BMC Bioinformatics

RESEARCH

Open Access

Marigold: a machine learning-based web app for zebrafish pose tracking

Gregory Teicher[1,2*], R. Madison Riffe[1,3], Wayne Barnaby[1,3], Gabrielle Martin[1], Benjamin E. Clayton[1], Josef G. Trapani[3,4,5] and Gerald B. Downes[1,2,3*]

*Correspondence:
gteicher@umass.edu;
gbdownes@umass.edu

[1] Biology Department, University of Massachusetts Amherst, Amherst, MA, USA
[2] Molecular and Cellular Biology Graduate Program, University of Massachusetts Amherst, Amherst, MA, USA
[3] Neuroscience and Behavior Graduate Program, University of Massachusetts Amherst, Amherst, MA, USA
[4] Biology Department, Amherst College, Amherst, MA, USA
[5] Neuroscience Program, Amherst College, Amherst, MA, USA

Abstract

Background: High-throughput behavioral analysis is important for drug discovery, toxicological studies, and the modeling of neurological disorders such as autism and epilepsy. Zebrafish embryos and larvae are ideal for such applications because they are spawned in large clutches, develop rapidly, feature a relatively simple nervous system, and have orthologs to many human disease genes. However, existing software for video-based behavioral analysis can be incompatible with recordings that contain dynamic backgrounds or foreign objects, lack support for multiwell formats, require expensive hardware, and/or demand considerable programming expertise. Here, we introduce Marigold, a free and open source web app for high-throughput behavioral analysis of embryonic and larval zebrafish.

Results: Marigold features an intuitive graphical user interface, tracks up to 10 user-defined keypoints, supports both single- and multiwell formats, and exports a range of kinematic parameters in addition to publication-quality data visualizations. By leveraging a highly efficient, custom-designed neural network architecture, Marigold achieves reasonable training and inference speeds even on modestly powered computers lacking a discrete graphics processing unit. Notably, as a web app, Marigold does not require any installation and runs within popular web browsers on ChromeOS, Linux, macOS, and Windows. To demonstrate Marigold's utility, we used two sets of biological experiments. First, we examined novel aspects of the touch-evoked escape response in *techno trousers* (*tnt*) mutant embryos, which contain a previously described loss-of-function mutation in the gene encoding Eaat2b, a glial glutamate transporter. We identified differences and interactions between touch location (head vs. tail) and genotype. Second, we investigated the effects of feeding on larval visuomotor behavior at 5 and 7 days post-fertilization (dpf). We found differences in the number and vigor of swimming bouts between fed and unfed fish at both time points, as well as interactions between developmental stage and feeding regimen.

Conclusions: In both biological experiments presented here, the use of Marigold facilitated novel behavioral findings. Marigold's ease of use, robust pose tracking, amenability to diverse experimental paradigms, and flexibility regarding hardware requirements make it a powerful tool for analyzing zebrafish behavior, especially in low-resource settings such as course-based undergraduate research experiences. Marigold is available at: https://downeslab.github.io/marigold/.

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Keywords: Zebrafish, Behavior, Pose tracking, Machine learning, Web app, Software, Open source

Introduction

The kinematic and ethological analysis of animal movements is an essential task for many researchers in neuroscience, psychology, and related fields. For investigators working with zebrafish *(Danio rerio)*, behavioral assays are key methods for characterizing the toxicological effects of environmental pollutants, identifying therapeutic agents using small molecule screens, and modeling neurological disorders such as autism and epilepsy [6, 17, 51, 69]. Zebrafish embryos are spawned in large clutches and develop rapidly, making high-throughput screening in multiwell plates an attractive experimental paradigm. Additionally, zebrafish have a well characterized behavioral repertoire, a relatively simple nervous system containing fewer neurons compared to many other vertebrates, and orthologs to numerous human disease genes [7, 36, 40, 44]. Thus, zebrafish are an excellent system for studying vertebrate behavior.

Despite the rising popularity of behavioral screening in zebrafish, currently available software for analysis has a variety of limitations. Behavioral analysis software has traditionally utilized classical image processing techniques such as pixel intensity thresholding, background subtraction, skeletonization, and center of mass calculations to isolate animals from the visual background and track their movements [13, 14, 16, 43, 54, 59]. These techniques are often sufficient, but can lack robustness when challenged with dynamic backgrounds, varying lighting conditions, or earlier developmental stages when embryos present with less contrast against the background. Many solutions provide relatively limited kinematic information by only tracking a single point representing a larva's overall position, while those programs capable of tracking multiple points often fail when foreign objects enter the field of view. Approaches based on classical image processing techniques are particularly problematic for touch-evoked response assays that require a probe to enter the field of view, especially when the tactile stimulus is applied to the tail rather than the more easily detectable head. Additionally, many programs are closed source and/or tied to proprietary hardware such as specialized imaging cabinets, thus restricting access and customizability.

Several recently introduced programs have extended machine learning-based advances in human pose tracking to other species [27, 28, 30, 55, 62, 66, 67]. Such programs can achieve high accuracy, but they commonly require a high-end graphics processing unit (GPU) to run the large deep learning models efficiently. Additionally, many of these programs require workflows combining GUI-, command line-, and cloud-based elements, making such software difficult to use for those who lack programming, machine learning, or other technical expertise. These factors also limit the use of such programs in resource-constrained settings such as course-based undergraduate research experiences (CUREs), where such software might otherwise be useful. Lastly, due to the general-purpose nature of existing deep learning-based pose tracking programs, they often lack support for popular zebrafish experimental paradigms, such as imaging in multiwell plates, which can limit their utility for zebrafish researchers.

Here, we present Marigold, a free and open source machine learning-based web app for automated pose tracking of embryonic and larval zebrafish that unites many of the

strengths of existing programs, while also addressing several common limitations. Marigold reliably tracks up to 10 user-defined body keypoints, supports both single- and multiwell configurations, and features a user-friendly interface (Fig. 1). Marigold facilitates a streamlined workflow consisting of three stages: (1) generating a dataset by extracting a modest number of frames from representative behavioral recordings and labeling the frames with animal pose coordinates; (2) training a neural network using the generated dataset; and (3) analyzing new behavioral recordings using the trained neural network and exporting the resulting raw kinematic parameters and visualizations. Despite using convolutional neural networks, which are notoriously computationally demanding, Marigold is able to train models and analyze behavioral recordings at reasonable speeds on current desktop or laptop computers without requiring a dedicated GPU. This speed is achieved by using a highly efficient, custom neural network architecture incorporating a series of macro-architectural and micro-architectural optimizations. For each animal analyzed, Marigold generates publication-ready trajectory plots and a CSV file of keypoint coordinates, speeds, and angles between keypoints, which facilitates downstream analysis pipelines. Notably, as a web app, Marigold does not require any installation and is highly cross-platform, running within popular web browsers on ChromeOS, Linux, macOS, and Windows across a wide range of devices.

To showcase its robustness and versatility, we used Marigold to obtain biological insights from two sets of zebrafish experiments. First, we examined novel aspects of the embryonic touch-evoked response phenotype in the previously described *techno trousers (tnt)* mutant [29, 57], which contains a loss-of-function mutation in the gene encoding Eaat2b, a glial glutamate transporter. By applying the stimulus to either the head or the tail, we identify differences and interactions between genotype and touch location, potentially providing new clues into how neural circuits may be disrupted in this mutant. Second, we investigated the effects of developmental stage and feeding on larval

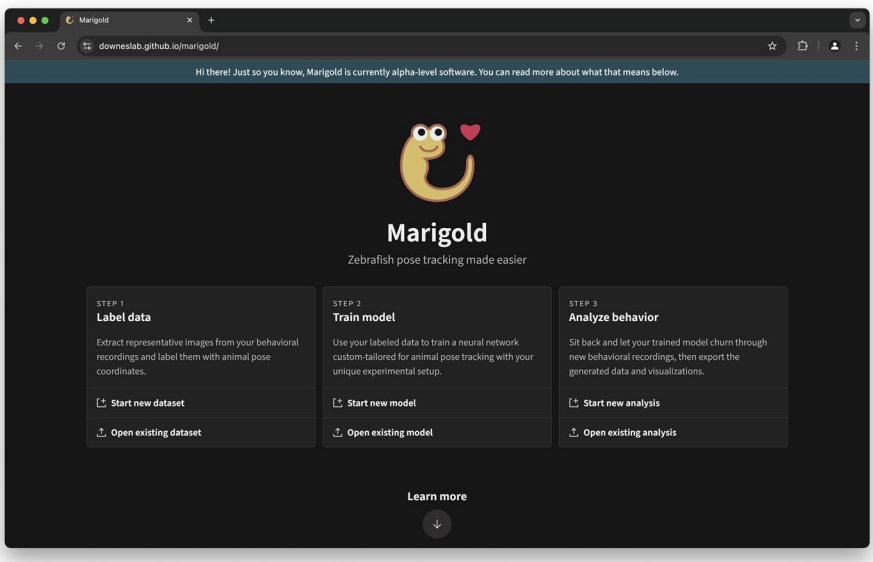

Fig. 1 Marigold features an intuitive, web-based graphical user interface (GUI). Marigold's welcome page (as viewed in Google Chrome on macOS) orienting the user to the workflow of labeling a dataset, training a model, and analyzing behavior

visuomotor response behavior. We identify differences and interactions between developmental stage and feeding regimen, highlighting the importance of considering both factors when designing and reporting the results of larval visuomotor response assays. In both biological experiments, the use of Marigold led to novel behavioral findings, demonstrating some of the types of biological questions that can be addressed using the detailed kinematic analysis provided by Marigold.

Methods

Zebrafish husbandry

All animals used in this study were obtained from our lab's fish facility at University of Massachusetts Amherst. Adult zebrafish were bred to obtain embryos and larvae for experimentation. Embryos and larvae younger than 5 dpf were maintained in the dark at 28.5 °C. Older fish were maintained on a 14 h light, 10 h dark cycle at approximately 28.5 °C. All wild type fish were from either Tübingen or Tüpfel long-fin (TL) strains as indicated. *techno trousers* (*slc1a2b*^{tk57}) fish [29, 57] were maintained on a TL background. All animal procedures were approved by the University of Massachusetts Amherst Institutional Animal Care and Use Committee (IACUC) under Animal Welfare Assurance number A3551-01. No animal procedures were performed at Amherst College.

Dataset collection and labeling

Two novel datasets were developed: one for touch-evoked responses and one for visuomotor responses. For both datasets, embryos were obtained from mass matings of Tübingen or TL fish. Behavior was monitored using an IDT X-Stream 1440p PCIe 2.0 high speed camera (Integrated Design Tools, Inc., Pasadena, CA, USA). Larvae were imaged from above and illuminated using a white light or near-infrared source from below. Recordings were made under varying conditions of lighting intensity, resolution, magnification, and focus.

From the generated movies, frames were manually selected for inclusion in the datasets with the goal of maximizing the representation of diverse poses and experimental conditions. Each frame was then manually labeled and the labeled coordinates were used to generate target heatmaps (i.e., two-dimensional Gaussians) that served as the "ground truth" for our neural networks during training.

For the touch-evoked response dataset, movies were recorded of embryos and larvae between the developmental stages of 2–4 dpf in wells in the inverted lids of 24-well plates. Labels targeted 7 keypoints evenly spaced across the rostral caudal axis. For the visuomotor response dataset, movies were recorded of larvae between the developmental stages of 5–7 dpf in 24-well plates. Labels targeted a single keypoint positioned between the two eyes.

Software development

Marigold was written in HTML, CSS, JavaScript, and C++ (with the latter used to generate WebAssembly) and is licensed under the GNU General Public License version 3 (GPLv3). No external libraries or components were used or incorporated other than the Adobe Source Sans 3 font (used under the SIL Open Font License Version 1.1) (Adobe, n.d.), the plasma and viridis colormaps (used under the CC0 "No Rights Reserved"

license) (Smith et al., n.d.), and the SplitMix64 [80] and xoshiro256** [8] pseudorandom number generators (both used under the CCO "No Rights Reserved" license) (Blackman & Vigna, n.d.). Building for deployment was automated using GitHub Actions, with hosting provided by GitHub Pages.

Measurement and calculation of performance metrics

Performance measurements were obtained using a Lenovo Thinkpad P14s Gen 4 laptop equipped with an AMD Ryzen 7 PRO 7840U central processing unit (CPU) and running Fedora Workstation 41 with Linux kernel version 6.11.7. PyTorch-based performance measurements were obtained using Python 3.13.0, NumPy 2.1.3 [34], pandas 2.2.3 [58], SciPy 1.14.1 [92], PyTorch 2.5.1 [2], and torchvision 0.20.1 [87]. WebAssembly-based performance measurements were obtained using the LLVM (version 19.1.0) Clang compiler and wasm-ld linker, with the generated WebAssembly running in Google Chrome 131. All performance measurements reflect single-threaded CPU performance.

Training speeds reflect the time required to complete a forward and backward pass through the neural network in training mode at the specified batch size. Inference speeds reflect the time required to complete a forward pass through the neural network in evaluation mode at the specified batch size. Notably, these measurements do not include the time required to perform data augmentation or gather data into batches.

Theoretical minimum memory footprints correspond to a hypothetical implementation in which all required memory buffers must be allocated before training begins and cannot be freed or repurposed until training has ended, individual computations are executed with the minimum possible number and size of memory buffers (e.g., by using implicit rather than explicit padding for depthwise convolutions), and gradient accumulation is used to the fullest extent possible wherever it can be used without impacting the results (i.e., in the absence of Batch Normalization).

Neural network design and training

Neural networks were trained at an input resolution of 512×512 (touch-evoked response dataset) or 256×256 (visuomotor response dataset). All MobileNetV3 blocks and variations used an expansion ratio of 2, with the number of channels as input to the block set to 32 (touch-evoked response dataset) or 16 (visuomotor response dataset) when operating at the smallest spatial resolution, except where otherwise noted. For the hierarchical macro-architecture, intro blocks consisted of a 4×4 convolution with 2×2 stride followed by a normalization layer. Outro blocks consisted of a 1×1 convolution layer with a channel expansion ratio of 2, a hard swish layer, a spatial dropout layer with drop probability $= 0.05$, and a 1×1 convolution layer reducing the channels to the number of keypoints. Downsampling and upsampling blocks consisted of a 2×2 convolution with 2×2 stride or a 2×2 transposed convolution with 2×2 stride stride, respectively, followed by a normalization layer. For the isotropic macro-architecture, intro blocks consisted of a 8×8 pixel unshuffle layer, a 1×1 convolution layer, and a normalization layer, except where otherwise noted. Outro blocks consisted of a 1×1 convolution layer with a channel expansion ratio of 2, a hard swish layer, a spatial dropout layer with drop probability $= 0.05$, a 1×1 convolution layer reducing the channels to $16 \times$ the number of keypoints, and a

4×4 pixel shuffle layer transforming the number of channels to the number of keypoints while increasing the spatial resolution, except where otherwise noted.

Training was performed using mean squared error (MSE) loss, a batch size of 16, and the AdamW optimizer [49] with learning rate $=5.0\times10^{-4}$ (touch-evoked response dataset) or 1.0×10^{-4} (visuomotor response dataset), $\beta1=0.9$, $\beta2=0.95$, and $\varepsilon=1.0\times10^{-6}$. Weight decay $=1.0\times10^{-5}$ was applied to the weights of all convolution layers. Convolution weights were initialized using Xavier initialization [26], with the exception of the final convolution layer which was initialized to zeros. When a convolution layer was followed by a nonlinearity or by a normalization and then nonlinearity, the initialization gain was set to the square root of two, with the gain otherwise set to one. Additionally, the initialization gain for the last convolution layer in each residual block was scaled by the reciprocal of the square root of the number of residual blocks, as suggested by [53]. All convolution biases were initialized to zeros. For all normalization layers, ε was set to 1.0×10^{-3}, which matches the value used for Batch Normalization in MobileNetV3 [38] and was found to have a stabilizing effect on training compared to the PyTorch default value of 1.0×10^{-5}. The PyTorch default value of momentum $=0.9$ was used for Batch Normalization layers.

Data augmentation

Datasets were randomly divided into training (75%) and validation (25%) splits. On-the-fly data augmentation [75] was applied to images in the training split by randomly adjusting gamma and brightness, randomly resizing, randomly flipping vertically and horizontally, randomly rotating, randomly padding and cropping, and (after standardization) randomly adding a small amount of Gaussian noise. Additionally, a small amount of "wiggle" was randomly applied to keypoint coordinates during generation of the two-dimensional gaussian labels. Data augmentation was not applied to images or labels in the validation split. All images were standardized using the mean and standard deviation calculated over the training split.

Touch-evoked response assay

Embryos were obtained from mass matings of heterozygous *techno trousers* (*slc1a2b*$^{+/tk57}$) fish. Behavior was monitored using an IDT X-Stream 1440p PCIe 2.0 high speed camera (Integrated Design Tools, Inc., Pasadena, CA, USA) in a temperature controlled setting (25 °C). Larvae were transferred in E3 solution to a well in the inverted lid of a CytoOne 24-Well Tissue Culture Plate (USA Scientific, Inc., Ocala, FL, USA) and allowed to acclimate briefly before touching on either the side of the head or tip of the tail as indicated with a 3.22/0.16 g of force von Frey filament held by a surgical blade holder. Larvae were imaged from above and illuminated using a white light source from below. Responses were recorded at a resolution of 1024×1024 pixels with a framerate of 1000 Hz and exposure time of 0.8 ms. Recordings were manually trimmed to remove extraneous frames before and after the response, as determined by the first and last visible movements initiated by the fish, respectively.

slc1a2b genotyping

Following behavioral analysis, fish were euthanized using an overdose of MS-222 (pH = 7.0) (Sigma-Aldrich, St. Louis, MO, USA). DNA was extracted from the euthanized larvae using Extract-N-Amp Tissue PCR Kit (Sigma-Aldrich, St. Louis, MO, USA) according to the manufacturer's protocol. A 192 bp region of exon 2 of *slc1a2b* flanking the site encoding the A393V mutation was amplified by PCR using the forward primer 5'-TGCTGGAACTCTGCCCGTGA-3' and the reverse primer 5'-ACG GTGACGATCTGTCCAGG-3'. PCR was performed using AmpliTaq Gold DNA Polymerase (Applied Biosystems/Thermo Fisher Scientific, Waltham, MA, USA) according to the manufacturer's protocol and using an annealing temperature of 59 °C. The PCR product was digested overnight using Fnu4HI (New England Biolabs, Ipswich, MA, USA) according to the manufacturer's protocol. Digestion yielded a single fragment of 192 bp for homozygous mutant fish ($slc1a2b^{tk57/tk57}$), two fragments of 133 and 59 bp for homozygous wild type fish ($slc1a2b^{+/+}$), and three fragments of 192, 133, and 59 bp for heterozygous fish ($slc1a2b^{+/tk57}$). The digested PCR products were resolved for genotypic determination by agarose gel electrophoresis.

Visuomotor response assay

Embryos were obtained from mass matings of Tübingen strain zebrafish and maintained in E3 media for the duration of experiments. Larvae were kept in petri dishes until 5 dpf, at which point they were transferred to 2.5 L fish tanks at a density of approximately 45 fish/tank and maintained on a 14 h light, 10 h dark light cycle. The initial volume of E3 media in the tanks was 250 mL, with an additional 250 mL added daily. Fish in fed conditions were given approximately 50 mg of Gemma Micro 75 (Skretting, Stavanger, Norway) once daily beginning at 5 dpf and continuing through 7 dpf. Feeding occurred at approximately 12:00 PM each day, with behavioral recordings occurring between 4 and 6 h after feeding. Behavior was monitored using an IDT X-Stream 1440p PCIe 2.0 high speed camera (Integrated Design Tools, Inc., Pasadena, CA, USA) in a temperature controlled setting (25 °C). Larvae were transferred to the wells of a CytoOne 24-Well Tissue Culture Plate (USA Scientific, Inc., Ocala, FL, USA) for recording. Larvae were imaged from above and illuminated from below using a custom-built light source which was toggled between white and near-infrared illumination modes. Fish were shielded from ambient light by an enclosing cabinet. Responses were recorded at a resolution of 1440 × 1440 pixels with a framerate of 100 Hz and exposure time of 0.8 ms.

Statistical analysis and data visualization

Statistical significance was determined by two-way analysis of variance (ANOVA) with Tukey post hoc test using the R language (version 4.4.2) [64]. Data visualizations were generated using the ggplot2 library (version 3.5.1) for R [95].

Results

Neural network design

Arguably, modern machine learning is dominated by the massively parallel training of extremely overparameterized deep neural networks on colossal datasets using

high-end GPU clusters [31, 85]. This paradigm is at odds with common use case scenarios in resource-constrained settings such as course-based undergraduate research experiences (CUREs) and academic life science research labs, in which hardware often lags behind the cutting edge and domain-specific datasets frequently must be collected and labeled de novo.

To reduce the processing and memory demands of deep convolutional neural networks sufficiently to make in-browser, on-device zebrafish pose tracking feasible with minimal hardware requirements and limited data availability, we collected and labeled two proof-of-principle datasets and used them to develop and evaluate a series of neural network architectural variants. Our datasets were designed to reflect behavioral assays commonly used within our lab and were composed of either labeled frames from touch-evoked response movies, in which case labels consisted of 7 keypoints evenly spaced across the rostral–caudal axis, or labeled frames from visuomotor response movies, in which case labels consisted of a single keypoint located between the two eyes (Fig. 2). Our neural networks were prototyped using PyTorch [2], a highly optimized machine learning library for Python, and evaluated based on training dynamics, training speed, inference speed, parameter count, and theoretical minimum memory footprint. We chose to evaluate training and inference speeds primarily based on single-threaded CPU performance as this represents a "lowest common denominator" target across the wide variety of platforms, devices, and research settings we aimed to support.

As an efficient and well-established baseline building block for our neural network architectural design experiments, we adopted the MobileNetV3 inverted bottleneck residual block [38]. As the most recent member of the MobileNet family of blocks designed for fast CPU inference on mobile devices [38, 72, 39], the MobileNetV3 design improved on that of MobileNetV2 by optionally incorporating larger

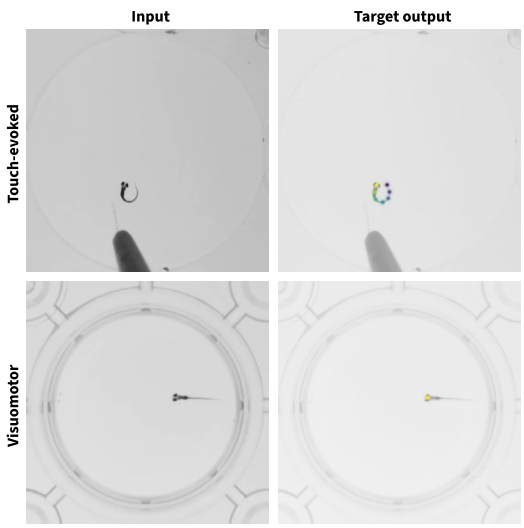

Fig. 2 Examples of input–output pairs from the touch-evoked response and visuomotor response datasets. Representative images from each dataset visualized with and without overlaid pseudocolored gaussians corresponding to labeled keypoints

depthwise convolution kernel sizes, the Hard Swish activation function, and the Squeeze and Excitation module.

Having selected the MobileNetV3 inverted bottleneck residual block as our baseline building block, we proceeded to incorporate it into two potential macro-architectural designs (Fig. 3A). In the first approach, we adopted a hierarchical architecture inspired by neural networks such as the Hourglass and Simple Baseline architectures for human pose estimation and the U-Net architecture for semantic segmentation of medical images [61, 68, 98]. In this architecture, as information flows through the successive layers of the neural network, it is first downsampled, then upsampled. The downsampling

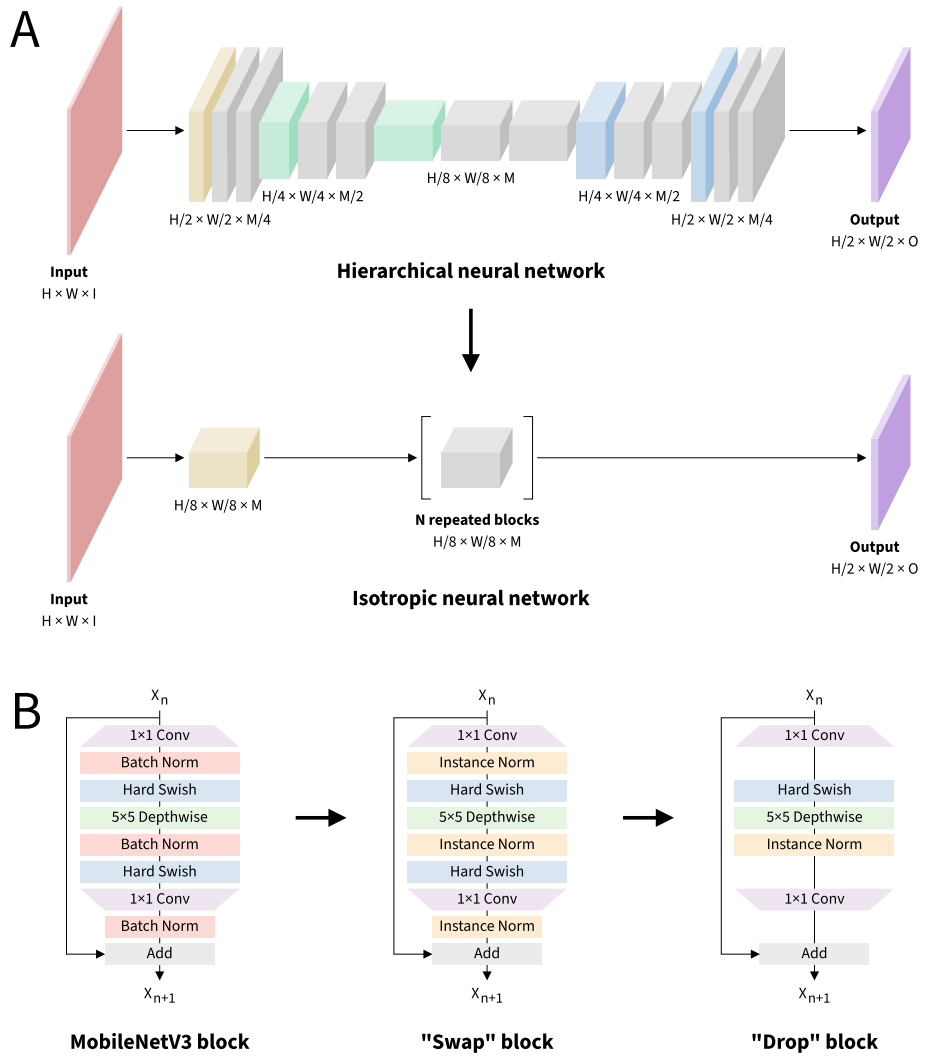

Fig. 3 Overview of macro-architectural and micro-architectural design decisions leading to a simple, efficient, and easily customizable neural network architecture. (A) Schematic of the macro-architectural design process illustrating the implementations of hierarchical and isotropic neural networks. Red indicates input images, yellow indicates intro blocks, green indicates downsampling blocks, blue indicates upsampling blocks, purple indicates outro blocks, and gray indicates residual blocks. H: input height; W: input width; I: input channels; M: middle channels; O: output channels; N: residual blocks. (B) Schematic of the micro-architectural design process illustrating the implementations of the MobileNetV3 inverted bottleneck residual block and subsequent modifications. Small arrows indicate the flow of information through the macro- and micro-architectures, while large arrows indicate the flow of the design process

path minimizes the computational cost of the expensive convolutional layers and aggregates spatial context, while the upsampling path restores the resulting representation to the larger spatial resolution required to produce accurate probability heatmaps for each body keypoint of interest. In the second approach, we adopted an isotropic architecture [71]. In contrast to hierarchical architectures which make use of multiple internal resolutions, isotropic architectures utilize a constant internal resolution, essentially performing all processing on non-overlapping image "patches". Isotropic architectures also remain relatively unexplored, particularly for computer vision tasks other than image classification. Interestingly, an isotropic architecture was recently used for keypoint-based fish morphometric analysis [70], suggesting its potential utility for animal pose estimation. We tailor our architectures to our two datasets as described in Table 1.

Despite its simplicity, the isotropic architecture exhibited compelling advantages over its hierarchical counterpart (Fig. 4). Across both datasets, the isotropic architecture reached lower loss values in less time (Fig. 4A–B) and achieved severalfold faster training and inference throughput (Fig. 4C–D). Although the isotropic architecture contains a greater number of trainable parameters, its theoretical minimum memory footprint is considerably smaller (Fig. 4E–F). To help tease apart any intrinsic benefit of patch-based processing from that of operating at a specific internal resolution, we also explored using patches larger and smaller than the 8×8 patches corresponding to the smallest internal resolution of our hierarchical architecture, adjusting the number of channels in inverse proportion to the number of spatial pixels to control for the overall processing and memory demands. Smaller patch sizes led to considerably slower training convergence while larger patch sizes led to similar training convergence but with lower inference speed; thus, 8×8 patches appeared to provide the most optimal balance across our metrics of interest (Supplementary Fig. 1). Altogether, moving from a hierarchical architecture to an isotropic architecture appears to be a net positive design decision.

With our macro-architectural design strategy in place, we turned our attention to micro-architectural improvements to the MobileNetV3 block (Figs. 3B and 5). As our first micro-architectural design step, we explored alternatives to the Batch Normalization layers used in the MobileNetV3 block. Batch Normalization is widely adopted in neural network architectures for computer vision due to its ability to improve training dynamics and help prevent overfitting [41]. During training, the computational overhead introduced by Batch Normalization leads to slower per-epoch training throughput, but this is generally outweighed by the resulting improvement in per-epoch training convergence. At inference time, the computational overhead can be avoided by "folding"

Table 1 Neural network configurations. Designations in parentheses correspond to those in Fig. 3A

Dataset	Touch-evoked	Visuomotor
Input resolution (H×W)	512×512	256×256
Output resolution (H/2×W/2)	256×256	128×128
Input channels (I)	1	1
Middle channels (M)	32	16
Output channels (O)	7	1
Residual blocks (N)	10	10

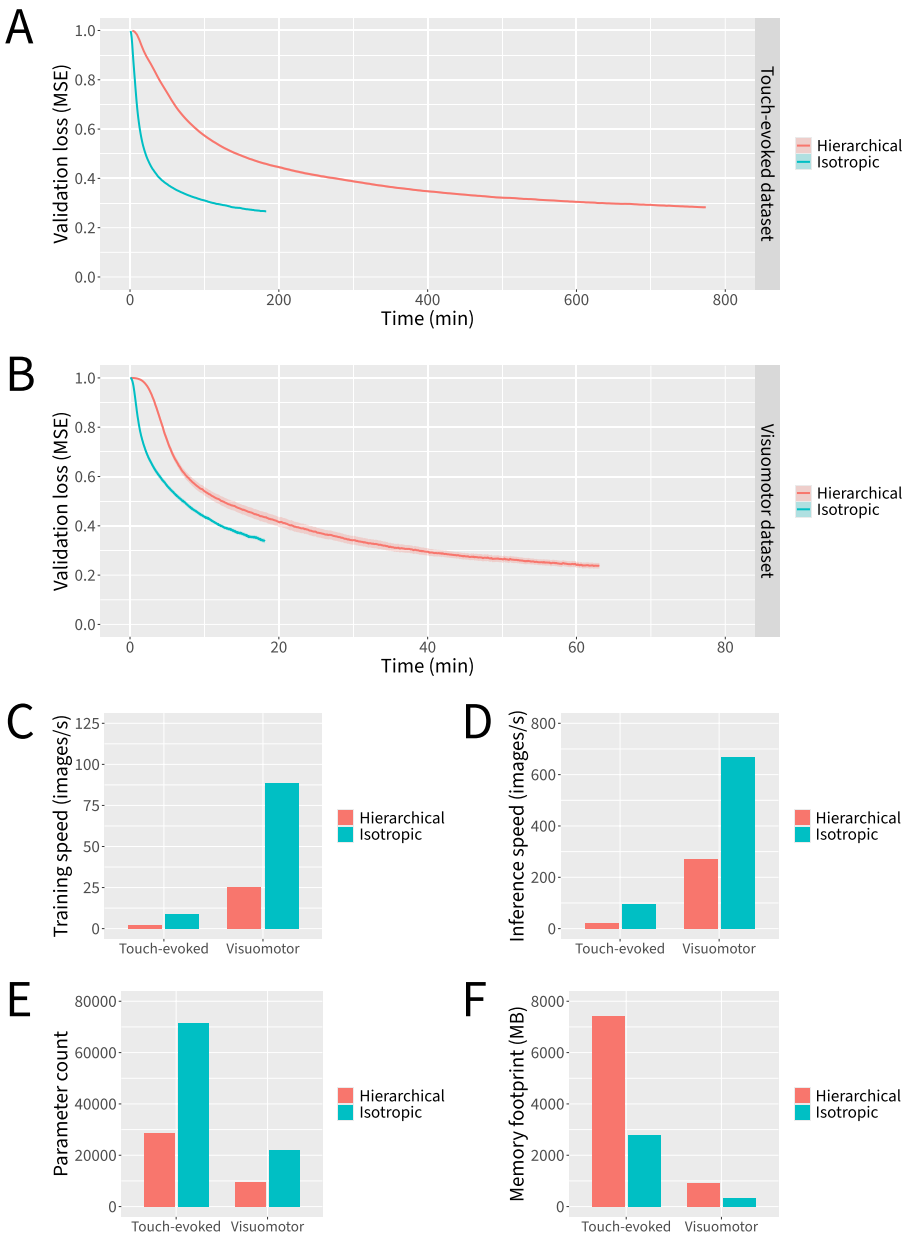

Fig. 4 Isotropic neural network macro-architectures outperform more traditional hierarchical neural network macro-architectures by multiple metrics. Neural networks were trained for 6000 parameter updates for each dataset and macro-architecture. **A** Loss curves for the touch-evoked response dataset. **B** Loss curves for the visuomotor response dataset. **C** Training speeds. **D** Inference speeds. **E** Parameter counts. **F** Theoretical minimum memory footprints. Data represent the mean or the mean plus and minus the standard error of the mean of 10 independent experiments in which datasets were partitioned and neural network weights were initialized using different random seeds

Batch Normalization layers into adjacent convolutional layers. Thus, Batch Normalization generally leads to more efficient training without negatively impacting inference speed. However, the dependence of Batch Normalization on the batch size used during training leads on the one hand to markedly poorer training convergence at smaller batch sizes and, on the other hand, to a larger training memory footprint at large batch

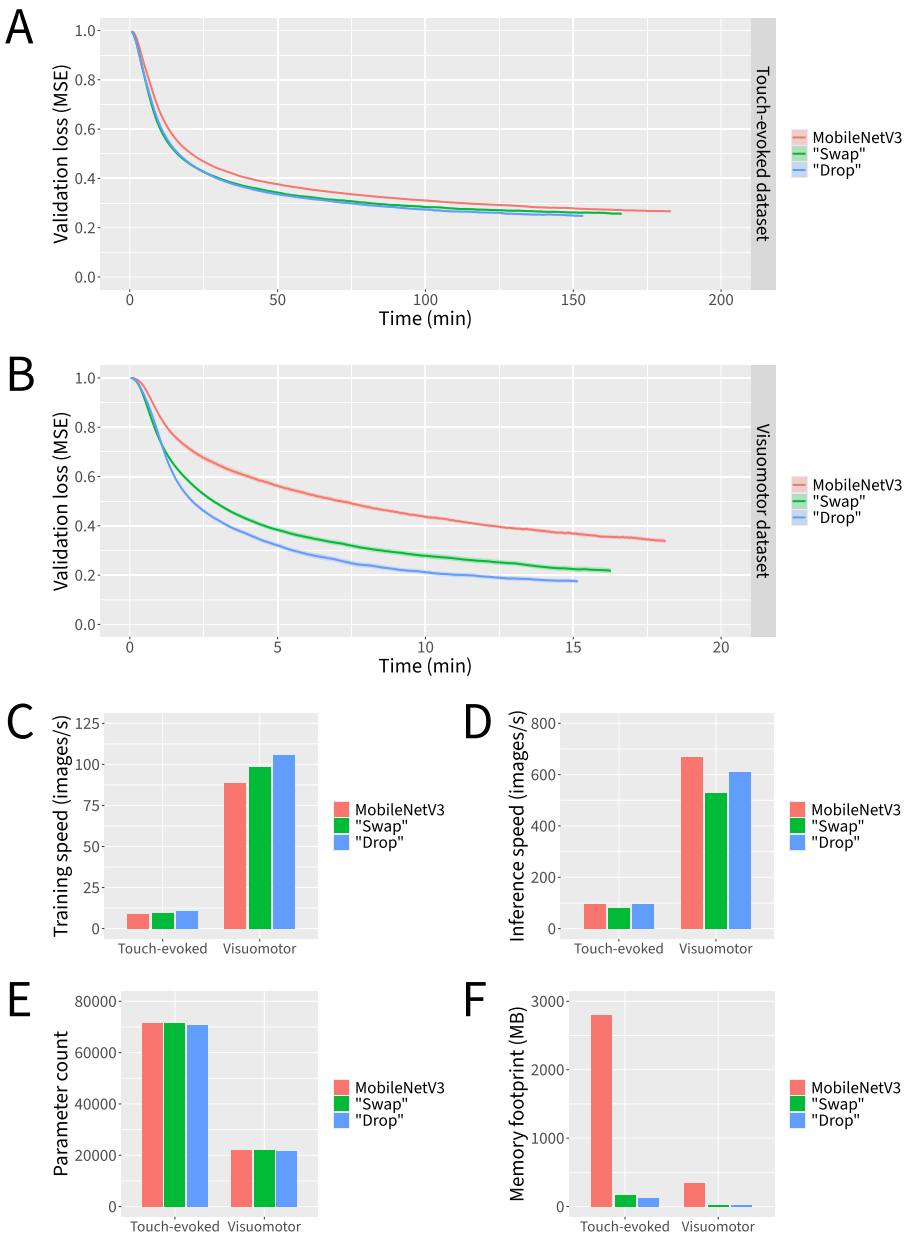

Fig. 5 Micro-architectural improvements to the MobileNetV3 block improve training dynamics and reduce memory requirements at minimal cost to inference speed. Neural networks were trained for 6000 parameter updates for each dataset and micro-architecture. **A** Loss curves for the touch-evoked response dataset. **B** Loss curves for the visuomotor response dataset. **C** Training speeds. **D** Inference speeds. **E** Parameter counts. **F** Theoretical minimum memory footprints. Data represent the mean or the mean plus and minus the standard error of the mean of 10 independent experiments in which datasets were partitioned and neural network weights were initialized using different random seeds

sizes, with the potential to exceed the limited amount of memory that web apps are permitted to allocate. By contrast, proposed alternatives to Batch Normalization, such as Layer Normalization [5], Group Normalization [97], and Instance Normalization [89], operate independently of batch size, enabling the use of gradient accumulation to maintain a constant training memory footprint independent of batch size. However, these

alternatives cannot be "folded" into preceding convolutional layers, introducing computational overhead at inference time. In our experiments, Layer Normalization, Group Normalization, and Instance Normalization all led to competitive training convergence compared to Batch Normalization at a modest cost to inference speed (Supplementary Fig. 2). However, "swapping" from Batch Normalization to Instance Normalization resulted in the greatest improvement in training dynamics across both datasets and thus was incorporated into our final architectural design (Figs. 3B and 5).

Next, we asked whether it was possible to recover some of the inference speed that was lost as a result of the switch from Batch Normalization to Instance Normalization without negatively impacting the training dynamics (Figs. 3B and 5). A logical way to accomplish this would be to "drop" some of the components of the MobileNetV3 block. We thus performed a series of ablation studies to determine which components were most dispensable. We were surprised to find that "dropping" two of the three Instance Normalization layers and one of the two Hard Swish layers actually led to more favorable training dynamics than any other design iteration, while also recovering much of the inference speed (Figs. 3B and 5). We note that the choices of which Instance Normalization layer and Hard Swish layer to keep were crucial in this design step, as other combinations did not perform as well (Supplementary Fig. 3).

Altogether, our series of macro-architectural and micro-architectural design steps resulted in markedly faster training and inference speeds, with substantially reduced theoretical memory footprints, compared to our already reasonably efficient baseline. Given that similar results were obtained across both datasets, our final architecture appears to be flexible enough to accommodate a range of experimental paradigms. All the more so considering that the isotropic nature of our architecture largely decouples design parameters such as input resolution, middle resolution, output resolution, number of middle channels, number of residual blocks, and residual block channel expansion ratio, thus supporting a large degree of potential customization.

Training data requirements

Given that manually labeling data is often the most labor-intensive step in a supervised machine learning workflow, we next asked to what extent similar results could be obtained with less training data available. We investigated this by incrementally ablating images from the training split of each of our two datasets while controlling for the total number of parameter updates (Fig. 6). To fairly evaluate each model's ability to generalize to previously unseen data, we held the number of images in the validation split constant.

Across both datasets, ablating increasing amounts of training data led to steadily decreasing training efficacy, with a disparity between the validation and training partitions in later stages of training indicating greater degrees of overfitting (Fig. 6A–B). To determine to what extent this affects the accuracy of predictions, we quantified the accuracy of predictions when varying amounts of training data are available (Fig. 6C–D). Because our datasets include a number of images with extreme variations in lighting conditions, occlusions, and other artifacts beyond what would normally be observed in data collected for routine biological experiments within our lab, the keypoints in some of these extreme images are not predicted well even by

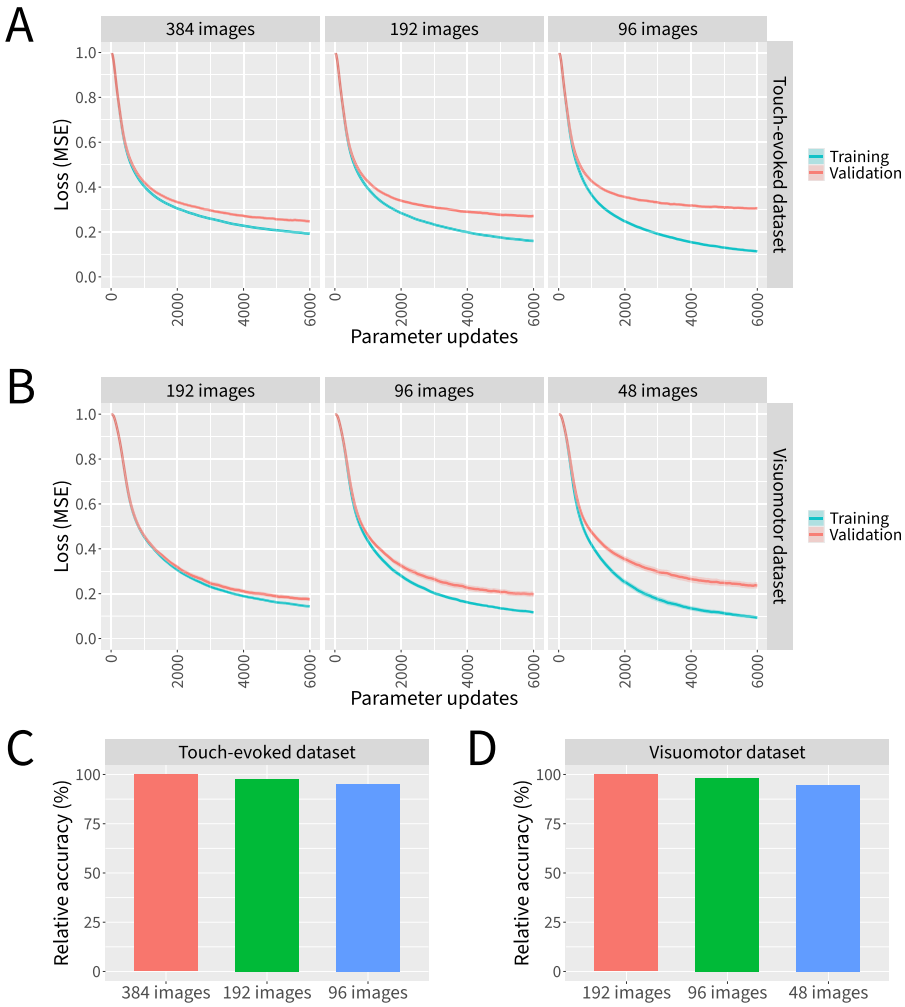

Fig. 6 Marigold's neural networks can be trained effectively even with limited training data. Neural networks were trained for 6000 parameter updates on increasingly ablated sets of training images while evaluating performance on a constant number of validation images. **A** Loss curves for the touch-evoked response dataset. **B** Loss curves for the visuomotor response dataset. **C** Relative accuracies for the touch-evoked response dataset. **D** Relative accuracies for the visuomotor response dataset. Data represent the mean or the mean plus and minus the standard error of the mean of 10 independent experiments in which datasets were partitioned and neural network weights were initialized using different random seeds

neural networks trained with all available training data. To account for this effect, we report relative accuracy as the ratio of correctly predicted keypoints for ablated datasets relative to the number of correctly predicted keypoints when all training data is available. We consider predictions to be correct when they are within 3 pixels from the ground truth label for the touch-evoked response dataset or within 1.5 pixels from the ground truth label for the visuomotor response dataset.

As data is ablated, there appears to be a steady increase in the final loss values (Fig. 6A–B) and a corresponding dropoff in the number of correctly predicted keypoints (Fig. 6C–D), but under the tested conditions we did not observe a catastrophic failure to converge even in the face of extremely scarce training data. We conclude that our neural networks can be trained effectively even with limited training data,

particularly given that our datasets contain several orders of magnitude fewer images than many conventional computer vision datasets. However, it is likely that refinements to our data augmentation pipeline and/or adjustments to the strength of regularization techniques such as weight decay and dropout could lead to improved performance when data is more limited. Additionally, whereas we included extreme examples in our datasets to make them more challenging, better results could likely be obtained with fewer images by restricting the dataset to images from more representative behavioral recordings.

WebAssembly implementation

Having optimized our neural network architecture using our PyTorch-based implementation and verified that it can be effectively trained with relatively little training data, we next focused on implementing this functionality within our web app. To achieve this, we implemented the necessary neural network layer and training operations in C++ and used this to generate WebAssembly, which allows such code to run within a browser at speeds approaching those of native C++ code [33, 63]. We also explored existing neural network libraries available to web apps, but found that such libraries are generally limited in functionality compared to those available in Python and other environments. TensorFlow.js [77], for example, was missing implementations for several of the required layers for our chosen architecture. It also did not support gradient accumulation, which would have forced the use of smaller, less effective batch sizes to minimize our web app's memory footprint. We also observed that many neural network implementations for the

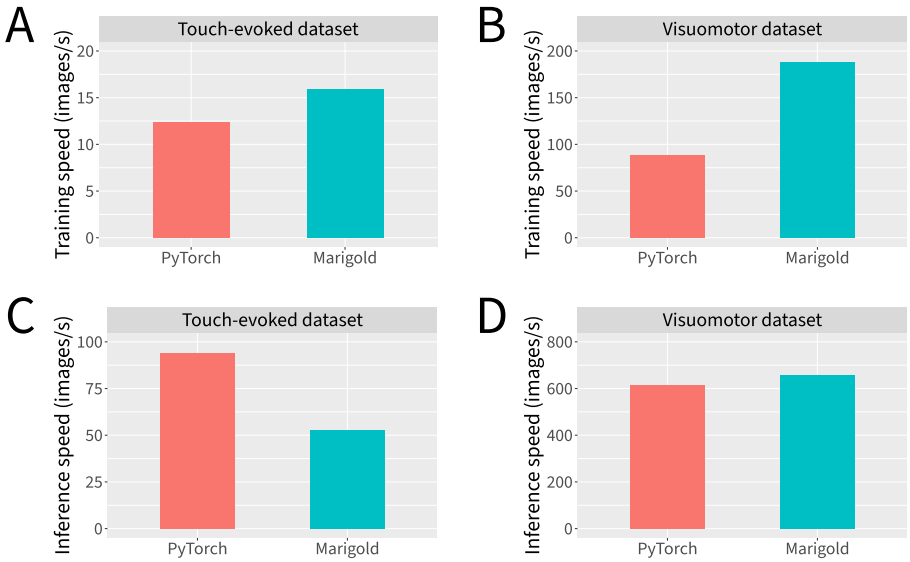

Fig. 7 Marigold's WebAssembly-based neural network implementation achieves CPU performance competitive with that of PyTorch. **A** Training speeds for the touch-evoked response dataset. **B** Training speeds for the visuomotor response dataset. **C** Inference speeds for the touch-evoked response dataset. **D** Inference speeds for the visuomotor response dataset. Measurements reflect single-threaded CPU performance. Training speeds were measured using gradient accumulation with a batch size of 1 to obtain an effective batch size of 16. Inference speeds were measured with a batch size of 1. Data represent the mean of 10 independent measurements

web, such as the emerging WebNN standard [94], are primarily focused on inference and lack support for training.

To evaluate the effectiveness of our WebAssembly-based approach, we compared the resulting training and inference speeds to those of our PyTorch implementation (Fig. 7). In terms of training speed, Marigold slightly to moderately outperformed PyTorch across neural networks configured for both the touch-evoked response and visuomotor response datasets (Fig. 7A–B). In terms of inference speed, Marigold slightly outperformed PyTorch for neural networks configured for the visuomotor response dataset, but was moderately outperformed by PyTorch for neural networks configured for the touch-evoked response dataset (Fig. 7C–D). Overall, our implementation achieves speeds that are competitive on CPU with the highly optimized PyTorch library. This is particularly noteworthy given the relatively limited access to memory and CPU features that web apps are permitted.

Effects of genotype and touch location in *slc1a2b*$^{tk57/tk57}$ and wild-type sibling touch-evoked responses

Using Marigold, we investigated novel aspects of the *techno trousers* (*tnt*) locomotor phenotype [29, 57]. This mutant, henceforth referred to as *slc1a2b*$^{tk57/tk57}$, harbors a loss-of-function missense mutation in *slc1a2b* (Ensembl ID: ENSDARG00000102453) located on chromosome 25, which encodes Eaat2b, a glutamate transporter predominantly expressed in astroglia [37, 57]. We note that this gene and its protein product have also been referred to as *slc1a2a* and Eaat2a, respectively [25, 37]; however, current genome databases indicate that these particular identifiers refer to a distinct gene found on chromosome 7 (Ensembl ID: ENSDARG00000052138) and its protein product [10]. *slc1a2b*$^{tk57/tk57}$ embryos demonstrate a hyperactive response to touch by 48 hpf, but the larvae are effectively paralyzed and shorter along the rostral–caudal axis by 96 hpf [57]. Small twitch responses to touch can be observed at 96 hpf, suggesting the lack of response is due to motor issues rather than sensory issues. Because the auditory and visual systems are not fully developed by 96 hpf, tactile stimulation at earlier time points is the most robust way to elicit the hyperactive phenotype.

Reticulospinal neurons, such as the Mauthner cell and its homologs (collectively referred to as the Mauthner array), are the primary mediators of the short latency touch-evoked response. Head-stimulated responses recruit the full Mauthner array while tail-stimulated responses recruit the Mauthner cell alone, resulting in kinematically distinct responses based upon touch location [46]. By comparing touch-evoked responses of head- and tail-stimulated *slc1a2b*$^{tk57/tk57}$ embryos, we reasoned that we could uncover novel aspects of the *slc1a2b*$^{tk57/tk57}$ locomotor phenotype.

Following head stimulation, wild-type embryos reliably exhibit a high amplitude body bend, reorienting the embryo, followed by lower amplitude undulations, allowing the embryo to swim away from the perceived threat (Fig. 8A). Head-stimulated *slc1a2b*$^{tk57/tk57}$ embryos swim significantly longer and farther than wild-type controls (two-way ANOVA, $F(1,86) = 59.58$, $p < 0.001$ and $F(1,86) = 57.81$, $p < 0.001$ for duration and distance, respectively) (Fig. 8B–C). Consistent with previous literature [57], *slc1a2b*$^{tk57/tk57}$ embryos perform more high amplitude body bends during the escape response (two-way ANOVA, $F(1,86) = 35.88$, $p < 0.001$) (Fig. 8D). However, the number of high amplitude

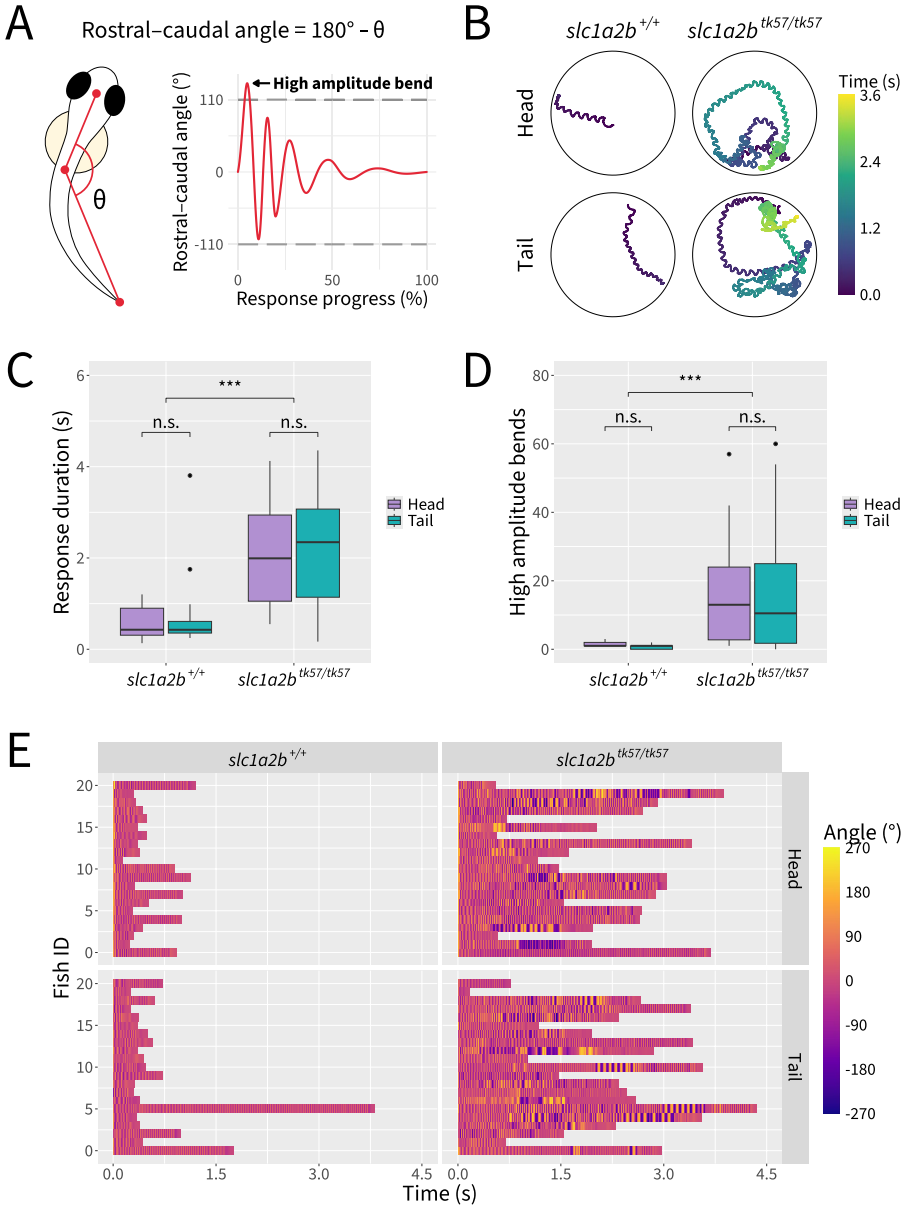

Fig. 8 *slc1a2b*$^{tk57/tk57}$ mutant zebrafish exhibit hyperactive touch-evoked startle behavior regardless of the touch location. **A** Schematic illustrating calculation of rostral–caudal angle and quantification of high amplitude body bends. **B** Representative trajectory traces. **C** Quantification of response durations. **D** Quantification of high amplitude body bends. **E** Visualization of individual rostral–caudal angles over time, with three fish omitted at random in order to show equal numbers of fish across conditions. Statistical significance determined by two-way ANOVA with Tukey post hoc; n.s.: not significant; ***: $p < 0.001$; n = 21–24 fish per condition across 3 independent experiments

body bends performed across head- and tail-stimulated embryos did not vary by genotype (two-way ANOVA, $F(1,86) = 0.00$, $p >= 0.05$). The distribution of high amplitude body bends throughout the startle response differs markedly between *slc1a2b*$^{tk57/tk57}$ and wild-type embryos. *slc1a2b*$^{tk57/tk57}$ embryos perform extended stretches of high amplitude body bends throughout the responses (Figs. 8E and 9A). Of all embryos that

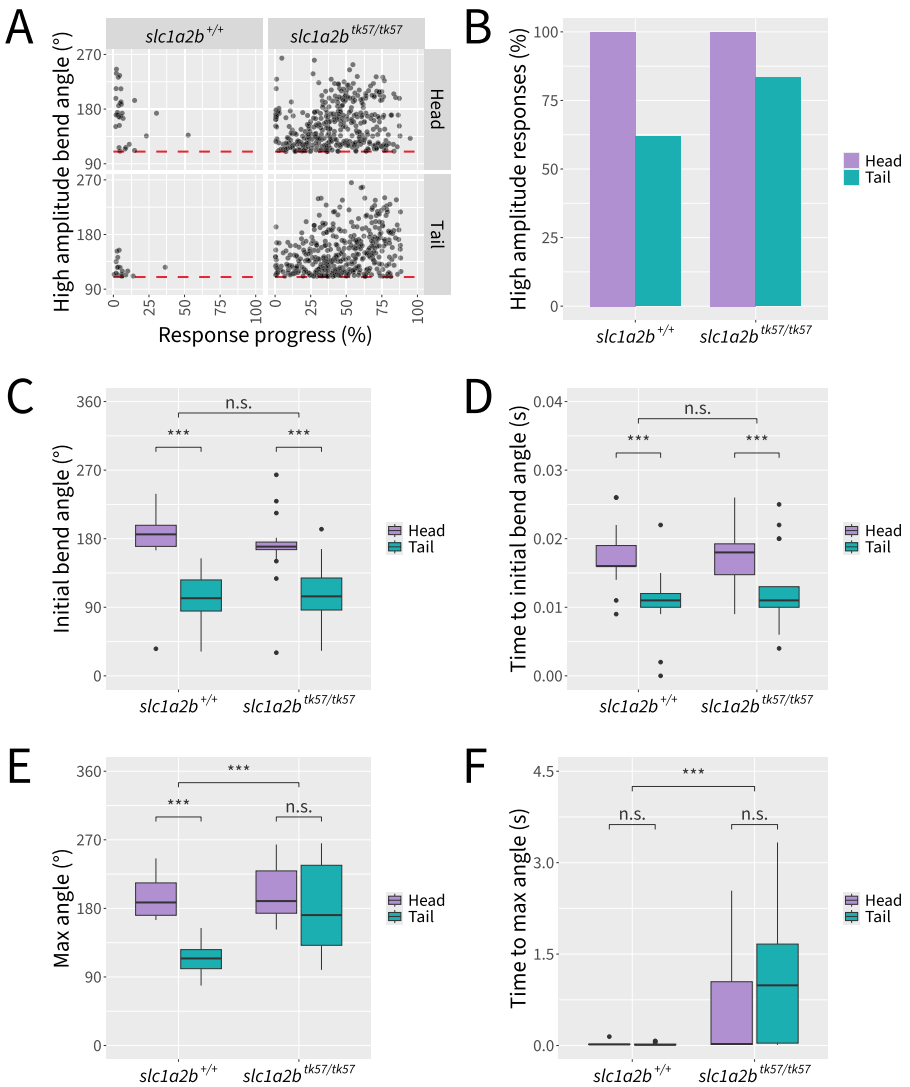

Fig. 9 *slc1a2b*$^{tk57/tk57}$ and wild-type embryos perform rostral–caudal body bends dependent on interactions between genotype and touch location. **A** Visualization of absolute rostral–caudal angle magnitudes at the peaks of high amplitude body bends, depicted over time according to overall response progress. Three fish were omitted at random in order to represent equal numbers of fish across conditions. Red line indicates the threshold of 110 ° for classification as a high amplitude body bend. **B** Percentage of responses which contain one or more high amplitude body bends. **C** Quantification of peak angle of first body bend. **D** Quantification of time to peak angle of first body bend. **E** Quantification of maximum angle across the entire response. **F** Quantification of time to maximum angle across the entire response. Statistical significance determined by two-way ANOVA with Tukey post hoc; n.s.: not significant; ***: $p < 0.001$; $n = 21–24$ fish per condition across 3 independent experiments

perform a high amplitude body bend, the distribution of *slc1a2b*$^{tk57/tk57}$ high amplitude body bends features more high amplitude body bends occurring later in the response (Fig. 9A–B).

We further investigated characteristics of the first body bend within each startle response. Regardless of genotype, the initial bend of tail-stimulated embryos is significantly lower in amplitude than that of head-stimulated embryos, often not crossing the 110 ° threshold (two-way ANOVA, $F(1,86) = 79.703$, $p < 0.001$) (Fig. 9C). Tail-stimulated

embryos also reached their maximum angle significantly earlier in the response (two-way ANOVA, $F(1,86) = 44.833$, $p < 0.001$) (Fig. 9D). Interestingly, we note genotype- and touch location-dependent effects on the maximum body angle of the startle response. Among wild-type embryos, the maximum body bend had a significantly lower amplitude in tail-stimulated embryos than in head-stimulated embryos (two-way ANOVA, $F(1,86) = 36.78$, $p < 0.001$). In contrast, the maximum angle for tail-stimulated *slc1a2b*$^{tk57/tk57}$ responses is nearly indistinguishable from head-stimulated *slc1a2b*$^{tk57/tk57}$ responses (two-way ANOVA interaction, $F(1,86) = 14.45$, $p < 0.001$) (Fig. 9E). To determine whether embryos execute the maximum angle after the initial bend, we measured the time required for the embryo to reach the maximum body angle of the response. *slc1a2b*$^{tk57/tk57}$ embryos perform their maximum body bend significantly later in the response than wild-type embryos (two-way ANOVA, $F(1,86) = 25.26$, $p < 0.001$) (Fig. 9F).

Effects of developmental stage and feeding on larval visuomotor response behavior

To demonstrate additional capabilities of Marigold, particularly its ability to accurately and efficiently analyze the behavior of larval zebrafish in multiwell plates, we examined the effects of developmental stage and feeding on the larval visuomotor response [12–14, 45, 65]. Larvae were fed or not fed once daily from 5 dpf through 7 dpf and behavior was recorded in 24-well plates 4–6 h after feeding at 5 dpf and 7 dpf. To examine the effects of age and feeding across multiple behavioral modes, we recorded fish during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus (Fig. 10A).

Visualization of fish trajectories during the four recording periods suggested reduced swimming in unfed fish, particularly at 7 dpf (Fig. 10B). Smoothed mean speed traces provided further insight into this pattern (Fig. 10C). During dark adaptation, both fed and unfed fish maintained stable activity levels, however this baseline activity level was lower in unfed fish, with the effect becoming more pronounced at 7 dpf. Interestingly, the baseline activity level was essentially identical across fed fish regardless of age. Immediately following the light stimulus, both fed and unfed fish mobilized a rapid burst of activity before returning to the baseline activity level. Fed and unfed fish exhibited similar levels of activity during light adaptation and dark adaptation, with unfed fish again showing a lower level of activity which was more pronounced at 7 dpf. Strikingly, whereas the acute reaction to the dark stimulus was followed by a return to baseline in fed fish (albeit more gradually than was observed following the light stimulus), unfed fish maintained a heightened level of activity which resembled the baseline activity level of fed fish.

Analysis of maximum speed levels during the 5 s immediately following light or dark stimuli revealed no significant differences or interactions between age and feeding status for light stimuli (two-way ANOVA, $F(1,174) = 3.537$ for age, 1.608 for diet, and 2.401 for interaction, $p >= 0.05$ in all three cases). In contrast, immediately following dark stimuli unfed fish exhibited an attenuated response at 7 dpf, but not at 5 dpf (Figs. 10C and 11A), reflecting a main effect of diet (two-way ANOVA, $F(1,174) = 21.659$, $p < 0.001$) and an interaction between age and diet (two-way ANOVA, $F(1,174) = 29.881$, $p < 0.001$), but no main effect of age (two-way ANOVA, $F(1,174) = 0.107$, $p >= 0.05$). Further insights were gleaned by examining the number of swim bouts, which we define as beginning when a fish's speed exceeds 2 mm/s and ending when the fish's speed drops below this threshold.

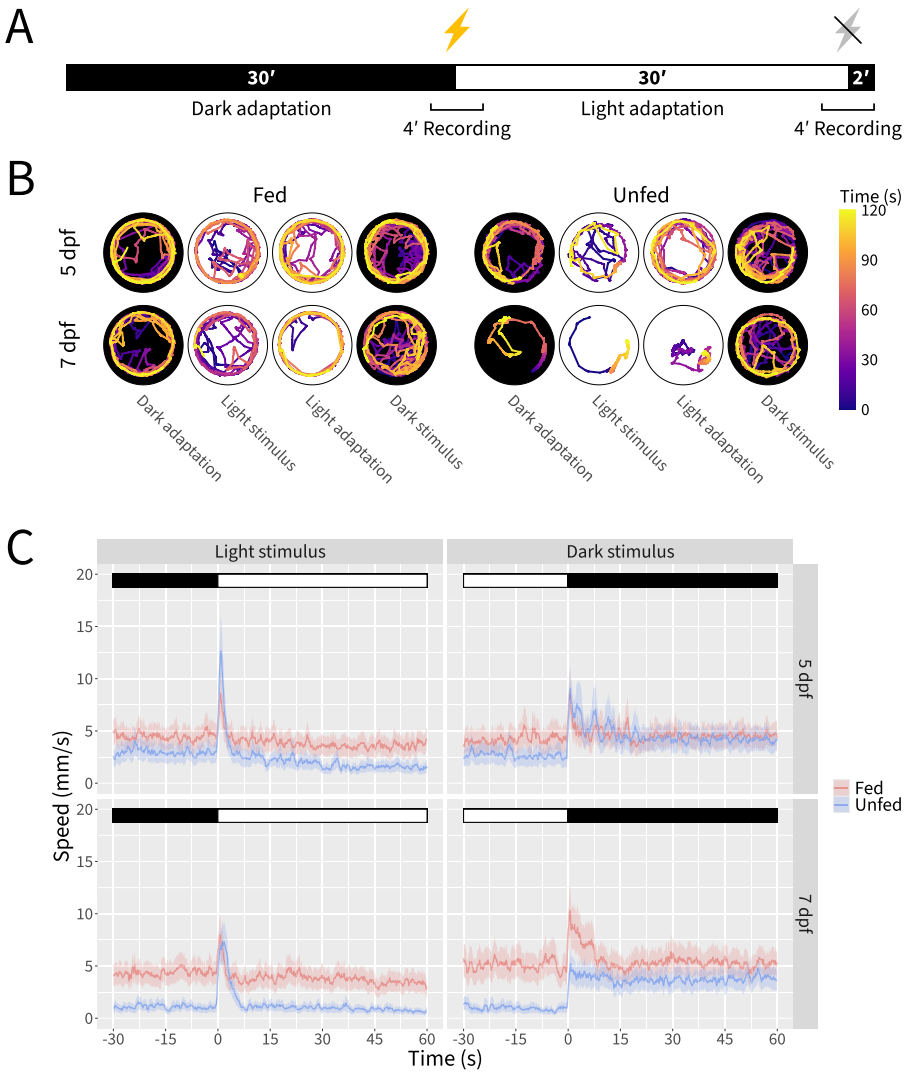

Fig. 10 Visuomotor response trajectory traces and speed plots illustrate reduced swimming in unfed larvae. **A** Schematic of visuomotor response recording paradigm. **B** Representative trajectory traces for fed and unfed fish at 5 and 7 dpf during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus. **C** Speed plots for fed and unfed fish at 5 and 7 dpf during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus. Traces represent the mean plus and minus the standard error of the mean after applying a mean filter with a sliding window corresponding to 1 s. Data represents n = 44–46 fish per condition across 3 independent experiments

Here, a main effect of diet was consistently observed during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus (two-way ANOVA, $F(1,178) = 144.94$, 100.513, 48.606, and 27.127, respectively, $p < 0.001$ during all four periods). However, a significant main effect of age was observed during only the first three of these periods (Fig. 11B) (two-way ANOVA, $F(1,178) = 23.61$, 11.855, 9.129, and 3.535, respectively, $p < 0.001$, $p < 0.001$, $p < 0.01$, and $p >= 0.05$, respectively). The interaction between age and diet followed a similar pattern (Fig. 11B) (two-way ANOVA, $F(1,178) = 26.08$, 9.942, 8.146, and 0.504, respectively, $p < 0.001$, $p < 0.01$, $p < 0.01$, and $p >= 0.05$, respectively). In addition to the number of swim bouts, we also qualitatively examined the vigor of swim bouts by visualizing the durations and peak speeds of swim bouts for each fish

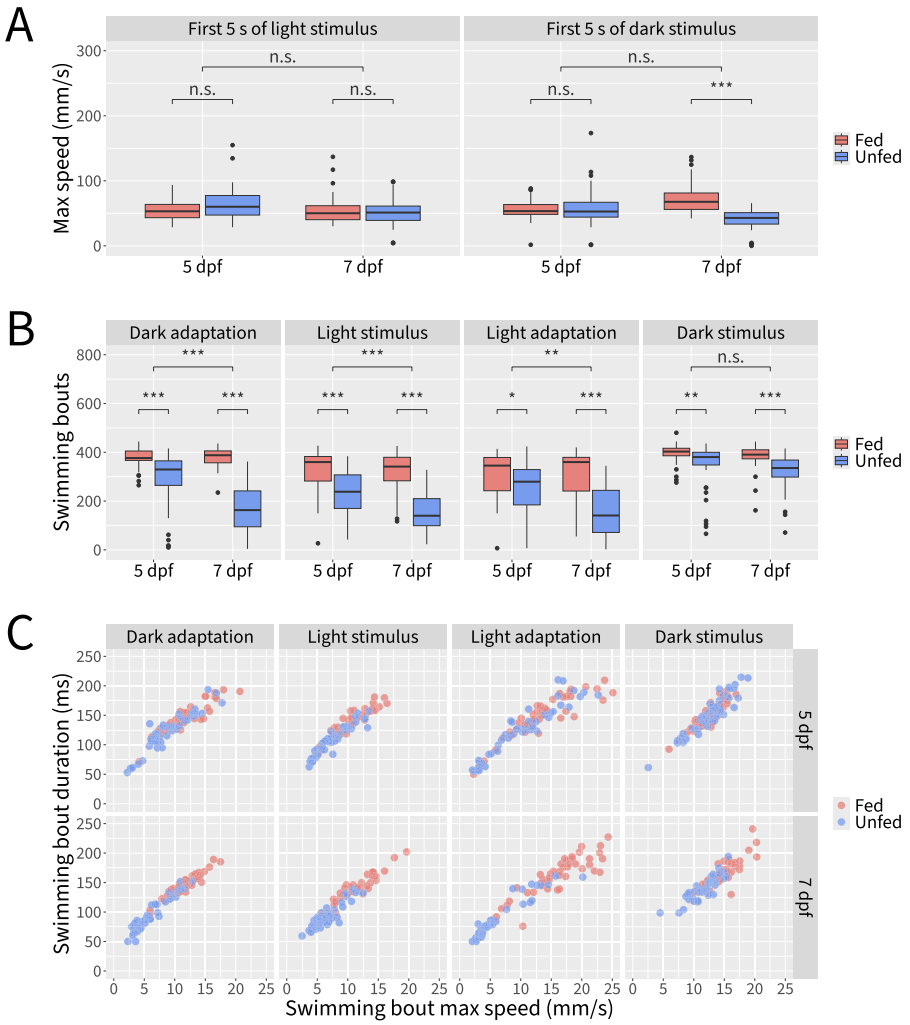

Fig. 11 Detailed behavioral profiling reveals effects and interactions of age and diet on larval visuomotor responses. **A** Quantification of maximum speed during the first 5 s of the response to light and dark stimuli for fed and unfed fish at 5 and 7 dpf. **B** Quantification of swimming bouts for fed and unfed fish at 5 and 7 dpf during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus. **C** Visualization of mean swimming bout duration and mean swimming bout maximum speed for fed and unfed fish at 5 and 7 dpf during periods of dark adaptation, light stimulus, light adaptation, and dark stimulus. Statistical significance determined by two-way ANOVA with Tukey post hoc; n.s.: not significant; *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$; n = 44–46 fish per condition across 3 independent experiments

(Fig. 11C). Whereas the swim bouts of fed and unfed fish generally align at 5 dpf, by 7 dpf unfed fish exhibit less vigorous swimming behavior compared to fed fish as evidenced by their shorter and lower-speed swim bouts.

Discussion

Biological experiments

To demonstrate the utility of Marigold, we analyzed touch-evoked responses in *slc1a2b*$^{tk57/tk57}$ embryos and their wild type siblings, with the stimulus being applied to either the head or tail. We first replicated the known phenotypic differences in the touch-evoked response between *slc1a2b*$^{tk57/tk57}$ embryos and their wild type siblings

when applying the stimulus to the head, including increases in response duration and the number of high amplitude body bends in *slc1a2b*$^{tk57/tk57}$ embryos. We then explored whether applying a stimulus to either the head or tail would further reveal distinct kinematic profiles in *slc1a2b*$^{tk57/tk57}$ embryos. Within tail-stimulated responses, fewer embryos demonstrate an initial bend that passes the high amplitude threshold compared to head-stimulated embryos, regardless of genotype. Because the embryo is already oriented away from the perceived threat when touched on the tail, reorientation via a high amplitude body bend may be less advantageous in this scenario.

We further show that while there are stimulus location differences in the maximum rostral–caudal angle that a wild-type embryo performs, those differences are lost in *slc1a2b*$^{tk57/tk57}$ embryos. Both head- and tail-stimulated *slc1a2b*$^{tk57/tk57}$ embryos reach the maximum rostral–caudal angle later in the response than wild-type embryos, which could help explain why no touch location differences were found for the maximum angle of *slc1a2b*$^{tk57/tk57}$ responses. Although *slc1a2b* is not expressed in the Mauthner cell, *slc1a2b*-expressing glial cells surround the Mauthner array at 48 hpf [37, 57]. Based on the data presented here, we propose that the disrupted kinematic profile of *slc1a2b*$^{tk57/tk57}$ embryos may be due to a reactivation of escape circuitry late into the response.

Interestingly, *slc1a2b* mutants harboring nonsense mutations are able to swim spontaneously at 5 dpf [37], whereas the *slc1a2b*$^{tk57/tk57}$ allele described here and in [57] is fully paralyzed by 96 hpf. This difference could be explained by the nature of the mutations; premature termination codons generated using CRISPR-Cas9 have been shown to trigger genetic compensation via transcriptional adaptation [22, 83]. Additionally, the *slc1a2b*$^{tk57/tk57}$ allele described here is maintained on a TL background, while [37] report using WIK and Tübingen wild-type strains. As notable differences have been reported between wild-type zebrafish strains [4, 56], this could also explain the difference in phenotypic severity. Lastly, [37] describe a mutation in exon 3 of *slc1a2b* while the mutation underlying *slc1a2b*$^{tk57/tk57}$ is located in exon 7 of *slc1a2b* [57].

We also used Marigold to examine visuomotor responses in wild-type larvae at two developmental stages and in fed and unfed states. Larval responses to the visuomotor response differ across developmental time [19, 47]. The first startle responses to visual stimulation can be observed around 3 dpf [21]. By 5 dpf, larvae inflate their swim bladders, exhibit robust swimming behavior, and actively hunt for food. Larvae utilize nutrients from their yolks until at least 7 dpf, and it has been shown that feeding can be delayed until 8 dpf without negatively affecting juvenile survival or growth [35, 50, 73]. A previous study examining the effects of feeding on larval swim behavior, visual stimuli avoidance, and inter-fish distance reported significant behavioral differences between fed and unfed fish at 6 and 7 dpf, but no significant effects at 5 dpf [16]. This observation led to the recommendation that 5 dpf generally be preferred for behavioral experiments. However, the effect of feeding has not previously been evaluated with respect to visuomotor responses.

While our data confirm that feeding status has a more pronounced effect on visuomotor response behavior at 7 dpf than at 5 dpf [16], we also provide evidence for significant effects of feeding on behavioral parameters even at 5 dpf. Thus, our results highlight the importance of reporting and controlling for developmental stage and feeding status in visuomotor response assays.

In both sets of experiments, the in-depth behavioral profiling facilitated by Marigold led to novel findings, despite the *slc1a2b*$^{tk57/tk57}$ mutant having been previously characterized and visuomotor responses being a widely adopted experimental paradigm.

Neural network experiments

To minimize the processing and memory demands of in-browser, on-device neural networks and thereby make them feasible for our zebrafish pose tracking web app, we introduced new neural network variants originating from a series of macro-architectural and micro-architectural design decisions.

Our switch from a hierarchical architecture to an isotropic architecture led to considerably faster training and inference speeds and a reduced memory footprint. Isotropic (also sometimes called isometric) architectures are relatively unexplored. [71] introduced a form of isotropic neural network and noted its memory efficiency and other intriguing properties, although they ultimately found that it did not perform as well for image classification as more traditional hierarchical architectures. More recently, the Vision Transformer family of architectures, which draw inspiration from transformer-based architectures for natural language processing, used patch embedding and isotropic design as a way to reduce the computational burden of self-attention used in transformers [20, 90]. Some works have been successful in modifying Vision Transformer-like architectures to more closely resemble the hierarchical architectures traditionally used for computer vision [48, 100]. Meanwhile, other works have explored whether the patch-based representation used by isotropic architectures may be useful in and of itself, perhaps underlying much of the success of Vision Transformers [23, 86, 88].

Our switch from Batch Normalization to Instance Normalization led to improved training dynamics and allowed us to use gradient accumulation to reduce memory requirements. Although there has been a strong preference for Batch Normalization in neural networks for computer vision, alternatives such as Layer, Group, and Instance Normalization have increasingly found niches in which they are effective. [42], for example, find Instance Normalization useful in U-Net-based architectures for semantic segmentation of biomedical images. Also using U-Net for biomedical image segmentation, [60] use neural architecture search to explore the performance of different normalization methods on a block-by-block basis, finding that Instance Normalization is preferred at most, but, interestingly, not all, positions. The ConvNeXt family of neural networks, which draws inspiration from Vision Transformer-like architectures where Layer Normalization is predominant, found a benefit to switching from Batch Normalization to Layer Normalization, but did not test other normalization methods [48, 96]. Another line of work, which we did not pursue, seeks to eliminate normalization layers altogether (e.g., [11, 24, 101, 102]).

We also achieved greater training and inference speeds by removing some of the normalization and activation layers present in the MobileNetV3 block. That the choice of which normalization and activation layers to retain led to different training outcomes emphasizes the importance of being strategic in the adoption of this approach. [48] also find some benefit to using fewer normalization and activation layers, although they use a standard ResNet bottleneck as their starting point and arrive at a somewhat different final configuration of layers.

As the state of the art in machine learning continues to demand ever-increasing computational resources to power deep neural network training and inference for computer vision and natural language processing applications, the associated carbon footprint exerts an increasingly negative impact on the environment [74, 82, 91, 99]. Thus, we feel the machine learning community has a moral responsibility to design more efficient neural networks. Our design choices highlight ways in which architectural optimizations can help to minimize the processing and memory demands of neural networks while leading to faster training convergence and inference speed. It is perhaps worth noting that we achieved our goal of reducing the processing and memory demands of neural networks for animal pose estimation primarily through architectural improvements, but a number of promising alternative strategies could have been explored instead or in addition, such as knowledge distillation, quantization, and pruning [18].

Comparison to existing free and open source software

Numerous free and open source software solutions are available for zebrafish behavioral analysis, encompassing a wide range of tracking capabilities, implementation details, hardware requirements, expectations for programming experience, and degrees of specificity to zebrafish. While a comprehensive review of all such software is beyond the scope of this work, we summarize a number of representative tools below.

Among animal-agnostic solutions, several recently introduced tools have sought to make advances in machine learning-based pose estimation techniques available to researchers interested in analyzing animal behavior [27, 30, 55, 62]. These tools have tremendously improved the precision with which animal pose tracking can be performed, but are not without limitations. Because these tools commonly use off-the-shelf neural network architectures which were originally designed for other datasets or tasks, the resulting models are frequently overparameterized and their training can be inefficient, often requiring the use of expensive GPUs. Additionally, such software can be challenging to install due to complex dependencies on language runtimes, libraries, and device drivers, as well as difficult to use due to workflows requiring integration of GUI-, command line-, and cloud-based elements. Additionally, most of these tools lack support for tracking animals in multiwell plates, a popular experimental paradigm for high-throughput behavioral studies using larval zebrafish.

Many zebrafish-specific or small animal-specific tools have also been introduced. Relatively few of these use machine learning, with most instead relying on traditional computer vision techniques such as background subtraction, centroid tracking, and following local pixel intensity cues to identify different parts of the fish [3, 14–16, 43, 54, 59]. These approaches can be adequate, but are often brittle, requiring carefully controlled experimental setups to deliver reliable results and breaking down under adverse conditions such as the introduction of foreign objects into the field of view. Additionally, some of these tools are limited to only tracking a single keypoint. Those tools that incorporate machine learning-based tracking typically do so by drawing on one of the more general purpose tools described above, such as DeepLabCut, to provide the required functionality, and thus can inherit many of the same limitations as these tools such as a complex installation process and the requirement for a powerful GPU [28, 32, 81, 84, 103]. A number of tools combine traditional or machine learning-based tracking abilities

with camera and other hardware integration, including affordable hardware setups such as those based on Raspberry Pi, to drive data collection and stimulus delivery for experiments [32, 43, 76, 81, 84].

Marigold represents a notable departure from existing software primarily in its relative lack of hardware requirements, elimination of installation procedures, and focus on making robust, machine learning-based pose tracking more widely accessible to both students and researchers. To our knowledge, Marigold is the first free and open source software to perform machine learning-based animal pose tracking for behavioral analysis entirely in-browser and on-device, an approach which allows us to meaningfully lower the financial and technical barrier for performing this type of analysis. By implementing our software as a web app, we are able to take advantage of the web platform to deliver a rich user interface supporting a highly streamlined workflow, all without requiring any installation. The web app implementation also allows Marigold to run entirely within the security sandbox provided by the user's web browser. Moreover, by designing a neural network architecture from the ground up to perform highly efficient animal pose estimation and implementing this architecture using the recently introduced WebAssembly technology [33, 63], we are able to achieve reasonable training and inference speeds even on modestly powered computers lacking a dedicated GPU and despite the limited access to memory and computational resources web apps are permitted.

Future directions

Marigold's ability to allow the user to create their own dataset and train custom models, while maintaining a focus on functionality that is of particular interest to the zebrafish community, makes the program applicable to a broad range of zebrafish behavioral experiments. For instance, in addition to the applications demonstrated here (i.e., analysis of touch-evoked and visuomotor responses), we also have successfully used Marigold to analyze spinalized larvae, head-embedded larvae, and adult fish (data not shown). However, Marigold is currently limited to analyzing a single animal in each region of interest (ROI), a decision made to maintain the program's simplicity and our focus on embryonic and early larval stages of zebrafish development, when social interaction is minimal [79]. Future work could extend Marigold to support the analysis of multiple fish in each ROI and for three-dimensional pose tracking, since both social interaction and three-dimensional swimming become more prominent during late larval and juvenile development [52, 79]. Additionally, while we have successfully used Marigold for pose tracking of other animal species such as mice (data not shown), there is clearly an opportunity to more formally extend the web app-based approach of Marigold for pose tracking in other species and to adapt it for other types of behavioral analysis.

Finally, our WebAssembly-based implementation was able to obtain reasonable performance on CPU, making our web app compatible with a wide range of platforms and devices and making it particularly valuable in low-resource settings. However, for devices equipped with a discrete GPU Marigold may leave performance on the table, especially when it comes to larger neural networks such as those we trained for our touch-evoked response dataset. The emerging WebGPU standard [93] could provide a means for Marigold to take advantage of advanced GPUs when they are available, helping to close this performance gap. In the meantime, we note that Marigold may not be

suited for all types of behavioral recordings. Marigold is quite efficient for high-throughput, low-resolution tracking with few keypoints, as exemplified by the models trained for analyzing visuomotor responses. It is also fast enough for high-resolution tracking with many keypoints when the recordings are relatively short, as exemplified by the models trained for analyzing touch-evoked responses, but may not be fast enough to make such high-resolution, many-keypoint tracking feasible for extended recordings.

Conclusions

Much of the existing software tools used for zebrafish behavioral analysis have constraints that limit their utility, including lacking support for efficient tracking in multiwell plates, being cost prohibitive or necessitating cost prohibitive hardware, or requiring substantial programming experience. Here, we describe Marigold, which aims to address these limitations. It is a free and open source web app that utilizes machine learning to perform zebrafish pose tracking. Marigold features efficient neural networks that allow for reasonable training and inference speeds even on basic laptop computers. We demonstrate the utility of Marigold by using it to uncover novel aspects of the *tnt* mutant touch-evoked escape response and of the effects of developmental stage and feeding on the larval visuomotor response. We expect that Marigold will serve as a user-friendly tool to aid the zebrafish community in conducting robust, high-throughput behavioral analysis.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-025-06042-2.

Additional file 1.

Acknowledgements
We thank Hanna E. Dorman Barclay and Caroline Martin for excellent fish care. We thank all members of the Downes Lab, as well as the University of Massachusetts Amherst and Amherst College zebrafish communities, for thoughtful discussion. We are extremely grateful to Scott Alfeld for helpful discussion and feedback on an early version of the manuscript.

Author contributions
GT, JGT, and GBD acquired funding for the project. GT, RMR, WB, JGT, and GBD conceptualized the software and the biological experiments. GT conceptualized and conducted the computational experiments and developed the software. GT and GM collected the training data and conducted the biological experiments. RMR and BEC labeled the training data. GT, RMR, and WB analyzed the biological experiments and prepared the figures. GT, RMR, WB, BEC, GBD, and JGT wrote the manuscript. All authors read and approved the final manuscript.

Funding
This work was funded by the National Science Foundation (IOS 1456866) to GBD and JGT, the American Epilepsy Society (AES2017SD) to GBD, anonymous gifts to GBD, and the University of Massachusetts Amherst Biology Department (2019 Distinguished Graduate Student Summer Research Fellowship) to GT.

Availability of data and materials
The datasets used and/or analyzed during this study are available from the corresponding authors upon reasonable request. The source code for Marigold is licensed under the GNU General Public License version 3 (GPLv3) and is available at: https://github.com/downeslab/marigold. The Marigold web app itself is available at: https://downeslab.github.io/marigold/.

Declarations

Ethics approval and consent to participate
All animal procedures were approved by the University of Massachusetts Amherst Institutional Animal Care and Use Committee (IACUC) under Animal Welfare Assurance Number A3551-01. No animal procedures were performed at Amherst College.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 9 May 2024 Accepted: 7 January 2025
Published online: 28 January 2025

References

1. Adobe. (n.d.). Adobe Source Sans 3. Retrieved 3 Dec 2024, from https://github.com/adobe-fonts/source-sans.
2. Ansel J, Yang E, He H, Gimelshein N, Jain A, Voznesensky M, Bao B, Bell P, Berard D, Burovski E, Chauhan G, Chourdia A, Constable W, Desmaison A, DeVito Z, Ellison E, Feng W, Gong J, Gschwind M, Chintala S (2024). PyTorch 2: faster machine learning through dynamic Python byte-code transformation and graph compilation. In: Proceedings of the 29th ACM international conference on architectural support for programming languages and operating systems, vol. 2, pp. 929–47. https://doi.org/10.1145/3620665.3640366.
3. Audira G, Sampurna BP, Juniardi S, Liang S-T, Lai Y-H, Hsiao C-D. A versatile setup for measuring multiple behavior endpoints in zebrafish. Inventions. 2018;3(4):75. https://doi.org/10.3390/inventions3040075.
4. Audira G, Siregar P, Strungaru S-A, Huang J-C, Hsiao C-D. Which zebrafish strains are more suitable to perform behavioral studies? A comprehensive comparison by phenomic approach [number: 8]. Biology. 2020;9(8):200. https://doi.org/10.3390/biology9080200.
5. Ba JL, Kiros JR, Hinton GE. Layer normalization [arXiv: 1607.06450 [cs, stat]]. 2016. https://doi.org/10.48550/arXiv.1607.06450.
6. Baraban SC, Dinday MT, Hortopan GA. Drug screening in *Scn1a* zebrafish mutant identifies clemizole as a potential Dravet syndrome treatment. Nat Commun. 2013;4(1):2410. https://doi.org/10.1038/ncomms3410.
7. Berg EM, Björnfors ER, Pallucchi I, Picton LD, El Manira A. Principles governing locomotion in vertebrates: lessons from zebrafish. Front Neural Circuits. 2018;12:73. https://doi.org/10.3389/fncir.2018.00073.
8. Blackman D, Vigna S. Scrambled linear pseudorandom number generators. ACM Trans Math Softw. 2021;36(1–36):32. https://doi.org/10.1145/3460772.
9. Blackman D, Vigna S (n.d.). xoshiro/xoroshiro generators and the PRNG shootout. Retrieved 3 Dec 2024, from https://prng.di.unimi.it/
10. Breuer M, Guglielmi L, Zielonka M, Hemberger V, Kölker S, Okun JG, Hoffmann GF, Carl M, Sauer SW, Opladen T. QDPR homologues in *Danio rerio* regulate melanin synthesis, early gliogenesis, and glutamine homeostasis. PLoS ONE. 2019;14(4): e0215162. https://doi.org/10.1371/journal.pone.0215162.
11. Brock A, De S, Smith SL, Simonyan K. High-performance large-scale image recognition without normalization [arXiv:2102.06171 [cs] version: 1] (2021, February). https://doi.org/10.48550/arXiv.2102.06171
12. Brockerhoff SE, Hurley JB, Niemi GA, Dowling JE. A new form of inherited red-blindness identified in zebrafish. J Neurosci. 1997;17(11):4236–42. https://doi.org/10.1523/JNEUROSCI.17-11-04236.1997.
13. Burgess HA, Granato M. Modulation of locomotor activity in larval zebrafish during light adaptation. J Exp Biol. 2007;210(Pt 14):2526–39. https://doi.org/10.1242/jeb.003939.
14. Burgess HA, Granato M. Sensorimotor gating in larval zebrafish. J Neurosci Off J Soc Neurosci. 2007;27(18):4984–94. https://doi.org/10.1523/JNEUROSCI.0615-07.2007.
15. Chen S, Cheng G, Xin J, Xin Y, Dong R, Wang S, Sík A, Han L, Wang X. Establishment of innovative ZebVortrack behavioral analysis system for quantitative epileptic seizure assessment in larval zebrafish. 2023. https://doi.org/10.21203/rs.3.rs-2428365/v1.
16. Clift D, Richendrfer H, Thorn RJ, Colwill RM, Creton R. High-throughput analysis of behavior in zebrafish larvae: effects of feeding. Zebrafish. 2014;11(5):455–61. https://doi.org/10.1089/zeb.2014.0989.
17. d'Amora M, Giordani S. The utility of zebrafish as a model for screening developmental neurotoxicity. Front Neurosci. 2018;12:976. https://doi.org/10.3389/fnins.2018.00976.
18. Dantas PV, Sabino da Silva W, Cordeiro LC, Carvalho CB. A comprehensive review of model compression techniques in machine learning. Appl Intell. 2024;54(22):11804–44. https://doi.org/10.1007/s10489-024-05747-w.
19. de Esch C, van der Linde H, Slieker R, Willemsen R, Wolterbeek A, Woutersen R, De Groot D. Locomotor activity assay in zebrafish larvae: influence of age, strain and ethanol. Neurotoxicol Teratol. 2012;34(4):425–33. https://doi.org/10.1016/j.ntt.2012.03.002.
20. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N . An image is worth 16x16 words: transformers for image recognition at scale [arXiv:2010.11929 [cs]] (2021, June). https://doi.org/10.48550/arXiv.2010.11929
21. Easter SS, Nicola GN. The development of vision in the zebrafish (*Danio rerio*). Dev Biol. 1996;180(2):646–63. https://doi.org/10.1006/dbio.1996.0335.
22. El-Brolosy MA, Kontarakis Z, Rossi A, Kuenne C, Günther S, Fukuda N, Kikhi K, Boezio GL, Takacs C, Lai S-L, Fukuda R, Gerri C, Giraldez AJ, Stainier DY. Genetic compensation triggered by mutant mRNA degradation. Nature. 2019;568(7751):193–7. https://doi.org/10.1038/s41586-019-1064-z.
23. Feng W, Zhang X, Song Q, Sun G. The incoherence of deep isotropic neural networks increases their performance in image classification. Electronics. 2022;11(21):3603. https://doi.org/10.3390/electronics11213603.
24. Gadhikar A, Burkholz R. Dynamical isometry for residual networks [arXiv:2210.02411[cs]] (2022, October). https://doi.org/10.48550/arXiv.2210.02411
25. Gesemann M, Lesslauer A, Maurer CM, Schönthaler HB, Neuhauss SC. Phylogenetic analysis of the vertebrate excitatory/neutral amino acid transporter (SLC1/EAAT) family reveals lineage specific subfamilies. BMC Evol Biol. 2010;10(1):117. https://doi.org/10.1186/1471-2148-10-117.

26. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 249–256. Retrieved 3 Dec 2024, from https://proceedings.mlr.press/v9/glorot10a.html
27. Goodwin NL, Choong JJ, Hwang S, Pitts K, Bloom L, Islam A, Zhang YY, Szelenyi ER, Tong X, Newman EL, Miczek K, Wright HR, McLaughlin RJ, Norville ZC, Eshel N, Heshmati M, Nilsson SRO, Golden SA. Simple Behavioral Analysis (SimBA) as a platform for explainable machine learning in behavioral neuroscience. Nat Neurosci. 2024;27(7):1411–24. https://doi.org/10.1038/s41593-024-01649-9.
28. Gore SV, Kakodkar R, Del Rosario Hernández T, Edmister ST, Creton R. Zebrafish Larvae Position Tracker (Z-LaP Tracker): a high-throughput deep-learning behavioral approach for the identifica- tion of calcineurin pathway-modulating drugs using zebrafish larvae [number: 1]. Sci Rep. 2023;13(1):3174. https://doi.org/10.1038/s41598-023-30303-w.
29. Granato M, van Eeden F, Schach U, Trowe T, Brand M, Furutani-Seiki M, Haffter P, Hammerschmidt M, Heisenberg C, Jiang Y, Kane D, Kelsh R, Mullins M, Odenthal J, Nusslein-Volhard C. Genes controlling and mediating locomotion behavior of the zebrafish embryo and larva. Development. 1996;123(1):399–413. https://doi.org/10.1242/dev.123.1.399.
30. Graving JM, Chae D, Naik H, Li L, Koger B, Costelloe BR, Couzin ID. DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. eLife. 2019;8:e47994. https://doi.org/10.7554/eLife.47994
31. Greener JG, Kandathil SM, Moffat L, Jones DT. A guide to machine learning for biologists. Nat Rev Mol Cell Biol. 2022;23(1):40–55. https://doi.org/10.1038/s41580-021-00407-0.
32. Guilbeault NC, Guerguiev J, Martin M, Tate I, Thiele TR. BonZeb: open-source, modular software tools for high-resolution zebrafish tracking and analysis. Sci Rep. 2021;11(1):8148. https://doi.org/10.1038/s41598-021-85896-x.
33. Haas A, Rossberg A, Schuff DL, Titzer BL, Holman M, Gohman D, Wagner L, Zakai A, Bastien J. Bringing the web up to speed with WebAssembly. In: Proceedings of the 38th ACM SIGPLAN conference on programming language design and implementation; 2017 185–200. https://doi.org/10.1145/3062341.3062363
34. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Oliphant TE. Array programming with NumPy [publisher: nature publishing group]. Nature. 2020;585(7825):357–62. https://doi.org/10.1038/s41586-020-2649-2.
35. Hernandez RE, Galitan L, Cameron J, Goodwin N, Ramakrishnan L. Delay of initial feeding of zebrafish larvae until 8 days postfertilization has no impact on survival or growth through the juvenile stage. Zebrafish. 2018;15(5):515–8. https://doi.org/10.1089/zeb.2018.1579.
36. Holtzman NG, Iovine MK, Liang JO, Morris J. Learning to fish with genetics: a primer on the vertebrate model *Danio rerio*. Genetics. 2016;203(3):1069–89. https://doi.org/10.1534/genetics.116.190843.
37. Hotz AL, Jamali A, Rieser NN, Niklaus S, Aydin E, Myren-Svelstad S, Lalla L, Jurisch-Yaksi N, Yaksi E, Neuhauss SCF. Loss of glutamate transporter *eaat2a* leads to aberrant neuronal excitability, recurrent epileptic seizures, and basal hypoactivity. Glia. 2022;70(1):196–214. https://doi.org/10.1002/glia.2410.
38. Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V, Le QV, Adam H. Searching for MobileNetV3 [arXiv:1905.02244 [cs]] (2019, November). https://doi.org/10.48550/arXiv.1905.02244
39. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: efficient convolutional neural networks for mobile vision applications. 2017. https://doi.org/10.48550/arXiv.1704.04861.
40. Howe K, Clark MD, Torroja CF, Torrance J, Berthelot C, Muffato M, Collins JE, Humphray S, McLaren K, Matthews L, McLaren S, Sealy I, Caccamo M, Churcher C, Scott C, Barrett JC, Koch R, Rauch G-J, White S, Stemple DL. The zebrafish reference genome sequence and its relationship to the human genome. Nature. 2013;496(7446):498–503. https://doi.org/10.1038/nature12111.
41. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift [Issue: arXiv:1502.03167 arXiv: 1502.03167 [cs]] (2015, March). https://doi.org/10.48550/arXiv.1502.03167
42. Isensee F, Jaeger PF, Kohl SAA, Petersen J, Maier-Hein KH. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. Nat Method. 2021;18(2):203–11. https://doi.org/10.1038/s41592-020-01008-z.
43. Joo W, Vivian MD, Graham BJ, Soucy ER, Thyme SB. A customizable low-cost system for massively parallel zebrafish behavioral phenotyping. Front Behav Neurosci. 2020;14:606900. https://doi.org/10.3389/fnbeh.2020.606900.
44. Kalueff AV, Gebhardt M, Stewart AM, Cachat JM, Brimmer M, Chawla JS, Craddock C, Kyzar EJ, Roth A, Landsman S, Gaikwad S, Robinson K, Baatrup E, Tierney K, Shamchuk A, Norton W, Miller N, Nicolson T, Braubach O. Zebrafish neuroscience research consortium. Towards a comprehensive catalog of zebrafish behavior 1.0 and beyond. Zebrafish. 2013;10(1):70–86. https://doi.org/10.1089/zeb.2012.0861.
45. Lee HB, Schwab TL, Sigafoos AN, Gauerke JL, Krug RG, Serres MR, Jacobs DC, Cotter RP, Das B, Petersen MO, Daby CL, Urban RM, Berry BC, Clark KJ. Novel zebrafish behavioral assay to identify modifiers of the rapid, nongenomic stress response. Genes Brain Behav. 2019;18(2):e12549. https://doi.org/10.1111/gbb.12549.
46. Liu KS, Fetcho JR. Laser ablations reveal functional relationships of segmental hindbrain neurons in zebrafish. Neuron. 1999;23(2):325–35. https://doi.org/10.1016/S0896-6273(00)80783-7.
47. Liu Y, Carmer R, Zhang G, Venkatraman P, Brown SA, Pang C-P, Zhang M, Ma P, Leung YF. Statistical analysis of zebrafish locomotor response. PLoS ONE. 2015;10(10):e0139521. https://doi.org/10.1371/journal.pone.0139521.
48. Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S. A ConvNet for the 2020s [arXiv:2201.03545 [cs]] (2022, March). https://doi.org/10.48550/arXiv.2201.03545
49. Loshchilov I, Hutter F. Decoupled weight decay regularization [arXiv: 1711.05101 [cs, math]] (2019, January). https://doi.org/10.48550/arXiv.1711.05101
50. Lucore EC, Connaughton VP. Observational learning and irreversible starvation in first-feeding zebrafish larvae: is it okay to copy from your friends? Zoology. 2021;145:125896. https://doi.org/10.1016/j.zool.2021.125896.
51. MacRae CA, Peterson RT. Zebrafish as a mainstream model for in vivo systems pharmacology and toxicology. Annu Rev Pharmacol Toxicol. 2023;63(1):43–64. https://doi.org/10.1146/annurev-pharmtox-051421-105617.

52. Macrì S, Neri D, Ruberto T, Mwaffo V, Butail S, Porfiri M. Three-dimensional scoring of zebrafish behavior unveils biological phenomena hidden by two-dimensional analyses. Sci Rep. 2017;7(1):1962. https://doi.org/10.1038/s41598-017-01990-z.
53. Marion P, Fermanian A, Biau G, Vert J-P. Scaling ResNets in the large-depth regime [arXiv:2206.06929] (2024, June). https://doi.org/10.48550/arXiv.2206.06929
54. Marques JC, Lackner S, Félix R, Orger MB. Structure of the zebrafish locomotor repertoire revealed with unsupervised behavioral clustering. Current biology: CB. 2018;28(2):181.e5–95.e5. https://doi.org/10.1016/j.cub.2017.12.002.
55. Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci. 2018;21(9):1281–9. https://doi.org/10.1038/s41593-018-0209-y.
56. Maximino C, Puty B, Matos Oliveira KR, Herculano AM. Behavioral and neurochemical changes in the zebrafish leopard strain. Genes Brain Behav. 2013;12(5):576–82. https://doi.org/10.1111/gbb.12047.
57. McKeown KA, Moreno R, Hall VL, Ribera AB, Downes GB. Disruption of Eaat2b, a glutamate transporter, results in abnormal motor behaviors in developing zebrafish. Dev Biol. 2012;362(2):162–71. https://doi.org/10.1016/j.ydbio.2011.11.001.
58. McKinney W. Data structures for statistical computing in Python. In: van der Walt S, Millman J, editors. Proceedings of the 9th python in science conference. 2010. pp. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a.
59. Mirat O, Sternberg JR, Severi KE, Wyart C. ZebraZoom: an automated program for high-throughput behavioral analysis and categorization. Front Neural Circuit. 2013;7:107. https://doi.org/10.3389/fncir.2013.00107.
60. Neubig L, Kist AM. Evolutionary normalization optimization boosts semantic segmentation network performance. In: Greenspan H, Madabhushi A, Mousavi P, Salcudean S, Duncan J, Syeda-Mahmood T, Taylor R, editors. Medical image computing and computer assisted intervention – MICCAI 2023. Springer Nature Switzerland. 2023. pp. 703–12. https://doi.org/10.1007/978-3-031-43901-8_67.
61. Newell A, Yang K, Deng J (2016, July). Stacked hourglass networks for human pose estimation [Issue: arXiv:1603.06937 arXiv: 1603.06937 [cs]]. https://doi.org/10.48550/arXiv.1603.06937
62. Pereira TD, Aldarondo DE, Willmore L, Kislin M, Wang SS-H, Murthy M, Shaevitz JW. Fast animal pose estimation using deep neural networks. Nat Methods. 2019;16(1):117–25. https://doi.org/10.1038/s41592-018-0234-5.
63. Perkel JM. No installation required: How WebAssembly is changing scientific computing. Nature. 2024;627(8003):455–6. https://doi.org/10.1038/d41586-024-00725-1.
64. R Core Team. (2024). R: a language and environment for statistical computing. In: R foundation for statistical computing. https://www.R-project.org/
65. Randlett O, Haesemeyer M, Forkin G, Shoenhard H, Schier AF, Engert F, Granato M. Distributed plasticity drives visual habituation learning in larval zebrafish. Curr Biol. 2019;29(8):1337.e4–45.e4. https://doi.org/10.1016/j.cub.2019.02.039.
66. Ravan A, Feng R, Gruebele M, Chemla YR. Rapid automated 3-D pose estimation of larval zebrafish using a physical model-trained neural network. PLoS Comput Biol. 2023;19(10): e1011566. https://doi.org/10.1371/journal.pcbi.1011566.
67. Romero-Ferrero F, Bergomi MG, Hinz RC, Heras FJH, de Polavieja GG. Idtracker.ai: tracking all individuals in small or large collectives of unmarked animals. Nat Methods. 2019;16(2):179–82. https://doi.org/10.1038/s41592-018-0295-5.
68. Ronneberger O, Fischer P, Brox T (2015, May). U-Net: convolutional networks for biomedical image segmentation [Issue: arXiv:1505.04597 arXiv: 1505.04597 [cs]]. https://doi.org/10.48550/arXiv.1505.04597
69. Sakai C, Ijaz S, Hoffman EJ. Zebrafish models of neurodevelopmental disorders: past, present, and future. Front Mol Neurosci. 2018;11:294. https://doi.org/10.3389/fnmol.2018.00294.
70. Saleh A, Jones D, Jerry D, Azghadi MR. MFLD-net: a lightweight deep learning network for fish morphometry using landmark detection. Aquat Ecol. 2023;57(4):913–31. https://doi.org/10.1007/s10452-023-10044-8.
71. Sandler M, Baccash J, Zhmoginov A, Howard A (2019, October). Non-discriminative data or weak model? On the relative importance of data and model resolution [Issue: arXiv:1909.03205 arXiv: 1909.03205 [cs]]. https://doi.org/10.48550/arXiv.1909.03205
72. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2019, March). MobileNetV2: inverted residuals and linear bottlenecks [arXiv:1801.04381 [cs]]. https://doi.org/10.48550/arXiv.1801.04381
73. Schwartz AV, Sant KE, Navarrete J, George UZ. Mathematical modeling of the interaction between yolk utilization and fish growth in zebrafish, *Danio rerio*. Development. 2021;148(9):dev193508. https://doi.org/10.1242/dev.193508.
74. Schwartz R, Dodge J, Smith NA, Etzioni O (2019, August). Green AI [Issue: arXiv:1907.10597 arXiv:1907.10597 [cs, stat]]. https://doi.org/10.48550/arXiv.1907.10597
75. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data. 2019;6(1):60. https://doi.org/10.1186/s40537-019-0197-0.
76. Singh BJ, Zu L, Summers J, Asdjodi S, Glasgow E, Kanwal JS. NemoTrainer: automated conditioning for stimulus-directed navigation and decision making in free-swimming zebrafish. Animals. 2023;13(1):116. https://doi.org/10.3390/ani13010116.
77. Smilkov D, Thorat N, Assogba Y, Yuan A, Kreeger N, Yu P, Zhang K, Cai S, Nielsen E, Soergel D, Bileschi S, Terry M, Nicholson C, Gupta SN, Sirajuddin S, Sculley D, Monga R, Corrado G, Viégas FB, Wattenberg M (2019, February). TensorFlow.js: machine learning for the web and beyond [arXiv:1901.05350 [cs] version: 2]. https://doi.org/10.48550/arXiv.1901.05350
78. Smith NJ, van der Walt S, Firing E (n.d.). magma, inferno, plasma, and viridis colormaps. Retrieved December 3, 2024, from https://github.com/BIDS/colormap
79. Stednitz SJ, Washbourne P. Rapid progressive social development of zebrafish. Zebrafish. 2020;17(1):11–7. https://doi.org/10.1089/zeb.2019.1815.

80. Steele GL, Lea D, Flood CH. Fast splittable pseudorandom number generators. SIGPLAN Not. 2014;49(10):453–72. https://doi.org/10.1145/2714064.2660195.
81. Štih V, Petrucco L, Kist AM, Portugues R. Stytra: an open-source, integrated system for stimulation, tracking and closed-loop behavioral experiments. PLOS Comput Biol. 2019;15(4):e1006699. https://doi.org/10.1371/journal.pcbi.1006699.
82. Strubell E, Ganesh A, McCallum A (2019, June). Energy and policy considerations for deep learning in NLP [Issue: arXiv:1906.02243 arXiv: 1906.02243 [cs]]. https://doi.org/10.48550/arXiv.1906.02243
83. Sztal TE, Stainier DYR. Transcriptional adaptation: a mechanism underlying genetic robustness. Development. 2020;147(15):dev186452. https://doi.org/10.1242/dev.186452.
84. Tadres D, Louis M. PiVR: an affordable and versatile closed-loop platform to study unrestrained sensorimotor behavior. PLoS Biol. 2020;18(7): e3000712. https://doi.org/10.1371/journal.pbio.3000712.
85. Togelius J, Yannakakis GN (2024, February). Choose your weapon: survival strategies for depressed AI academics [arXiv:2304.06035 version: 2]. https://doi.org/10.48550/arXiv.2304.06035
86. Tolstikhin I, Houlsby N, Kolesnikov A, Beyer L, Zhai X, Unterthiner T, Yung J, Steiner A, Keysers D, Uszkoreit J, Lucic M, Dosovitskiy A (2021, June). MLP-mixer: an all-MLP architecture for vision [Issue: arXiv:2105.01601 arXiv: 2105.01601 [cs]]. https://doi.org/10.48550/arXiv.2105.01601
87. TorchVision maintainers and contributors. (2016, November). TorchVision: PyTorch's Computer Vision library. https://github.com/pytorch/vision
88. Trockman A, Kolter JZ (2022, January). Patches are all you need? [arXiv: 2201.09792 [cs]]. https://doi.org/10.48550/arXiv.2201.09792
89. Ulyanov D, Vedaldi A, Lempitsky V (2017, November). Instance normalization: the missing ingredient for fast stylization [arXiv: 1607.08022 [cs]]. https://doi.org/10.48550/arXiv.1607.08022
90. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017, June). Attention is all you need [arXiv:1706.03762 [cs] version: 1]. https://doi.org/10.48550/arXiv.1706.03762
91. Verdecchia R, Sallou J, Cruz L (2023, May). A systematic review of green AI [Issue: arXiv:2301.11047 arXiv: 2301.11047 [cs]]. https://doi.org/10.48550/arXiv.2301.11047
92. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, van Mulbregt P. SciPy 10: fundamental algorithms for scientific computing in Python. Nat Method. 2020;17(3):261–72. https://doi.org/10.1038/s41592-019-0686-2.
93. WebGPU, W3C World Wide Web Consortium Recommendation Draft. 2025. https://www.w3.org/TR/2025/CRD-webgpu-20250116/, Latest version at https://www.w3.org/TR/webgpu/.
94. Web Neural Network API, W3C World Wide Web Consortium Recommendation Draft. 2025. https://www.w3.org/TR/2025/CRD-webnn-20250116/, Latest version at https://www.w3.org/TR/webnn/.
95. Wickham H. ggplot2: elegant graphics for data analysis. New York: Springer-Verlag; 2016. https://ggplot2.tidyverse.org.
96. Woo S, Debnath S, Hu R, Chen X, Liu Z, Kweon IS, Xie S. ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders. 2023. https://doi.org/10.48550/arXiv.2301.00808.
97. Wu Y, He K (2018, June). Group normalization [Issue: arXiv:1803.08494 arXiv: 1803.08494 [cs]]. https://doi.org/10.48550/arXiv.1803.08494
98. Xiao B, Wu H, Wei Y (2018, August). Simple baselines for human pose estimation and tracking [arXiv: 1804.06208 [cs]]. https://doi.org/10.48550/arXiv.1804.06208
99. Xu Y, Martínez-Fernández S, Martinez M, Franch X (2023, February). Energy efficiency of training neural network architectures: an empirical study [Issue: arXiv:2302.00967 arXiv: 2302.00967 [cs]]. https://doi.org/10.48550/arXiv.2302.00967
100. Yu W, Luo M, Zhou P, Si C, Zhou Y, Wang X, Feng J, Yan S (2022, July). MetaFormer is actually what you need for vision [arXiv: 2111.11418 [cs]]. https://doi.org/10.48550/arXiv.2111.11418
101. Zhang H, Dauphin YN, Ma T (2019, March). Fixup initialization: residual learning without normalization [arXiv:1901.09321 [cs] version: 2]. https://doi.org/10.48550/arXiv.1901.09321
102. Zhang H, Yu D, Yi M, Chen W, Liu T-Y. Stabilize deep ResNet with a sharp scaling factor tau. Mach Learn. 2022;111(9):3359–92. https://doi.org/10.1007/s10994-022-06192-x.
103. Zhu Y, Auer F, Gelnaw H, Davis SN, Hamling KR, May CE, Ahamed H, Ringstad N, Nagel KI, Schoppik D. SAMPL is a high-throughput solution to study unconstrained vertical behavior in small animals. Cell Rep. 2023;42(6):112573. https://doi.org/10.1016/j.celrep.2023.112573.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.