Project Jenkins:
Monkey Thinks, Robot Moves... and Back?

Andrii Zahorodnii and Dima Yanovsky, MIT

Project Webpage: https://www.808robots.com/projects/jenkins

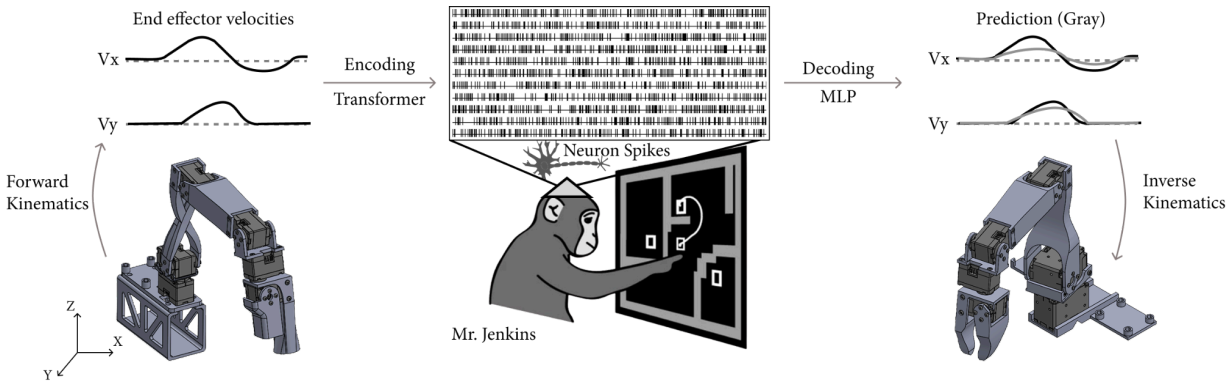

Figure 1. Project Jenkins. Leader arm velocities are computed via forward kinematics, then a transformer generates synthetic neural data. An MLP trained only on real monkey neural data decodes it back into velocity space, commanding the follower arm's movement. Monkey diagram adapted from Kaufman et al. (2014); robotic arm images from github.com/jess-moss/koch-v1-1.

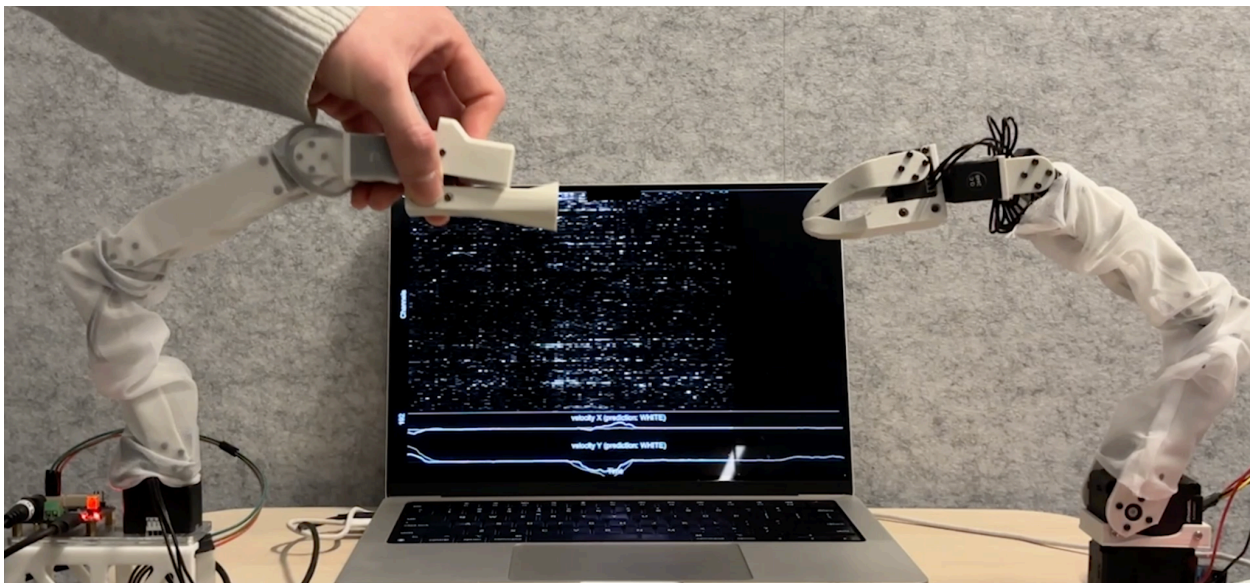

Figure 2. Our approach in action. For the full video, please see the project's webpage: https://www.808robots.com/projects/jenkins

Abstract

Project Jenkins explores how neural activity in the brain can be translated into robotic movement and, conversely, how movement patterns can be used to reconstruct brain activity. Using real neural data recorded from a macaque monkey named Jenkins, we develop models for decoding (converting brain signals into robotic arm movements) and encoding (simulating brain activity with a given movement).

For the interface between the brain simulation and the physical world, we utilized Koch v1.1 leader and follower robotic arms. We developed an interactive web console that allows users to generate synthetic brain data from joystick movements in real time.

Our results are a step towards brain-controlled robotics, prosthetics, and enhancing normal motor function. By accurately modeling brain activity, we take a step toward flexible brain-computer interfaces that generalize beyond predefined movements.

Obtaining Jenkins' Data

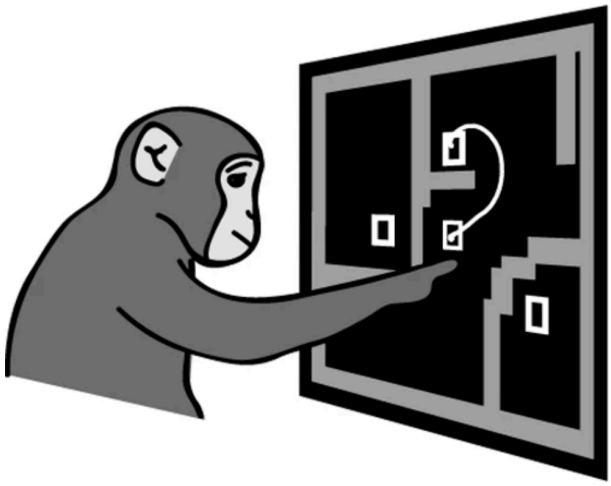

Figure 3. A typical monkey reaching task. Figure adapted from Kaufman et al. (2014).

The neural data used in our project came from the primary motor cortex (M1) and caudal portion of the dorsal premotor cortex (PMd) areas of the brain of a rhesus macaque monkey, Jenkins. The dataset was published by Mark Churchland and others in 2021 and is publically available here. Jenkins was trained on multiple reaching tasks, where the goal of the task is to press on dots that randomly light up on the screen in front of it. Every time the monkey completes a trial successfully, he is rewarded with fruit juice. In our project, we used data from a center-outreach task, where the monkey always starts out with its hand in the middle of the screen, and the dots light up in one of 8 positions (0°, 45°, 90°, 135°, 180°, 225°, 270°, or 310°).

All in all, there are more than a dozen hours of brain recordings together with the tracking of the monkey arm as it pressed dots on the screen thousands of times.

Decoding: from Neural Data to Robot Movement

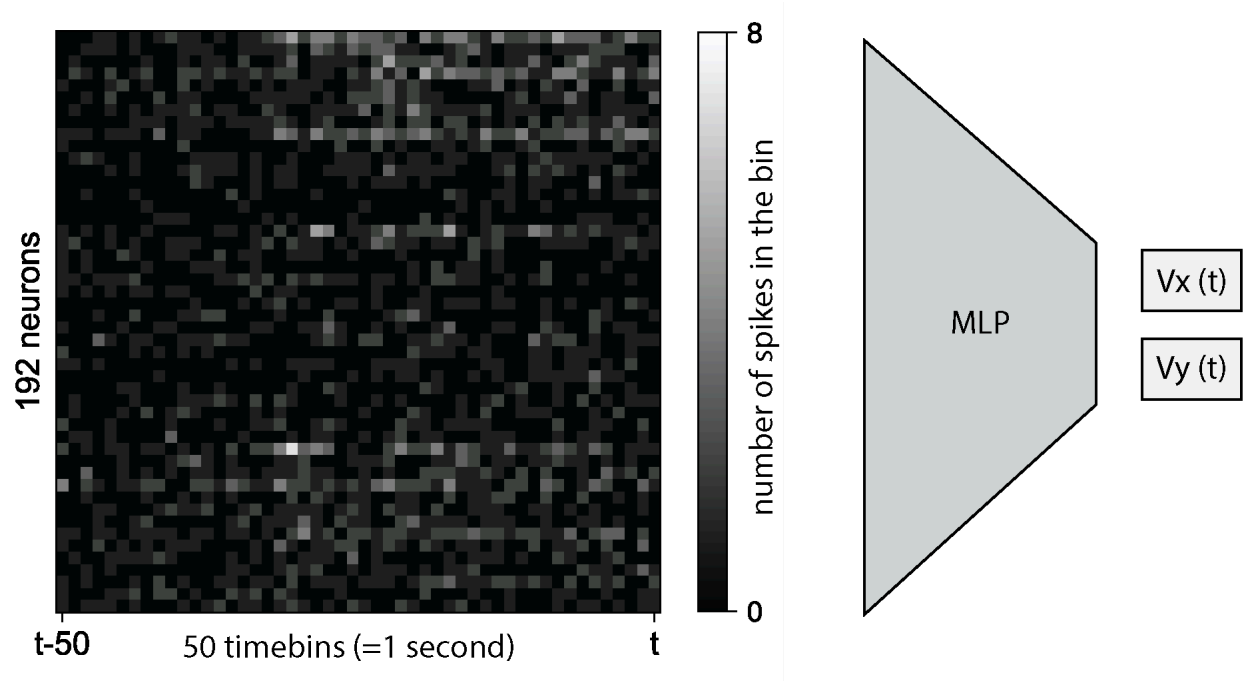

Figure 4. The decoding procedure.

Decoding is where we convert monkey brain data into the movements of the robotic arm. Whenever Jenkins wants to move his arm, neurons in his brain activate, calculating the trajectory of his planned movement, and this activation signal is sent through his spinal cord into the muscles in the monkey's arm. This sequence of events in time means that the movement of the arm depends on the recent history of neural data, and our model should take this fact into account. To build a model for predicting the monkey's hand movement from its brain recordings, first we split continuous time into discrete chunks (time bins) of 20 milliseconds each.

Then, for every time bin, we counted how many spikes occurred for every neuron recorded, which gave us 192 numbers (for 192 neurons) for every 20 ms. We also recorded the average velocity of the monkey's arm for every time bin, which gave us two numbers per 20 ms (x and y velocity). After this data processing step, we trained a simple MLP with two hidden layers (of dimensions 256 and 128) to take as input 50 time bins worth of the most recent neural data, corresponding to one second of time, and predict the velocities for the current time bin. The

decoding step turned out to be surprisingly easy; almost any architecture worked well, giving R^2 performance of around 0.9 on the test set.

To have a robotic arm move according to decoded neural data, we built the Koch v1.1 robotic follower arm. The model outputs velocities (Vx, Vy) which we integrate to obtain X and Y coordinates for the robot. The robotic arm consists of 6 servo motors in a chain, with the end effector (gripper) position determined by the servos' rotational positions. To achieve precise positioning, we implemented inverse kinematics using the ipky library. We defined a kinematic chain by specifying the servo motors' properties and the connecting link geometries. This chain configuration enables both forward kinematics (calculating end effector position from servo rotations) and inverse kinematics (determining required servo angles from desired coordinates).

As a result we have a continuous loop: neural data is decoded into velocities, converted to coordinates, processed through inverse kinematics to calculate necessary servo rotations, and finally transmitted to the robot for positioning.

Encoding: from Robot Movement to Neural Data

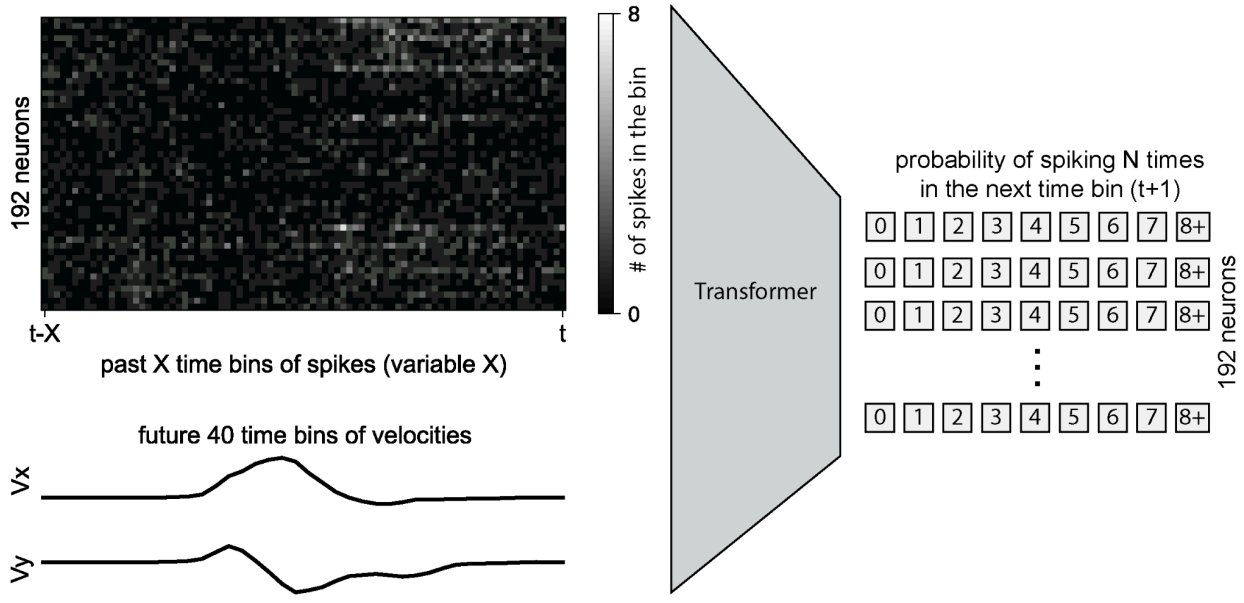

Figure 5. Generating synthetic neural data.

While training the decoding MLP was surprisingly easy, it was extremely hard to get this encoding model to work. Encoding is where we generate what Jenkins' brain would've looked like for any given movement of the arm. The key difficulty lies in closed-loop simulation: to continuously produce simulated neural data, the model needs to take its own past outputs as new

inputs, to produce future predictions based on what it has already said before. So if a model is a little bit off every time, then the errors from its output get fed back into the model, producing even larger errors in the next output, and so on. The errors can build up really quickly, and after even a fraction of a second (5-10 bins), the outputs of the model can stop looking like neural data, either blowing up to all be spiking like crazy, or dying down to zero activity. Success required fiddling with multiple different ways of formatting the inputs, finding the right architecture (transformer or LSTM) and training hyperparameters, as well as training for a long time (400 epochs).

Since the present movement of the arm depends on past neural data, it follows that the present neural data depends on the future (planned) movement of the arm. That is, to generate neural data, the model needs to know what the future arm movement will be. Accordingly, we process the data so that for every time bin, we input all of the past brain activity to the model, as well as the future 800 milliseconds (40 bins) of arm movement velocities. The model is trained to predict how many times each neuron will spike in the current 20ms time bin. We formulate this problem as a 9-way classification: either a neuron will spike 0 times (be quiet), or 1, 2, …, 7, or 8+ times. Like for next-token prediction in large language models, we train our encoding model to output what it thinks are the probabilities of each neuron's number of spikes being in any of our defined categories.

Conclusion

Project Jenkins demonstrates the feasibility of translating real neural activity into robotic movement and vice versa. By decoding signals from a macaque monkey’s brain, we successfully controlled a robotic arm, and by encoding movement into neural activity, we took steps toward realistic brain simulation. These methods are not limited to monkeys—similar approaches could be applied to humans, enabling direct brain control of robotic systems or even enhancing motor function in individuals with disabilities. While the model was trained on only eight reaching directions, preliminary experiments suggest it may generalize to more complex movements, such as drawing circles and simple shapes. In the future, further testing is needed to confirm its adaptability and refine its ability to handle novel motion patterns.

References

Kaufman, M., Churchland, M., Ryu, S. et al. (2014). Cortical activity in the null space: permitting preparation without movement. Nature Neuroscience, 17, 440–448. https://doi.org/10.1038/nn.3643

Churchland, Mark; Cunningham, John P.; Kaufman, Matthew T.; Foster, Justin D.; Nuyujukian, Paul; Ryu, Stephen I.; Shenoy, Krishna V. (2024) Neural population dynamics during reaching (Version draft) [Data set]. DANDI archive. https://dandiarchive.org/dandiset/000070/draft

Koch v1.1 robotic arm images. Retrieved from: https://github.com/jess-moss/koch-v1-1

Project Jenkins webpage: https://www.808robots.com/projects/jenkins