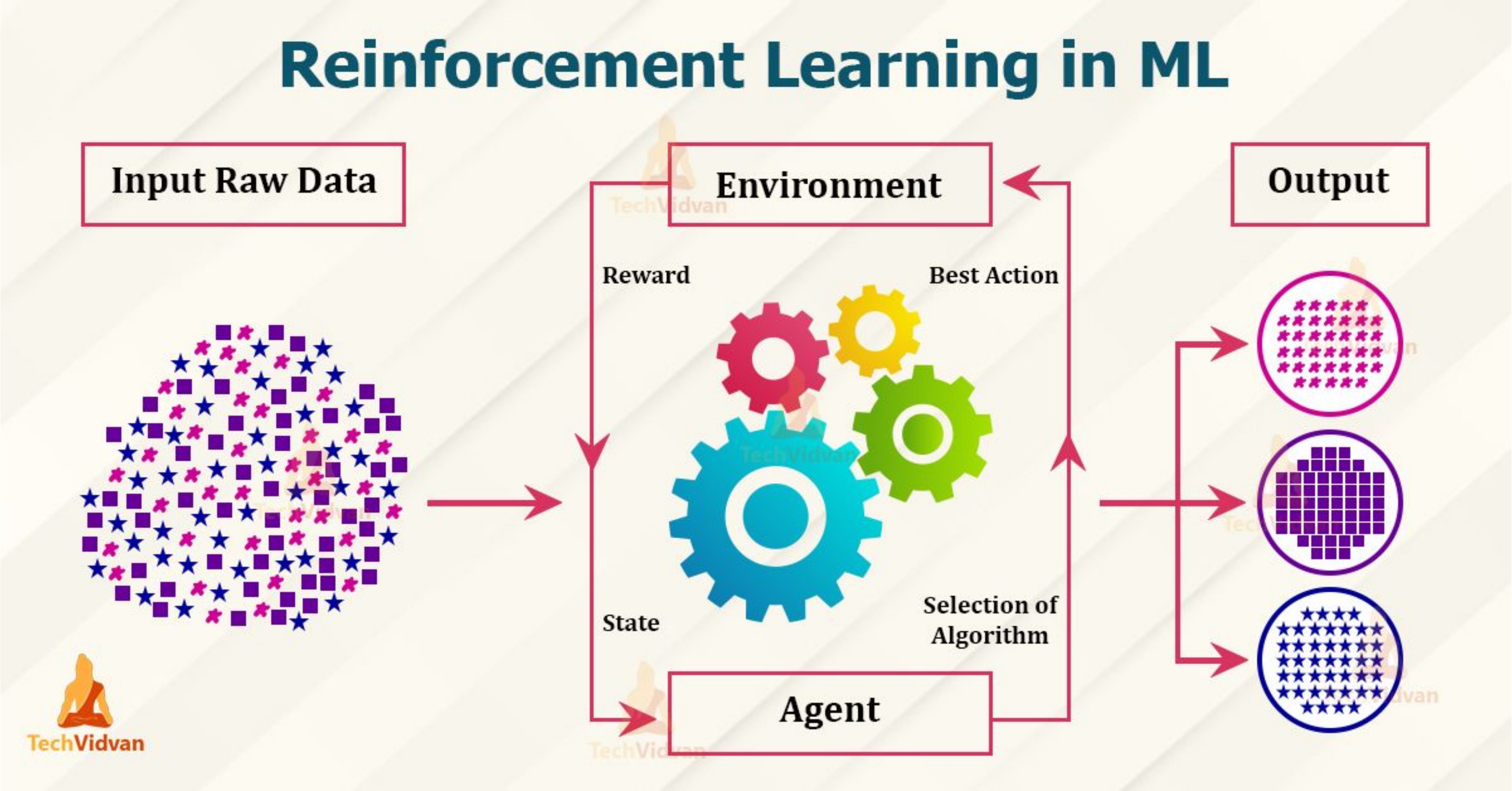
Attention is all you need



# Graph Machine Learning

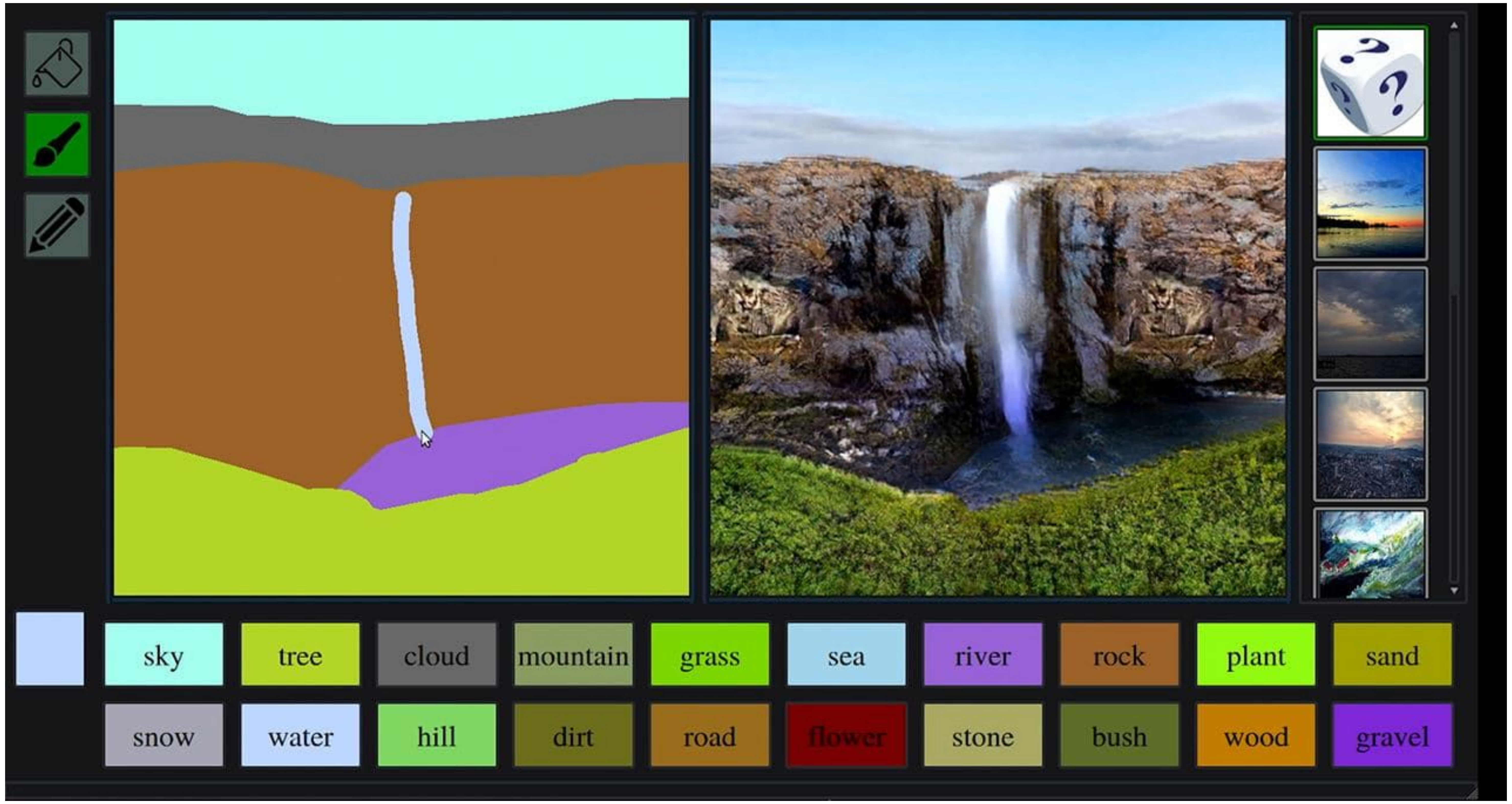


# Reinforcement Learing



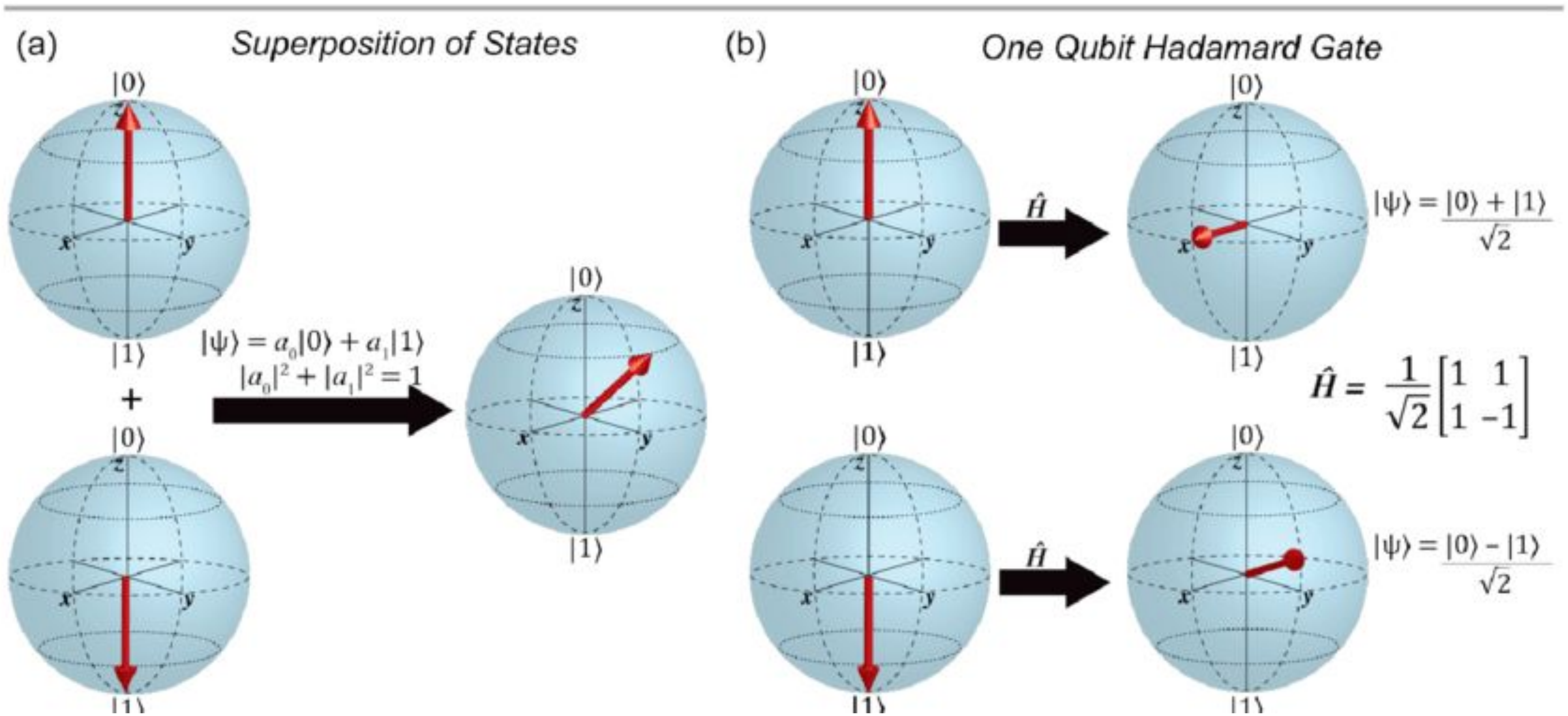
# GANs

Generative Adversarial Nets



# Quantum Machine Learning

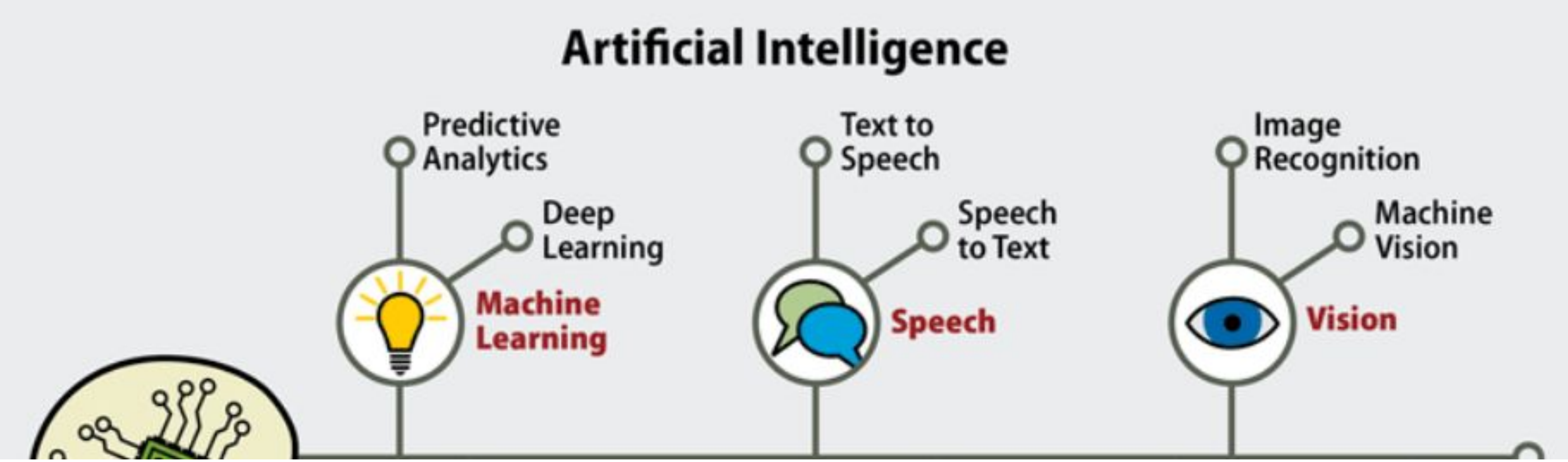
## Quantum Computing and Machine Learning



# Ethics in A.I.

# ML Career Planning

Choose a domain to focus



Elon Musk
@elonmusk

Deus ex machine learning


6:07 AM · Nov 18, 2020 · Twitter for iPhone

13.8K Retweets 2,382 Quote Tweets 183K Likes