World Academy of Science, Engineering and Technology
International Journal of Biotechnology and Bioengineering
Vol:18, No:9, 2024

Codes beyond Bits and Bytes: A Blueprint for Artificial Life

Rishabh Garg, Anuja Vyas, Aamna Khan, Muhammad Azwan Tariq

Abstract**—**The present study focuses on integrating Machine Learning and Genomics, hereafter termed 'GenoLearning', to develop Artificial Life (AL). This is achieved by leveraging gene editing to imbue genes with sequences capable of performing desired functions. To accomplish this, a specialized sub-network of Siamese Neural Network (SNN), named Transformer Architecture specialized in Sequence Analysis of Genes (TASAG), compares two sequences: the desired and target sequences. Differences between these sequences are analyzed, and necessary edits are made on-screen to incorporate the desired sequence into the target sequence. The edited sequence can then be synthesized chemically using a Computerized DNA Synthesizer (CDS). The CDS fabricates DNA strands according to the sequence displayed on a computer screen, aided by microprocessors. These synthesized DNA strands can be inserted into an ovum to initiate further development, eventually leading to the creation of an Embot, and ultimately, an H-Bot. While this study aims to explore the potential benefits of Artificial Intelligence (AI) technology, it also acknowledges and addresses the ethical considerations associated with its implementation.

Keywords**—**Machine Learning, Genomics, Genetronics, DNA, Transformer, Siamese Neural Network, Gene Editing, Artificial Life, H-Bot, Zoobot.

I. Introduction

As modern technology advances into the territories once confined to science fiction, the fusion of Genetic Engineering and Artificial Intelligence opens doors to the next step of advancement: creating Computerized Human-Bot Entities (H-bots) and animal-bot entities (Zoobots) [1]. Coined as 'Genetronics' by [2] and [3], this technology aims to integrate the branch of Genomics with that of Artificial Intelligence and Mechatronics, creating an entity that possesses human-like physical, chemical, and biological attributes while operating functionally as robots. Genomics, the study of an organism's complete genetic makeup encoded in DNA, serves as the foundation for this transformative leap.

During translation, three sets of nitrogenous bases of DNA, known as codons, are responsible for gene expression via protein synthesis. If the function of each of these codons is worked out, attributes of an individual that correspond to their genetic make-up can be manipulated. With the help of Machine Learning and chemical synthesis processes, the creation of AL may be possible using this concept. For instance, an individual is required with the necessity of a particular trait 'A'. 'A' is coded by gene 'X'. If this gene is not pre-existent or is under-expressed in a normal human genome, it can be sufficiently expressed via artificial methods and thus enable the newly created individual to possess this trait. If trait 'B' controlled by gene 'Y' needs to be eliminated, the DNA can be designed such that 'Y' is not expressed.

We suppose that a geneticist requires the creation of a human entity that has an athletic build. To customize such a human, he would first locate the position of genes responsible for good athletics. Let it be called 'the athlete gene'. Next, he needs to edit the genome in order to sufficiently express the athlete gene and then chemically synthesize it. Thereafter, he will require a human ovum which comprises of all cytoplasmic factors required for growth and development of a zygote. Since a zygote is the product of fertilization, the term 'Zybot' can be used to refer to the zygotic stage of the H-Bot. Similarly, the embryo can be referred to as 'Embot'.

II. Proposed Methodology

A. Gene Sequencing

In order to implement the concept, Sanger sequencing or Next Generation Sequencing (NGS) methods like Pyro-sequencing, Ion semiconductor sequencing, etc. would be employed to determine the order of nucleotides within a DNA molecule. Genomic files can be obtained after using any of the present-day sequencing methods. For each chromosome of the organism, separate files would be created and stored. File types like GFF3 or GTF can be used to store these data for gene annotation [4]. This would enable the demarcation of protein coding and non-protein coding regions making it easier and quicker to train the ML model based on TASAG.

B. Computerized Gene Sequencing

In case of the absence of a genetic sequence, AI can be used to generate one using DNA sample of the organism. This can be made possible by the use of Artificial Neural Networks (ANNs), specifically Convolutional Neural Networks (CNNs) that possess the ability to perform image classification. A CNN, namely the classifier CNN, can be trained on dataset containing NMR spectrum of each nitrogenous base. For this, the DNA sample has to be put through NMR spectroscopy. The different chromosomes can be isolated using separation techniques followed by separate sequencing of each chromosome. To demarcate the start position of the chromosome, either a radio-labeled phosphate group can be associated with the 5' end of

Rishabh Garg was with Birla Institute of Technology & Science - Pilani, KK Birla Goa Campus 403726 India (corresponding author, phone: 91-916-526-9669; e-mail: rishabhgargdps@gmail.com).

Anuja Vyas and Aamna Khan are with Institute for Excellence in Higher Education, Bhopal, M.P. 462016 India (e-mail: anujavyas20@gmail.com, jhinu911@gmail.com).

Muhammad Azwan Tariq is with ULC Mechanical Engineering, University College, London UK (e-mail: tariqazwan@gmail.com).

sequence or a primer specific to the first few nucleotides of the DNA sequence can be de-signed and bound to it. In both cases, a distinction in spectrum would be observed, due to difference in magnetic properties, from the rest of the strand which would mark the beginning of DNA strand. After obtaining the sequence of each chromosome, algorithms like CYANA can be applied to derive the molecular structure of this DNA sequence. The classifier CNN can be trained on data containing molecular structures of the four nitrogenous bases- Adenine, Guanine, Thymine and Cytosine. Wherever the CNN detects the presence of these molecules in the molecular structure sequence, it would label the molecules as A, G, T or C respectively after having compared the sample and training data. Thus, the model should be able to classify molecular structures of Adenine, Guanine, Thymine and Cytosine as ‘A’, ‘G’, ‘T’ and ‘C’ respectively upon receiving the molecular structures derived from NMR spectra of these molecules as inputs in a serial manner, ultimately generating the gene sequence.

The process can be implemented as follows:

```
  import numpy as np
  import tensorflow as tf
  from tensorflow.keras import layers, models

  # Step 1: Prepare dataset (NMR spectra of nitrogenous bases)
  # Prepare your dataset with NMR spectra and corresponding
  labels (A, G, T, C)

  # Step 2: Design and train CNN classifier
  def build_cnn_model(input_shape, num_classes):
     model = models.Sequential([
         layers.Conv1D(32, 3, activation='relu',
         input_shape=input_shape),
         layers.MaxPooling1D(2),
        layers.Conv1D(64, 3, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Conv1D(128, 3, activation='relu'),
        layers.MaxPooling1D(2),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

# Assuming data X_train and corresponding labels y_train are
being trained
# input_shape = shape of your input NMR spectra data
# num_classes = number of classes (4 for A, G, T, C)
input_shape = (num_features, num_timestamps)  # Update with
actual shape
num_classes = 4  # A, G, T, C
model = build_cnn_model(input_shape, num_classes)

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(X_train,    y_train,    epochs=10,    batch_size=32,
validation_split=0.2)

# Step 3: Classify DNA samples and generate gene sequence
def classify_dna_sample(model, dna_sample):
    # Assuming dna_sample is the NMR spectra of the DNA sample
    # Reshape the input according to the model input shape
    dna_sample = np.reshape(dna_sample, input_shape)
    # Make prediction
    prediction = model.predict(dna_sample)
    # Get the class with highest probability
    predicted_class = np.argmax(prediction)
    # Map predicted class to corresponding nucleotide
    nucleotides = ['A', 'G', 'T', 'C']
    gene_sequence = nucleotides[predicted_class]
    return gene_sequence

# Example usage:
# dna_sample_1 = ...  # Load NMR spectra of DNA sample 1
# gene_sequence_1 = classify_dna_sample(model, dna_sample_1)
# Repeat above for each DNA sample to generate the complete
gene sequence
```

C. Gene Editing

Though gene editing can be performed by following Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR); CRISPR-Associated Protein 9 (Cas9); Transcription Activator-Like Effector Nucleases (TALEN); or Zinc-Finger Nucleases (ZFNs), yet gene editing appears much more feasible if it is aided by AI. A neural network that can compare two given sequences - one that the H-Bot would be based-off of (target sequence) and the other that would carry the gene required to be incorporated into the H-Bot (desired sequence) - would be required in order to achieve this out-come. SNNs are neural networks operating on different sub networks that take an input each and compare them [5]. For the purpose of sequence comparison, neural networks based on Transformer Architecture specialized in the task of semantic similarity and difference of Genetic sequences can be designed and used. Each TASAG would take input sequences, for instance, Sequence ‘A’ which is the sequence that needs to be edited and incorporated into the H-Bot (target sequence) and Sequence ‘B’ which is the sequence carrying the desired gene which needs to be introduced in Sequence ‘A’ (desired sequence). Chromosome files which carry the desired genes can be selected and sent as input sequences for comparison to each TASAG. The TASAG compares these sequences and draws out differences between them. Since transformer architecture works on attention mechanism, it can be made to focus on functional sequences by weighing importance of protein coding regions over that of non-protein coding regions. These neural networks can be pre-trained on labeled data containing functional codons and their expression proteins in order to recognize which region of the sequence is responsible for that desired expression. Gene annotation can be used to facilitate these functions.

Once a particular difference in the desired sequence from the H-Bot sequence has been fetched out by TASAG, gene editing can be performed on-screen to induce necessary mutation(s) and introduce the desired gene(s) in that particular location of chromosomal file of H-Bot.

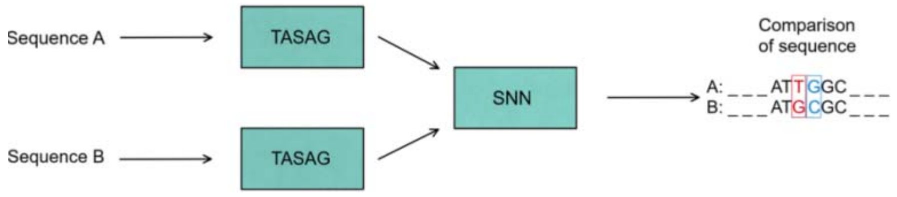

Fig. 1 Framework of TASAG

To implement the procedure described so far, one would typically need a combination of tools and libraries such as Python, TensorFlow or PyTorch for the neural network implementation (using the Transformer architecture), and possibly libraries like Biopython for handling genetic sequences. Here is a simplified version of code through which one could approach this implementation:

```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
from Bio import SeqIO
from Bio.Seq import Seq

# Define the Transformer-based neural network model
class TASAG(nn.Module):
    def __init__(self, input_size, output_size):
        super(TASAG, self).__init__()
        # Define layers and parameters of Transformer
architecture
        # Typically define transformer layers, attention
mechanisms, etc.

    def forward(self, x):
        # Implement forward pass of the model
        # Apply attention mechanisms, transformer layers, etc.
        return x

# Load and preprocess sequences
def load_sequences(file_path):
    sequences = []
    with open(file_path, "r") as file:
        for record in SeqIO.parse(file, "fasta"):
            sequences.append(str(record.seq))
    return sequences

def preprocess_sequences(sequences):
    # Preprocess sequences as needed (e.g., tokenization,
padding)
    # May convert sequences into numerical representations
    return processed_sequences

# Train the TASAG model
def train_model(model, train_data, train_labels, epochs=10,
batch_size=32):
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    for epoch in range(epochs):
        running_loss = 0.0
        for i in range(0, len(train_data), batch_size):
            inputs = train_data[i:i+batch_size]
            labels = train_labels[i:i+batch_size]

            optimizer.zero_grad()

            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

        print(f"Epoch {epoch+1}, Loss: {running_loss}")

# Perform sequence comparison and gene editing
def     sequence_comparison_and_gene_editing(target_sequence,
desired_sequences):
    model = TASAG(input_size, output_size)  # Initialize TASAG
model
    # Train the model using pre-labelled data if available

    for desired_sequence in desired_sequences:
        # Use the trained model to compare sequences and
identify differences
        # Perform gene editing based on identified differences
        # For demonstration, let's assume a simple edit of
replacing a substring
        edited_sequence = target_sequence.replace("old_gene",
"new_gene")

        # Save or return edited sequence
        print("Edited sequence:", edited_sequence)

# Example usage
if __name__ == "__main__":
    # Load sequences
    H_Bot_sequence = "ATCGATCGATCG..."
    desired_sequences                                         =
load_sequences("desired_sequences.fasta")

    # Preprocess sequences
    processed_target_sequence                                 =
preprocess_sequences([target_sequence])
    processed_desired_sequences                               =
preprocess_sequences(desired_sequences) //if ‘[]’ means that
target sequence is a sequence and thus stored in a list, it
should be ([desired sequences]) because both are lists.

    # Compare sequences and perform gene editing
sequence_comparison_and_gene_editing(processed_target_sequ
ence, processed_desired_sequences)
```

The given code provides a basic framework for implementing sequence comparison and gene editing using a

Transformer-based neural network (TASAG). However, the details of the TASAG model architecture, as well as any additional preprocessing steps specific to the dataset and requirements, have to be furnished separately.

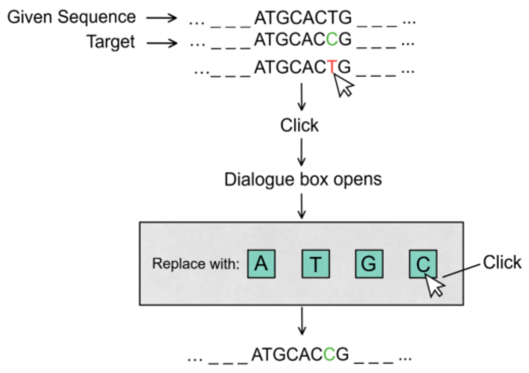
Given Sequence → ... _ _ _ ATGCACTG _ _ _ ...
Target → ... _ _ _ ATGCACCG _ _ _ ...
... _ _ _ ATGCACTG _ _ _ ...
Click
Dialogue box opens
Replace with: A T G C
Click
... _ _ _ ATGCACCG _ _ _ ...

Fig. 2 On-screen gene editing

D.Chemical Synthesis of Desired DNA Strand

Once the desired sequence has been obtained, DNA synthesis needs to be performed in order to obtain the H-Bot in physical form. For this purpose, a hypothetical 'Computerized DNA Synthesizer (CDS)' can be used. Under controlled temperature and pH, set concentrations of deoxyribonucleoside triphosphates (dNTPs) are pipetted out to extend a DNA stand as per the desired sequence.

A closed control loop system can be set in place in order to implement this. The components include four vials that store dNTPs namely deoxyadenosine triphosphate (dATP), deoxyguanosine triphosphate (dGTP), deoxythymosine tri-phosphate (dTTP) and deoxycytosine triphosphate (dCTP) with a micropipette associated with each of them, a chamber with synthesis plate that would accommodate the DNA strand during synthesis with temperature and pH regulators as well as various necessary enzymes, thus mimicking natural conditions of the nucleus. To ensure that a precise concentration of dNTP is pipetted out (for example, 50 µL of each dNTP of 10 mM concentration as in PCR) or a desired range of temperature and pH is maintained, a negative feedback closed loop control system can be formulated. When the process of synthesis is flanked, the first nitrogenous base displayed on screen causes CDS micropipette, attached to robot arm, to pump out a set concentration of that dNTP. This process continues until all 23 separate strands of DNA are synthesized. The physical DNA strand thus synthesized can be alluded to as 'artificial single stranded DNA' or 'assDNA'.

In order to implement the process, a basic application in Python code utilizing control structures and simulated actions of the DNA synthesizer, is given hereunder:

```
import random
import time

class DNA_Synthesizer:
    def __init__(self):
        self.dNTP_concentration = {'dATP': 10, 'dGTP': 10,
'dTTP': 10, 'dCTP': 10}  # mM
        self.temperature = 37  # Celsius
        self.pH = 7
        self.sequence = ""
        self.artificial_DNA = []

    def set_sequence(self, sequence):
        self.sequence = sequence

    def synthesis(self):
        for base in self.sequence:
            self.pump_dNTP(base)
            self.regulate_temperature_and_pH()
            time.sleep(0.5)  # Simulating synthesis time
            self.extend_DNA_strand(base)
            print("Synthesizing", base, "...")
        print("DNA synthesis completed.")

    def pump_dNTP(self, base):
        concentration = self.dNTP_concentration['d' + base +
'TP']
        print("Pumping out", c oncentration, "µL of d" + base
+ "TP")

    def regulate_temperature_and_pH(self):
        print("Regulating temperature and pH...")

    def extend_DNA_strand(self, base):
        self.artificial_DNA.append(base)

if __name__ == "__main__":
    sequence = "ATCG" * 6  # Example sequence
    synthesizer = DNA_Synthesizer()
    synthesizer.set_sequence(sequence)
    synthesizer.synthesis()
    print("Synthesized                DNA               strand:",
''.join(synthesizer.artificial_DNA))
```

In order to match the DNA synthesis process, this code initializes a DNA synthesizer object, sets the desired DNA sequence, and then executes the synthesis process by iterating over each base in the sequence, to simulate the pumping of dNTPs, regulating the temperature and pH, and extending the DNA strand. The code can be modified and expanded upon to incorporate more complex functionalities if needed.

With a desired DNA sequence displaying on the computer screen, the next step is for the machine to recognize the sequence and draw specific volumes of dATP, dGTP, dTTP and dCTP to place on the synthesis plate in a specific order. To do this, the machine must correlate specific inputs to specific robot arms that hold micropipettes corresponding to different dNTPs. This is done by establishing links between inputs A, T, G and C and microcontrollers such as an Arduino Board. This code can tie distinct inputs to each individual actuator. For instance, if 'A' is the next base in the DNA sequence, then robot arm 'A' will actuate to draw a specific volume of dATP and then place it into the sequence.

One would need to connect the actuators and input pins to the appropriate pins on the Arduino board, and adjust any necessary delays or parameters based on the specific setup. Here is a simple example code that demonstrates how one can achieve this using Arduino:

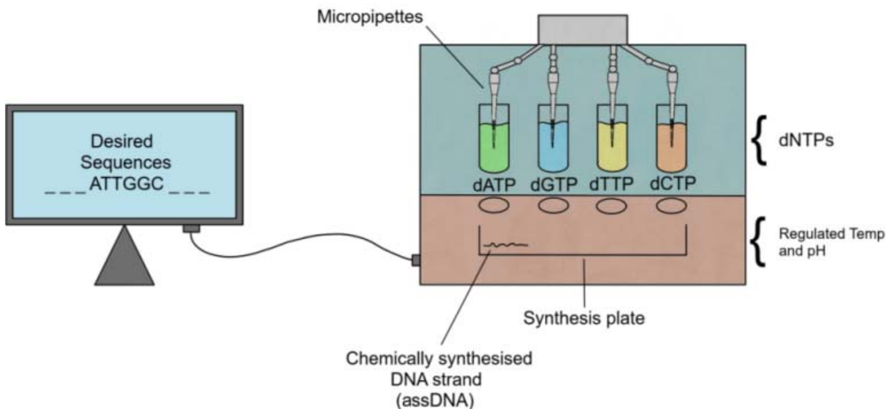

Fig. 3 Overview of CDS

```
// Define the pins connected to the actuators (robot arms)
const int armAPin = 2; // Arm for dATP
const int armTPin = 3; // Arm for dTTP
const int armGPin = 4; // Arm for dGTP
const int armCPin = 5; // Arm for dCTP

// Define the input pins connected to the DNA sequence display
const int inputAPin = 6; // Input for A
const int inputTPin = 7; // Input for T
const int inputGPin = 8; // Input for G
const int inputCPin = 9; // Input for C

void setup() {
  // Set the pins as outputs for the robot arms
  pinMode(armAPin, OUTPUT);
  pinMode(armTPin, OUTPUT);
  pinMode(armGPin, OUTPUT);
  pinMode(armCPin, OUTPUT);

  // Set the pins as inputs for the DNA sequence display
  pinMode(inputAPin, INPUT);
  pinMode(inputTPin, INPUT);
  pinMode(inputGPin, INPUT);
  pinMode(inputCPin, INPUT);
}

void loop() {
  // Read the input pins to determine the next base in the DNA
sequence
  int inputA = digitalRead(inputAPin);
  int inputT = digitalRead(inputTPin);
  int inputG = digitalRead(inputGPin);
  int inputC = digitalRead(inputCPin);

  // If the next base is A, actuate arm A to draw dATP
  if (inputA == HIGH) {
    actuateArm(armAPin);
  }
  // If the next base is T, actuate arm T to draw dTTP
  else if (inputT == HIGH) {
    actuateArm(armTPin);
  }
  // If the next base is G, actuate arm G to draw dGTP
  else if (inputG == HIGH) {
    actuateArm(armGPin);
  }
  // If the next base is C, actuate arm C to draw dCTP
  else if (inputC == HIGH) {
    actuateArm(armCPin);
  }
  // Add any necessary delay between iterations
  delay(1000);
}

// Function to actuate the specified arm
void actuateArm(int armPin) {
  digitalWrite(armPin, HIGH); // Actuate the arm
  delay(1000); // Wait for arm to move (adjust delay as needed)
  digitalWrite(armPin, LOW); // Deactivate the arm
}
```

This code sets up four robot arms (actuators) corresponding to dATP, dTTP, dGTP, dCTP, and correlates each input (A, T, G, C) to a specific arm. When an input corresponding to a particular base is detected, the corresponding arm is actuated to draw the appropriate nucleotide and place it into the sequence.

Closed Loop Control System for CDS

This system ensures that desired volume of A, T, C or G would be acquired for chemical synthesis of the DNA sequence. The user would input a desired volume, for instance, 50 µL of each base via the software that enables chemical synthesis for desired sequence(s). This volume signal would then pass through the machine’s controller, a microcontroller. Taking the desired volume input into account, the controller will output a control signal to the actuator, which can be a high accuracy syringe pump, in order to draw a specific volume of the desired base. Once a specific amount of dNTP is drawn, a volume transducer/sensor will measure output of the actual volume drawn and send a feedback signal to the input summing junction. At this point, the desired input volume is compared to the actual output volume and if any discrepancy between the two is detected, an error signal is generated. This signal will provide feedback into the controller and prompt it to activate the actuators again to readjust output to the desired volume. This system is quite dependable and its principles can be applied to both temperature and pH control which would make use of different sensors. Also, the time to reach a steady state output can be altered via the utilization of a PID controller which makes use of proportional, integral and/or derivative constants to speed the time response in acquiring a specific desired output.

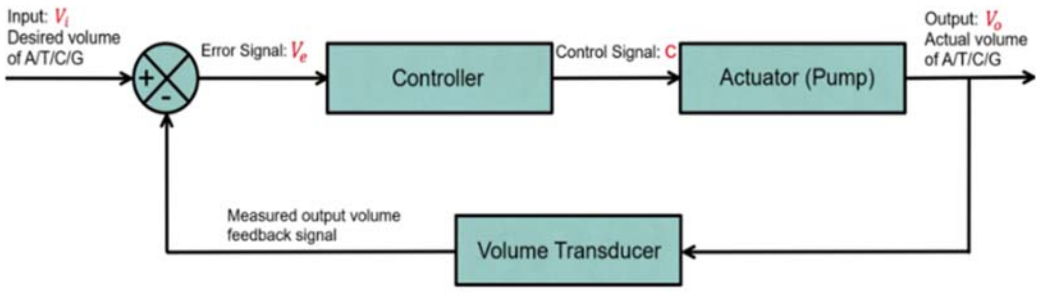

Fig. 4 Closed Control Loop System for CDS

Here is a simplified Python code to implement the process:

```
import time

class DNA_Synthesis_Controller:
    def __init__(self, base):
        self.base = base
        self.desired_volume = 0
        self.actual_volume = 0
        self.error_threshold = 0.1  # Adjust as needed
        self.actuator = SyringePump()   # Initialize syringe
pump
        self.volume_sensor  =  VolumeSensor()   #  Initialize
volume sensor

    def set_desired_volume(self, volume):
        self.desired_volume = volume

    def compare_volumes(self):
        return abs(self.desired_volume - self.actual_volume)
> self.error_threshold

    def adjust_output(self):
        while self.compare_volumes():
            if self.desired_volume > self.actual_volume:
                self.actuator.draw_volume(self.base,
self.desired_volume - self.actual_volume)
            elif self.desired_volume < self.actual_volume:
                self.actuator.retract_volume(self.base,
self.actual_volume - self.desired_volume)
            self.actual_volume                              =
self.volume_sensor.measure_volume(self.base)

class SyringePump:
    def draw_volume(self, base, volume):
        print(f"Drawing {volume} µL of {base}")
        time.sleep(1)  # Simulate drawing volume

    def retract_volume(self, base, volume):
        print(f"Retracting {volume} µL of {base}")
        time.sleep(1)  # Simulate retracting volume

class VolumeSensor:
    def measure_volume(self, base):
        volume = 50  # Placeholder for measured volume
        print(f"Measured {volume} µL of {base}")
        return volume

# Example usage
controller = DNA_Synthesis_Controller("A")
controller.set_desired_volume(50)
controller.adjust_output()
```

This code simulates the process of drawing a desired volume of a base (A, T, C, or G) using a syringe pump controlled by a controller. The controller adjusts the output volume based on feedback from a volume sensor until the desired volume is reached.

Robot Arm of CDS

Within the machine, once the high accuracy pump draws in specific amounts of a base, a robot arm with multiple joints will maneuver the pipette to the exact location of the synthesized DNA strand in order to add on to the sequence. In order for the arm to recognize the precise location it needs to go to, it will require a kinematic model and numerous sensors within each joint that measure and sense parameters, such as angles and angular accelerations of the joint.

To formulate a kinematic model, a world reference frame would be placed on the rigid connection point at the base of the robot arm. This is the frame from which positions of the end effector will be determined. For instance, if we have a fixed 3-D XYZ reference frame, we can then determine where the end effector is along the X, Y and Z axes. As an example, the tip of the pipette could be 0.05m along in the X axis, 0.02 m along the Y axis and 0.01 m along the Z axis, providing a coordinate vector of $\begin{bmatrix} 0.05 \\ 0.02 \\ 0.01 \end{bmatrix}$. At each subsequent joint a new reference frame would be placed until the end effector, in our case the tip of the pipette, is reached. Using these frames, Denavit-Hartenberg (DH) parameters will be formed, which define the geometry of how the center of each joint is related to the center of its parent joint, for example, it includes angles and distances between the axes of reference frame 1 to reference frame 2 in order to derive the forward kinematic relationships.

Once DH parameters are characterized, the location of the pipette can be determined via forward kinematics, which effectively links each joint's reference frame to the next via homogenous transformation matrices that incorporate the angles between the frames and the distance between their origins. The standard form of such a matrix is shown in (1):

$$H = \begin{bmatrix} R & d \\ 0_{1x3} & 1 \end{bmatrix} \tag{1}$$

where R = rotation matrix, d = displacement from origin of frame 1 to frame 2 in the X, Y and Z directions. As an illustration, using Fig. 5, the transformation matrix between the world frame and Joint 1 frame would be:

$$H = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_2\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using forward kinematics, we can find the relationships needed for inverse kinematics. Inverse kinematics is the process of determining the joint angles required to place the end effector (pipette) at a desired position. The Jacobian Inverse Method can be used for this, which involves an algorithm that iteratively guesses and adjusts the joint angles until the end effector reaches the desired position.

To summarize, if the pipette needs to be at location A, the robot arm joints can form angles θ1 to θn (n being the number of joints) with respect to each joint reference frame to orient the robot arm to the desired position. With sensors such as rotary encoders, which are used to measure the angular position of a joint, or accelerometers and gyroscopes that provide additional information on joint orientation and movement, a closed loop negative feedback system can be used to output desired robot joint parameters with high accuracy sensors to ensure dNTP is placed exactly where it needs to be and with fast responses.

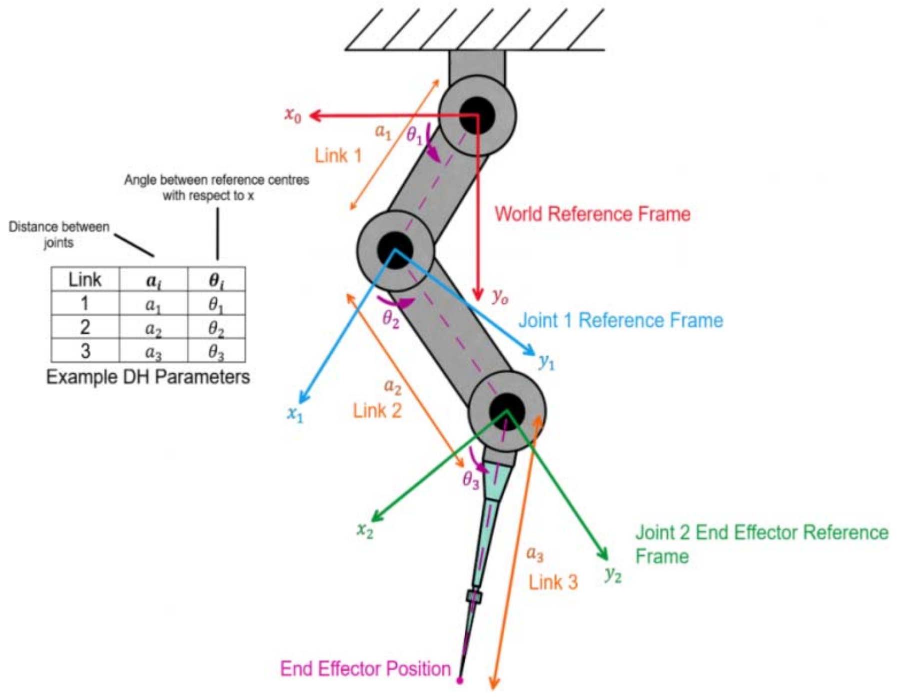

Fig. 5 Robot arm of CDS

Here's a high-level execution of the process:

//forward kinematics to be added before inverse kinematics, preferable to add code calculating transformation matrix, further changes as per theory may be added (if needed)

```
class RobotArm:
    def __init__(self, num_joints):
        self.num_joints = num_joints
        self.joint_angles = [0] * num_joints
        self.joint_velocities = [0] * num_joints
        self.joint_accelerations = [0] * num_joints
        self.sensor_data = {}

    def update_sensor_data(self, sensor_data):
        self.sensor_data = sensor_data

    def calculate_inverse_kinematics(self, target_position):
        # Use sensor data and DH parameters to calculate joint
angles
        # This function would implement the inverse kinematics
model
        # Placeholder for demonstration purposes
        joint_angles = [0.5, -0.3, 0.8]  # Example joint angles
        return joint_angles
    def move_to_position(self, target_position):
        self.joint_angles                                      =
self.calculate_inverse_kinematics(target_position)

    def feedback_control(self, target_angles):
        # Implement a closed-loop feedback control system
        # This could involve using sensor data to adjust joint
angles to reach target angles
        # Placeholder for demonstration purposes
        self.joint_angles = target_angles

    def add_dNTP(self, location):
        #  Perform  the  motion  to  add  dNTP  to  the  desired
location
        self.move_to_position(location)
        # Placeholder for adding dNTP to the DNA strand
        print("dNTP added at location:", location)


# Example usage
if __name__ == "__main__":
    num_joints = 3  # Example number of joints
    robot_arm = RobotArm(num_joints)

    # Example sensor data (could be rotary encoder readings,
accelerometer data, etc.)
    sensor_data = {
        "joint1_angle": 0.5,
        "joint2_angle": -0.3,
        "joint3_angle": 0.8,
        # Additional sensor data as needed
    }
    robot_arm.update_sensor_data(sensor_data)

    # Example target position
    target_position = {
        "x": 0.02,
        "y": 0.05,
        "z": 0.01,
    }

    # Move the robot arm to the target position
    robot_arm.move_to_position(target_position)

    # Add dNTP at the target location
    robot_arm.add_dNTP(target_position)
```

While providing a basic structure for a robot arm with multiple joints, this implementation includes functions for updating sensor data, calculating inverse kinematics, moving to a specified position, and adding dNTP at a specified location.

Complementary Strand Synthesis

The natural state of DNA in the nucleus is double stranded and helical. Both strands of DNA are complementary to one another and are called template and coding strands. The single strand that has been used for artificial synthesis by CDS is the coding strand as it carries majority of genetic information. To synthesize the complementary template strand, wherever 'A' is present in the coding strand, 'T' needs to be attached on template and wherever 'G' is present, 'C' needs to be attached and vice versa. For this task, DNA polymerase, primers and dNTPs can be directly used in a pH and temperature-controlled environment, similar to Polymerase Chain Reaction (PCR) [6]. The only difference here is that instead of denaturation, renaturation needs to be performed. For renaturation, gradual lowering of temperature is a prerequisite [7].

E. Insertion of Target Strand into an Ovum for Development

In order to grow the adsDNA obtained into a functional H-Bot, it requires a cell and cytoplasmic factors. Both these cellular parts can be provided by an ovum or female sex gamete. First, the ovum cell is enucleated, either naturally or artificially, in order to isolate its nucleus. If a natural ovum is used, all the genetic material present in the nucleus can be degraded and adsDNA can be inserted via microinjection. If an artificial ovum is used, it can be equipped with a hollow nucleus to incorporate adsDNA. Once the synthesized adsDNA strands are inserted into this nucleus, the diploid egg can be stimulated to cleave and develop under artificial conditions. The development and maturation of zygote take place within ovum cytoplasm as all factors necessary to support its growth are present in it. The initial stage of ovum containing adsDNA can be referred to as 'Zybot' and once multicellular development stage is reached, it can be called 'Embot'. This has been conceptualized while carrying the assumption that the nucleus of human ovum is comparatively larger in size and can accommodate female as well as male genetic material at the time of fertilization. Hence, the nucleus would be able to withstand insertion and contain this artificial genome.

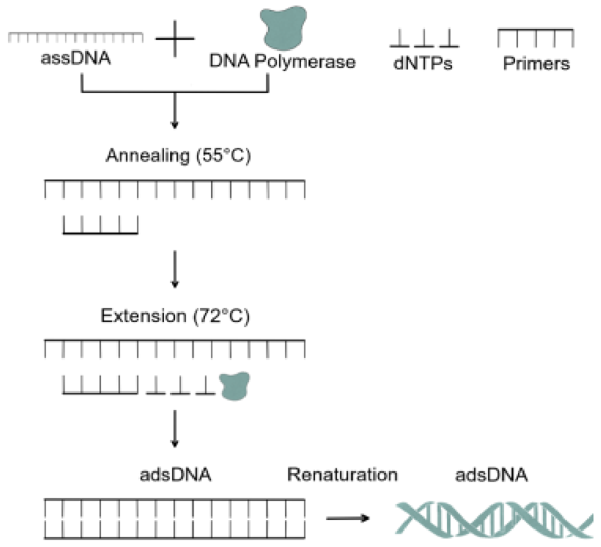

Fig. 6 Complementary strand synthesis

The Zybot, so created, can thereafter be placed under suitable developmental conditions *in vitro* for complete ectogenesis [8] to obtain a developed H-Bot baby. Here's a Python code to create instance of ovum, enucleate it, insert ad-sDNA, add cytoplasmic factors, stimulate cleavage, and develop the Zybot:

```
class Ovum:
    def __init__(self, artificial=False):
        self.artificial = artificial
        self.nucleus = None
        self.cytoplasmic_factors = []

    def enucleate(self):
        self.nucleus = None

    def insert_adsDNA(self, adsDNA):
        if self.artificial:
            self.nucleus = "Hollow nucleus with adsDNA"
        else:
            self.nucleus = adsDNA

    def add_cytoplasmic_factors(self, factors):
        self.cytoplasmic_factors.extend(factors)

    def stimulate_cleavage(self):
        if     self.nucleus     is     not     None     and
self.cytoplasmic_factors:
            print("Stimulating cleavage...")
            # Cleavage process

            print("Cleavage stimulated successfully.")
        else:
            print("Failed to stimulate cleavage. Nucleus and/or cytoplasmic factors missing.")

    def develop_zybot(self):
        if self.nucleus is not None and self.cytoplasmic_factors:
            print("Developing Zybot...")
            # Zybot development process
            print("Zybot developed successfully.")
        else:
            print("Failed to develop Zybot. Nucleus and/or cytoplasmic factors missing.")

# Example usage:
# Create an artificial ovum
artificial_ovum = Ovum(artificial=True)

# Enucleate the ovum
artificial_ovum.enucleate()

# Insert adsDNA into the ovum
adsDNA_sequence = "ACGTACGT"
artificial_ovum.insert_adsDNA(adsDNA_sequence)

# Add cytoplasmic factors
cytoplasmic_factors_list = ["Factor1", "Factor2", "Factor3"]
artificial_ovum.add_cytoplasmic_factors(cytoplasmic_factors_list)

# Stimulate cleavage
artificial_ovum.stimulate_cleavage()

# Develop Zybot
artificial_ovum.develop_zybot()
```

III. Customizable Organisms

Gene editing and chemical synthesis of DNA are crucial processes that can be used to create H-Bots which have the ability to perform functions that are near to impossible for normal human beings. A few of these applications are:

A. Infrasonic Communication

Infrasonic communication involves producing and perceiving sound waves with frequencies below 20 Hz. While humans cannot naturally generate or sense infrasonic sounds due to our hearing range of 20 to 200 Hz, they can be valuable for detecting distant noises, traveling hundreds of miles [9].

Infrasonic voice production can be made possible with the help of longer vocal cords. A good example of this is the American singer and composer - Tim Storms who can produce notes with frequency as low as 0.189 Hz. His exceptional ability stems from his vocal cords that are twice the length of an average human with increased muscular flexibility [10]. Genes responsible for this mutation can be worked out and induced in the genome of H-Bot at the step of gene editing.

Infrasound is produced by animals such as elephants, whales, rhinos, pigeons, etc. Out of these, pigeons can hear the lowest frequency of infrasound i.e., around 0.05 Hz in a sound isolation chamber [11]. While the precise workings of the pigeon ear remain incompletely understood, it is conjectured that several factors, including the ability to filter environmental infrasound, the low stiffness of the middle and inner ear in pigeons facilitating efficient transmission, and the presence of infrasonic-sensitive neurons in the basilar papilla, contribute to infrasound perception in pigeons [12]. Proteins and thus genes responsible for such modifications can be worked out in the pigeon genome and used to re-place those in human genome by the process of gene editing.

B. Enhanced Olfaction

Organisms possess olfaction, enabling them to perceive smells. Dogs, in particular, exhibit an exceptional olfactory sense, surpassing that of humans by a significant margin. They have around five folds more olfactory receptors in their noses than humans and a fold of tissue in their nose that separates the function of inhalation and olfaction [13]. They can detect bombs, drugs, cancer, etc.

Gene clusters governing olfaction in both humans and dogs are orthologous [14]. However, dogs possess certain features due to which their sense of smell is significantly more powerful than humans. These are: higher number of olfactory receptors, more surface area of olfactory epithelium incorporating more neurons, larger brain region responsible for perception of smell, etc. In dogs, there are estimated to be around 800 to 1,000 different olfactory receptor genes, making up a significant portion of their genome [15]. These genes are highly diverse and can detect a wide range of odors, allowing dogs to have an incredibly acute sense of smell compared to humans [16]. These genes can be recognized and thus introduced in H-Bot genome.

C. Adhesive Capability

Geckos are reptiles that possess unique specialized appendages allowing them to cling to any surface via adhesive foot pads. These foot pads consist of millions of hair-like tiny setae and each possess several spatulae. These reptiles contain β-keratin genes that have been associated with gecko's adhesive ability. It has been found that these β-keratin genes have undergone major expansion via duplication. These gene expansions can be further studied and either introduced or triggered within H-Bot genome. Since geckos have existed around millions of years [17], it is very likely that such genes are present as non-protein coding regions within the human genome. Moreover, for the H-Bot to have an efficient adhesive climbing ability, around 80% of its anterior body surface needs to be covered with setae. Since an adult human has a lower body surface, the H-Bot can be made infant-sized (< 10 kg) [18]. The calculations have been shown below.

Preferable regions to be covered by setae: Anterior torso, each arm and each leg, with surface areas 16%, 8% each and 16% each.

%Total body surface area (TBSA) = %SA of Anterior torso + %SA of arms + %SA of legs = 16% + 16% + 32% = 64%

To further increase surface area, one can decrease mass or incorporate additional accessory appendages such as patagium or extra integumentary folds.

```
class H_Bot:
    def __init__(self, weight):
        self.weight = weight

    def calculate_surface_area(self):
        # Assuming the infant-sized H-Bot
        torso_surface_area = 0.16  # 16% of total body surface
area
        arm_surface_area = 0.08  # 8% each
        leg_surface_area = 0.16  # 16% each

        total_surface_area   =   torso_surface_area   +   2   *
arm_surface_area + 2 * leg_surface_area
        return total_surface_area

    def                  calculate_required_setae_area(self,
total_surface_area):
        required_setae_area = 0.8 * total_surface_area  # 80%
coverage
        return required_setae_area

    def    adjust_surface_area(self,    total_surface_area,
required_setae_area):
        #  Adjust  surface  area  by  decreasing  mass  or
incorporating additional appendages
        # For simplicity, let's assume no mass reduction or
additional appendages
        return total_surface_area

    def implement_setae(self, total_surface_area):
        # Implement setae coverage efficiently
        setae_area_covered  =  total_surface_area    #  For
demonstration, assuming full coverage
        return setae_area_covered


# Main function
def main():
    h_bot_weight = 10  # Assuming weight < 10kg for infant-
sized H-Bot
    h_bot = H_Bot(h_bot_weight)

    total_surface_area = h_bot.calculate_surface_area()
    required_setae_area                                    =
h_bot.calculate_required_setae_area(total_surface_area)
    total_surface_area                                     =
h_bot.adjust_surface_area(total_surface_area,
required_setae_area)
    setae_area_covered                                     =
h_bot.implement_setae(total_surface_area)

    print("Total surface area of H-Bot:", total_surface_area)
    print("Required   setae   area   for   80%   coverage:",
required_setae_area)
    print("Actual setae area covered:", setae_area_covered)


if __name__ == "__main__":
    main()
```

This code calculates the total body surface area of the H-Bot, determines the required setae area for 80% coverage, adjusts the surface area if needed, and implements the setae coverage.

D. Visibility Enhancement

Eagles are known to have one of the best visions in the animal kingdom. They have a 20/5 vision as compared to 20/20 human vision. They also have features like the presence of 2 foveae in each eye with over a million cone cells in their central fovea. They also have the ability to 'zoom in' their prey as they have lenses that can change shape to focus on an object with accuracy [19]. Moreover, the universal master control gene for eye morphogenesis, the Pax 6 gene, is conserved for all organisms [20]. Thus, the differences between human and eagle eye genes can be worked out, and the same may be incorporated into the H-Bot genome for better vision. This would prove more effective if genes for associated morphological and physiological changes such as larger eye size and the reflective layer behind the retina could also be identified and incorporated. In addition, genes related to UV light-sensitive 'opsin' can also be introduced into the H-Bot genome through gene editing to capture and sense UV light. Furthermore, if certain mutations can be induced in genes related to vision, such as induction of new photoreceptors sensitive to radio waves, the ability to sense these radio wave signals in the brain could be induced and mechanisms that allow them to pass through artificial filters. By filtering out atmospheric refraction or designing telescopic retinas, it may be possible for the H-Bot to see galaxies that are billions of light years away from Earth's surface.

E. Flyers Overcoming Barriers of Flight and Visibility

Birds can be designed to fly without energy and visibility constraints. These changes can be brought about to create a 'watchdog'. The former could be made possible by increasing the efficiency of respiration since sustained flight would require higher energy. This can be achieved by making necessary mutations in the genes of birds responsible for high lung efficiency. These genes can be mutated to create larger lungs to increase oxygen intake. Furthermore, their flight muscles have to be well developed so that they do not succumb to frequent fatigue. For this, the proteins involved in muscle contraction – actin, myosin etc. will have to be increased. This can be done by performing corresponding mutations. These factors can help the bird-bot to fly for miles and record essential data which can be accessed through sensors in the brain.

F. Organisms from de Novo Proteins

Plastic waste and oil spills are a major threat to the environment. It is the need of the hour to design proteins that are capable of breaking down these wastes and thus cleaning the ecosystem. Amino acid sequences for proteins capable of performing the said functions can be designed and reverse translation can be performed to obtain the respective mRNA strand. Furthermore, reverse transcription can then ultimately produce the DNA strand which can be used to synthesize organisms capable of producing proteins under suitable conditions to perform such functions. These proteins can be designed to be crucial in the metabolism of these 'bio-sweepers'.

IV. Discussion

The fusion of Artificial Intelligence with genetics has the potential to transmute the lives of living beings. This integration enables them to create or even replace their own selves. The present study opens vistas to the creation of H-bots - AL forms - a subject currently sparking fervent discussion and debate. This stands in contrast to earlier researches focused on humanoid robots, which are programmable machines that resembles human in its build or functionality [21].

The current concept of creating H-bots stems from human curiosity that may be traced back to Garg (2021-24) [1]. 'Human bots' (H-bots) are biologically engineered entities designed to closely resemble and function as humans. This curiosity-driven pursuit extends to the creation of other life forms: 'Zoobots' (Z-bots) for animals, A-bots for aves or birds, B-bots for plants, and M-bots for microorganisms. Despite lacking inherent life, these entities are seen to be perfect mimics of live matters, full of potential [2].

The current study builds on prior research [3] by expanding upon the idea and offering a detailed exploration of its practical application. The present study explores it more deeply and attempts to provide a more thorough framework involving, extracting the sequence from the sample, aligning the sequence on the system, comparing it to the target sequence, modifying the target sequence, inserting it into the ovum, and eventually its development.

Studies [1]-[3], and the present research offer promising new concepts that could represent significant breakthroughs in the field of biotechnology. Nevertheless, it is essential to consider both the potential benefits and drawbacks of such an advancement.

A. Merits

(i) Abandon animal testing: This study aims to eradicate animal testing across various research domains, including vaccine development, cosmetics, and medications. By doing so, we can halt the unethical practice of subjecting defenseless animals to experimentation while upholding animal welfare standards.

(ii) Tailored characteristics for specific tasks: The upcoming H-bot will boast enhanced functionalities tailored for specific tasks, potentially reducing the reliance on both human labor and machines lacking human-like abilities. For instance, in modeling scenarios, H-bots can be designed with desirable physical attributes and a congenial demeanor.

(iii) Pharmaceutical and vaccine testing surrogate: Current trials often involve animals at various stages, leading to inefficiencies and adverse outcomes due to species differences. To circumvent these challenges, H-bots can be subjected to experimentations and trials, and can yield accurate and comprehensive results. Furthermore, tailored experiments can ascertain optimal dosages and effects of medications, considering variables like height, weight, gender, and age among others.

(iv) Organ and biological sample collection: Utilizing H-bots that meet necessary standards and patient compatibility criteria, one can engineer H-bots to isolate organs suitable for transplantation. Their versatility extends to collecting blood and its components, liver segments, extremities, and more.

(v) Research Advantages and Associated Studies: Human subjects pose limitations due to social constraints and ethical considerations. In contrast, non-personal entities like H-bots offer opportunities for expansive and transformative research without such barriers.

B. Demerits

(i) Social and ethical issues: The creation of such beings challenges the very concept of 'ethics.' The social ramifications of H-bots, including their acceptance in society, code of behavior, accountability, and human-H-bot contact, are a cause for serious concern.

(ii) Threat to human expertise: H-bots, adept at specific tasks, pose a potential threat to human expertise. If left unchecked, this trend could lead to severe consequences, overshadowing human involvement.

(iii) Loss of human dignity: Despite being artificial, H-bots bear striking resemblance to humans, potentially compromising human dignity. Interactions between humans and H-bots could disrupt regular social dynamics, impacting interpersonal relationships significantly.

(iv) Accountability and responsibility: The absence of ownership over H-bots presents a clear accountability challenge. Failure to address this issue could escalate into a significant problem. If these entities display inappropriate behavior or engage in unforeseen activities, prompt action is crucial. To address these issues, it is essential to set up a monitoring organization, keep updated records of their activities and locations, and ensure accountability.

(v) Risk of new species development: It is widely acknowledged that advanced technology could yield H-bots mirroring humans, albeit with intentionally designed traits. These H-bots may potentially reproduce among themselves and/or with humans, posing a significant threat to the natural equilibrium.

(vi) Creation and misuse of bizarre or extraordinary H-Bots: There is a possibility of introduction of unusual traits into H-Bots or gene mutations which may arise during the H-bot development process. Formation of such H-Bots can result in detrimental consequences which could be beyond the control of its developers.

Despite being entirely conceptual, this research serves as a critical foundation for understanding the potential of the idea and its implications. With thorough analysis, it clarifies the potential results and impacts of deploying such technology. It delineates both favorable aspects - such as effective testing and integration of desired traits - and adverse consequences - such as ethical dilemmas and inherent risks. This holistic strategy underscores the significance of ad-dressing challenges and constraints, while also accentuating potential benefits. In essence, despite its theoretical nature, this research significantly contributes to our comprehension and decision-making about the adoption and advancement of this technology.

V. Conclusion

With the advent of novel and powerful technologies like Neural Networks as well as advancements in the field of Genomics, it is necessary to integrate these domains in order to achieve new advanced technologies such as AL. It has been observed that changes in an organism's functioning require changes in morphological, anatomical and neural structures related to it, which ultimately require changes in its genome. There has been a growing prevalence and rapid evolution of gene sequencing technologies, which makes the process of gene editing via use of artificial intelligence possible to implement. Software that aids in machine designing has accelerated development in areas such as mechatronics, giving birth to advancements in robotics. Integration of Mechatronics with Genomics is another asset contributing to AL which can be used for physical synthesis of gene, the code of life. Further research is necessary in order to elevate the possibilities of AL or to derive valuable insights from related researches to produce innovations that may contribute to aiding humanity. However, it needs to be ensured that such technologies and innovations do not prove to be detrimental in the long run. Necessary ethical considerations and arguments need to be re-iterated and enunciated so as to prevent foreseeable adverse effects.

References

[1] Garg R. (2021). Decoding Tomorrow: Traversing the Landscape of Artificial Life. Scholars Park. Available at: https://scholarspark.wordpress.com/2021/04/21/example-post-3/
[2] Garg, R. (2024) From Virtual World to Real Lives -1: Sculpting New Realities with ML, AI, and IoT. Taylor & Francis, Routledge, Oxfordshire, UK. 1-400.
[3] Garg, H. K. and Vyas, A. (2024) Exploring Genome Projects and Beyond - Untwining the Past, Present & Future for Transformative Applications | Aiming to Decode You. Zenodo. doi: 10.5281/ zenodo.10466512.
[4] Definition and supported options (no date) GFF/GTF File Format. Available at: https://asia.ensembl.org/info/website/upload/gff.html.
[5] Benhur, S. (2022) A friendly introduction to Siamese networks, Built In. Available at: https://builtin.com/machine-learning/siamese-network.
[6] Polymerase chain reaction (PCR) (no date) National Center for Biotechnology Information. Available at: https://www.ncbi.nlm.nih.gov/probe/docs/techpcr/.
[7] Denaturation (melting curve) and Renaturation of DNA (no date) Denaturation (Melting Curve) and Renaturation of DNA ~. Available at: https://www.biotechfront.com/2021/04/denaturation-and-renaturation-of-dna.html.
[8] Bulletti FM, Sciorio R, Palagiano A, Bulletti C. The artificial uterus: on the way to ectogenesis. Zygote. 2023 Oct;31(5):457-467. doi: 10.1017/S0967199423000175. Epub 2023 Jun 26. PMID: 37357356.
[9] Infrasound Sensor Technology (no date) NASA. Available at: https://technology.nasa.gov/patent/LAR-TOPS-106.
[10] Reneau, A. (2021) Tim storms, the guy with the world's lowest voice, gives 'bass' a whole new meaning, Up-worthy. Available at: https://www.upworthy.com/tim-storms-lowest-voice-in-the-world.
[11] Kreithen, M.L., Quine, D.B. Infrasound detection by the homing pigeon: A behavioral audiogram. J. Comp. Physiol. 129, 1–4 (1979). https://doi.org/10.1007/BF00679906
[12] Zeyl JN, den Ouden O, Köppl C, Assink J, Christensen-Dalsgaard J, Patrick SC, Clusella-Trullas S. Infrasonic hearing in birds: a review of audiometry and hypothesized structure-function relationships. Biol Rev Camb Philos Soc. 2020 Aug;95(4):1036-1054. doi: 10.1111/brv.12596. Epub 2020 Mar 31. PMID: 32237036.
[13] How powerful is a dog's nose? (2020) Phoenix Veterinary Center - Veterinarian in Phoenix, AZ US. Available at: https://phoenixvetcenter.com/blog/214731-how-powerful-is-a-dogs-nose.
[14] Quignon P, Kirkness E, Cadieu E, Touleimat N, Guyon R, Renier C, Hitte C, André C, Fraser C, Galibert F. Compar-ison of the canine and human olfactory receptor gene repertoires. Genome Biol. 2003;4(12):R80. doi: 10.1186/gb-2003-4-12-r80. Epub 2003 Nov 28. PMID: 14659017; PMCID: PMC329419.
[15] Olender T., Fuchs T., Linhart C., Shamir R., Adams M., Kalush F., Khen M., and Lancet D., (2004) The canine ol-factory subgenome, Genomics, 83(3), pp. 361–372. doi: 10.1016/j.ygeno.2003.08.009.
[16] Kokocińska-Kusiak A., Woszczyło M., Zybala M., Maciocha J., Barłowska K., and Dzięcioł M., Canine Olfaction: Physiology, Behavior, and Possibilities for Practical Applications. Animals (Basel). 2021 Aug 21;11(8):2463. doi: 10.3390/ani11082463. PMID: 34438920; PMCID: PMC8388720.
[17] Liu, Y., Zhou, Q., Wang, Y. et al. Gekko japonicus genome reveals evolution of adhesive toe pads and tail regeneration. Nat Commun 6, 10033 (2015). https://doi.org/10.1038/ncomms10033.
[18] Correspondent, D. (2016) Why humans can't walk up walls like Spiderman decoded, Deccan Chronicle. Available at: https://www.deccanchronicle.com/pets-and-environment/190116/why-humans-can-t-walk-up-walls-like-spiderman-decoded.html.
[19] Neimark, J. (2019). What is eagle eye vision? All About Vision. Available at: https://www.allaboutvision.com/resources/eagle-vision/.
[20] Gehring WJ. The genetic control of eye development and its implications for the evolution of the various eye-types. Int J Dev Biol. 2002 Jan;46(1):65-73. PMID: 11902689.
[21] Hasan, A. (2023). Humanoid Robots-Recent Developments & Human-Robot Interaction: A paper review. 10.13140/RG.2.2.19016.80641.

Rishabh Garg | BITS Pilani, India is a Software Development Engineer (L3) with Google. He has worked in Data Science for Indian Institute of Technology, New Delhi, SDE with ServiceNow and Brand Partner with Cuvette. He is currently enrolled for Masters in Computer Science – On campus program with Georgia Institute of Technology, Atlanta, Georgia, United States.

He has authored books on Blockchain for Real World Applications (John Wiley & Sons Inc. US); From Virtual World to Real Lives (in two volumes - Taylor & Francis, Oxfordshire, UK); One World - One Identity, Self Sovereign Identities, and a number of books in foreign languages that were published in US, UK, Germany, France, Italy, Moldova, Russia, Spain, and Portugal.

He is a Journal Referee with IEEE Internet of Things. He has published 5 research papers, 12 chapters, 15 conference papers, and 42 blogs / online articles. He has been Program Committee Member cum Reviewer for more than 30 International Conference on Artificial Intelligence, Machine Learning, Cloud Computing, and Blockchain held in San Francisco (USA), Toronto, Vancouver (Canada), London (UK), Zurich (Switzerland), Copenhagen (Denmark), Youngs, Sydney. Melbourne, New South Wales (Australia), and Dubai (UAE).

Rishabh is a recipient of the National Award for Exceptional Achievements in Innovation from the President of India, National CSIR Innovation Award from the Prime Minister of India, and honors from various ministries with Government of India. He has also received an International Bronze Award from the Royal Commonwealth Society, London and Young Scientist Award from Ministry of Science & Technology, and Earth Sciences, Government of India for creative technological solutions.

Anuja Vyas is currently pursuing her Masters in Biotechnology from Institute for Excellence in Higher Education, Bhopal, India. In the midst of a riveting research project on artificial life, she coined the term 'Genetronics'. Anuja concocted an innovative application that impeccably detects variations within a genome compared to the standard one.

Aamna Khan is a student of Institute for Excellence in Higher Education, Bhopal, Madhya Pradesh, India. She is presently pursuing her Bachelor of Science (B.Sc.) in Biotechnology. Intrigued by Genomics, Bioinformatics, Healthcare, Sustainability, Artificial Intelligence, and Machine Learning, she has published a paper on feasibility and constraints in implementing Artificial Intelligence in Biotechnology and Healthcare.

Azwan Tariq is a student of University College London in the United Kingdom. He is currently pursuing his Masters in Mechanical Engineering (M. Eng.) with Business Finance. He is the control, robotics and simulation lead in his fourth year project, the construction of a bio-inspired underwater surveillance robot. His interests lie in Control implementation, Robotics, Dynamics, Design, Simulations and Sustainability.