ORIGINAL RESEARCH
published: 23 June 2022
doi: 10.3389/frobt.2022.799644

Network Layer Analysis for a RL-Based Robotic Reaching Task

Benedikt Feldotto[1]*, *Heiko Lengenfelder*[1], *Florian Röhrbein*[2] and *Alois C. Knoll*[1]

[1]Robotics, Artificial Intelligence and Real-Time Systems, Department of Computer Science, Technical University of Munich, Munich, Germany, [2]Neurorobotics, Department of Computer Science, Chemnitz University of Technology, Chemnitz, Germany

Recent experiments indicate that pretraining of end-to-end reinforcement learning neural networks on general tasks can speed up the training process for specific robotic applications. However, it remains open if these networks form general feature extractors and a hierarchical organization that can be reused as in, for example, convolutional neural networks. In this study, we analyze the intrinsic neuron activation in networks trained for target reaching of robot manipulators with increasing joint number and analyze the individual neuron activation distribution within the network. We introduce a pruning algorithm to increase network information density and depict correlations of neuron activation patterns. Finally, we search for projections of neuron activation among networks trained for robot kinematics of different complexity. As a result, we show that the input and output network layers entail more distinct neuron activation in contrast to inner layers. Our pruning algorithm reduces the network size significantly and increases the distance of neuron activation while keeping a high performance in training and evaluation. Our results demonstrate that robots with small difference in joint number show higher layer-wise projection accuracy, whereas more distinct robot kinematics reveal dominant projections to the first layer.

Keywords: reinforcement learning, robotics, machine learning, robot manipulator (arms), neural networks

OPEN ACCESS

Edited by:
Pedro U. Lima,
University of Lisbon, Portugal

Reviewed by:
Aysegul Ucar,
Firat University, Turkey
Plinio Moreno,
Instituto Superior Técnico (ISR), Portugal

***Correspondence:**
Benedikt Feldotto
feldotto@in.tum.de

Specialty section:
This article was submitted to Computational Intelligence in Robotics, a section of the journal Frontiers in Robotics and AI

Received: *21 October 2021*
Accepted: *09 May 2022*
Published: *23 June 2022*

Citation:
Feldotto B, Lengenfelder H, Röhrbein F and Knoll AC (2022) Network Layer Analysis for a RL-Based Robotic Reaching Task. Front. Robot. AI 9:799644. doi: 10.3389/frobt.2022.799644

1 INTRODUCTION

Convolutional neural networks (CNNs) are well known to demonstrate a strong general feature extraction capability in lower network layers. In these network features, kernels can not only be visualized but pretrained general feature extractors can also be reused for efficient network learning. Recent research experiments propose efficient reusability for reinforcement learning (RL) neural networks as well: networks are pretrained on similar tasks and continued learning for the goal application.

Reusing (sub) networks that can be re-assembled for an application never seen before can reduce network training time drastically. A better understanding of uniform or heterogeneous network structures improves the evaluation of network performance and at the same time unveils opportunities for the interpretability of networks which is crucial for the application of machine learning algorithms, for example, in industrial scenarios. Finally, methodologies and metrics estimating network intrinsic- and inter-correlations in artificial neural networks may also enhance the understanding of biological learning. Eickenberg et al. (2017) recently demonstrated that layers serving as feature extractors in CNN could actually be found in the human visual cortex by correlating artificial networks to biological recordings.

Successful experiments to reuse end-to-end learned networks for similar robotic tasks leave open whether such networks also self-organize feature extractors or in a dynamical domain motion primitives. Here, we analyze neuron activation in networks in order to investigate activation distribution and mapping between different networks trained on similar robot reaching tasks.

In this study, we execute target-reaching end-to-end RL experiments with homogeneous robot manipulators and a variable number of revolute joints. Our work focuses on robot kinematics in a vertical plane, with every joint orthogonal to the previous link, so that subsequent joints are in principle able to replicate movements of previous joints. We introduce metrics applied to evaluate individual neuron activation over time and compare activity within individual networks all-to-all (every neuron is correlated to any other neurons in the network) and layer-wise (only corrections between neurons on the same layer are inspected). These metrics are utilized to set up a pruning procedure to maximize the information density in learned neural networks and reduce redundancy as well as unused network nodes. Exploiting these optimization procedure, we learn various neural networks with variable dimensions on robot manipulators with two to four joints, representing two to four degrees of freedom (DOF), in order to analyze similarities between network activation patterns. We hereby question whether network structures learned on a robot manipulator may find its equivalent in a network for another more or less complex robot with similar kinematics.

As a result, we demonstrate experimentally that the introduced pruning process reduces the network size efficiently keeping performance loss in bounds and hereby builds a valid basis for network analysis. We show that networks trained and iteratively pruned on the robot manipulators form distinct neuron activation. By analyzing neuron activation correlations between different networks of various sizes, mappings between neurons trained on different manipulators are found. A layer-wise interpretation reveals that networks trained for the same tasks build similar structures, but we can also discover partially similar structures between networks trained on three- or four-joint manipulators.

2 RELATED WORK

The capability of feature extraction in CNNs, alongside with a variety of analysis and visualization tools, serves as a motivation for this work on training, analysis, and pruning for networks trained with RL. Analysis methods for CNNs include regional-based methods, for example, image occlusion (Zeiler and Fergus, 2014), that aim to expose the region of an image most relevant for classification as well as feature-based methods, for example, deconvolution (Zeiler and Fergus, 2014) or guided backpropagation (Selvaraju et al., 2017). Methods combining the described techniques are, for example, introduced as Grad-CAM in Springenberg et al. (2014). These networks demonstrate class discrimination for features of deeper network layers (Zeiler and Fergus, 2014) as a basis to apply such general feature extractors to different applications after pretraining. Pretrained networks such as ResNet (He et al., 2016), which has been trained on the ImageNet1 data set, speed up training drastically by initializing CNNs applied for similar tasks. Kopuklu et al. (2019) demonstrated that even reusing individual layers in the same network can lead to a performance increase.

Recent advances pushed RL agents to reach super-human performance in playing Atari video games (Bellemare et al., 2013) (Mnih et al., 2015), chess (Silver et al., 2017), and Go (Silver et al., 2016). These results were extended to cope with continuous action spaces in, for example, Lillicrap et al. (2015) and demonstrated great performance on highly dynamic multi-actuated locomotion learning tasks such as demonstrated in the NIPS 2017 Learning to Run challenge (Kidziński et al., 2018). Vuong et al. (2019) and Carlo et al. (2020) demonstrated experimentally that knowledge learned by a neural network can be reused for other tasks in order to speed up training and hereby translate modularity concepts from CNNs to RL frameworks. Hierarchical reinforcement learning incorporates these ideas, utilizing the concept of subtask solving into neural networks, for example, in Jacob et al. (2016) for answering questions. A successful example of transfer learning to build up a general knowledge base could be demonstrated with RL in Atari games in Parisotto et al. (2016). Gaier and Ha (2019) emphasized the importance of neural architectures that can perform well even without weight learning.

With the main motivation of improving learning efficiency and reducing computational requirements, network pruning is introduced for various network architectures. Early approaches for pruning weights in artificial neural networks using second-order derivative information have been introduced in LeCun et al. (1990) and further improved in Hassibi et al. (1992). Examples for unit-based pruning by removing neurons based on their redundancy can be found for deep neural networks in He et al. (2014) and Srinivas and Babu (2015) as well as for deep RL in Livne and Cohen (2020). They use a node importance function, minimize the expected squared difference of corresponding units, or apply transfer learning techniques. A review of network compression methods can be found in Ali et al. (2021) in which additional techniques for node and weight pruning using quantization and low-rank factorization are described—all with the goal of reducing computational costs while maintaining a high level of performance.

Unit-based pruning algorithms assume the existence of redundant neurons in the network that can be safely removed. Similar to He et al. (2014) and Srinivas and Babu (2015), we aim to identify these redundant neurons by analyzing their characteristics but in contrast to static training tasks inspecting their weights only does not seem reasonable in the dynamic case of a robotic trajectory execution. He et al. (2014) proposed an entropy-based approach based on the neuron activation. To account for the dynamic motion execution, we build up on the history of activation values and a clustering methodology for the identification of redundant neurons. We provide an alternative approach for network pruning in RL tasks that is not based on transfer learning.

3 EXPERIMENTAL SETUP

In this study, we focus on a robot manipulator with operation limited to a vertical plane, a homogeneous kinematic with each

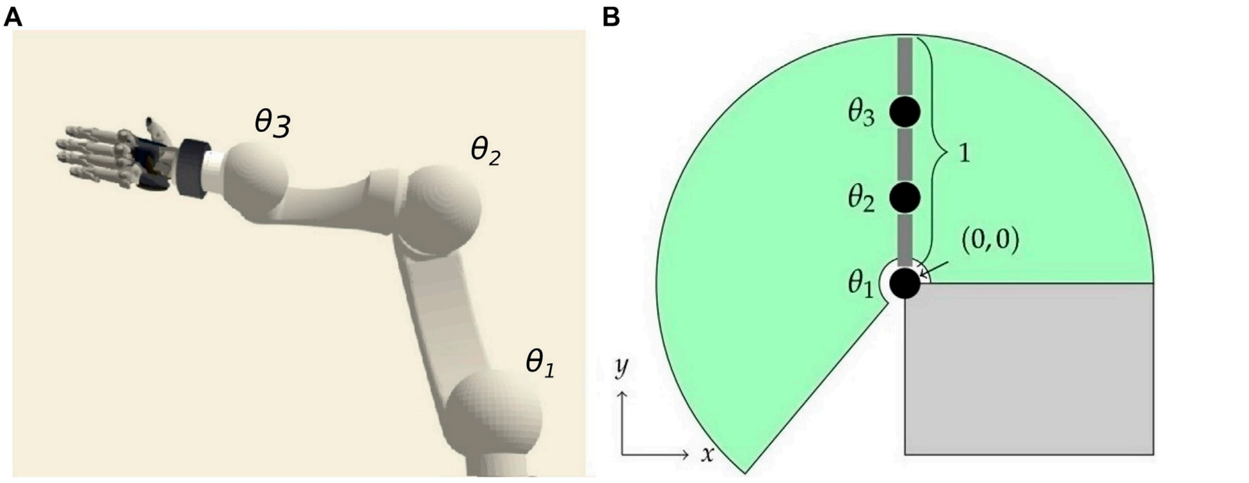

FIGURE 1 | Neuron activity is analyzed in neural networks trained for target reaching of multi-joint robotic manipulators utilizing end-to-end deep Q-Learning. We train the network in a simplified environment of a robot with 2–4 controllable joints operating in a vertical plane [**(A)**, initial configuration with joints θ_i]. The transferability to a robotic simulation **(B)** has been demonstrated with movements restricted to a vertical space.

joint orthogonal to its previous link. A neural network is trained with end-to-end reinforcement learning in order to reach predefined locations in 2D space without prior knowledge of neither robot dynamics nor the environment. Hereby, end-to-end refers to a mapping from sensory feedback in terms of actual joint positions in the cartesian space and the desired goal location to output actions as joint position commands. We apply deep Q-learning, as proposed in Mnih et al. (2015), to predict q-values, an action is selected by means of the softmax exploration policy and gradient descent of the network weights is handled by the Adam solver (Kingma and Ba, 2015).

For performance reasons, our experiments are executed within a simplified simulation environment, as shown conceptually in **Figure 1B**, but exemplary behaviors have been successfully transferred to a realistic robotic simulation (**Figure 1A**) on the basis of a commonly used industrial robot. We simulate robots with 2–4 DOF that are implemented as revolute joints restricted to movements in the vertical space. They are actuated with PID position controllers. For all experiments, the neural networks originally consist of six fully connected hidden layers with ReLU activation functions but may be reduced in the pruning process we introduce.

The network input vector $\boldsymbol{x}$ encodes actual robot joint angles $\widehat{\theta}_i$ for n joints as their sine and cosine contribution for every control step t (control cycle time of 50 ms) as well as the desired goal position in Cartesian coordinates $[x*, y*]$ as:

$$\boldsymbol{x}^{(t)} = \begin{bmatrix} sin\left(\hat{\theta}_1^{(t)}\right) \\ cos\left(\hat{\theta}_1^{(t)}\right) \\ \cdots \\ sin\left(\hat{\theta}_n^{(t)}\right) \\ cos\left(\hat{\theta}_n^{(t)}\right) \\ x^* \\ y^* \end{bmatrix}. \quad (1)$$

The output layer contains $3n$ neurons as the action of every individual joint i is quantized into the three change of motion states $\{1, -1, 0\}$ as forward, backward, and no motion for each joint with joint angle changes of $\pm 0.05 rad$. The goal state of an agent is a randomly instantiated 2D location to be reached with the robot fingertip in maximum 60 control steps, each representing $50\, ms$. The distance between the goal position p^* and the tip $\hat{p}$ is mapped into [0, 1] and squared to serve as the reward function $r(t_i) := \left(\frac{1}{|\hat{p}(t_i) - p^*|_{L2} + 1}\right)^2$. All network results that are presented passed a validation test consisting of 300 test episodes. This test also serves as the pruning baseline: the probability of a type two error for reaching the final reward threshold $\bar{r} = 0.9$ with an accuracy $\bar{\rho} = 0.9$ lies below significance $\alpha = 0.05$ on the test data.

4 NEURON ACTIVATION ANALYSIS

We first analyze individual neuron activation inside multiple neural networks trained on the introduced target reaching a robotic manipulator. This initial analysis serves as baseline for pruning and projection evaluation; therefore, we study only three-joint robotic manipulators in depth before we investigate a comparison of different kinematic structures.

We define a distance metric between neurons that is based on the neuron activation history in scope of every episode in order to account for the dynamics in motion trajectory learning. All neuron activation values over the course of an episode are collected in a vector $\boldsymbol{z}_{n_i}^{(E)}$ for every neuron n_i of the network in episode E. Utilizing the linearity of applied ReLU activation functions, we normalize this activation in the range [0, 1] in reference to the maximum value attained. For a set of sample episodes $\mathcal{E}$, representing a set of potential robot actions, we define the distance of neurons n_i and n_j as:

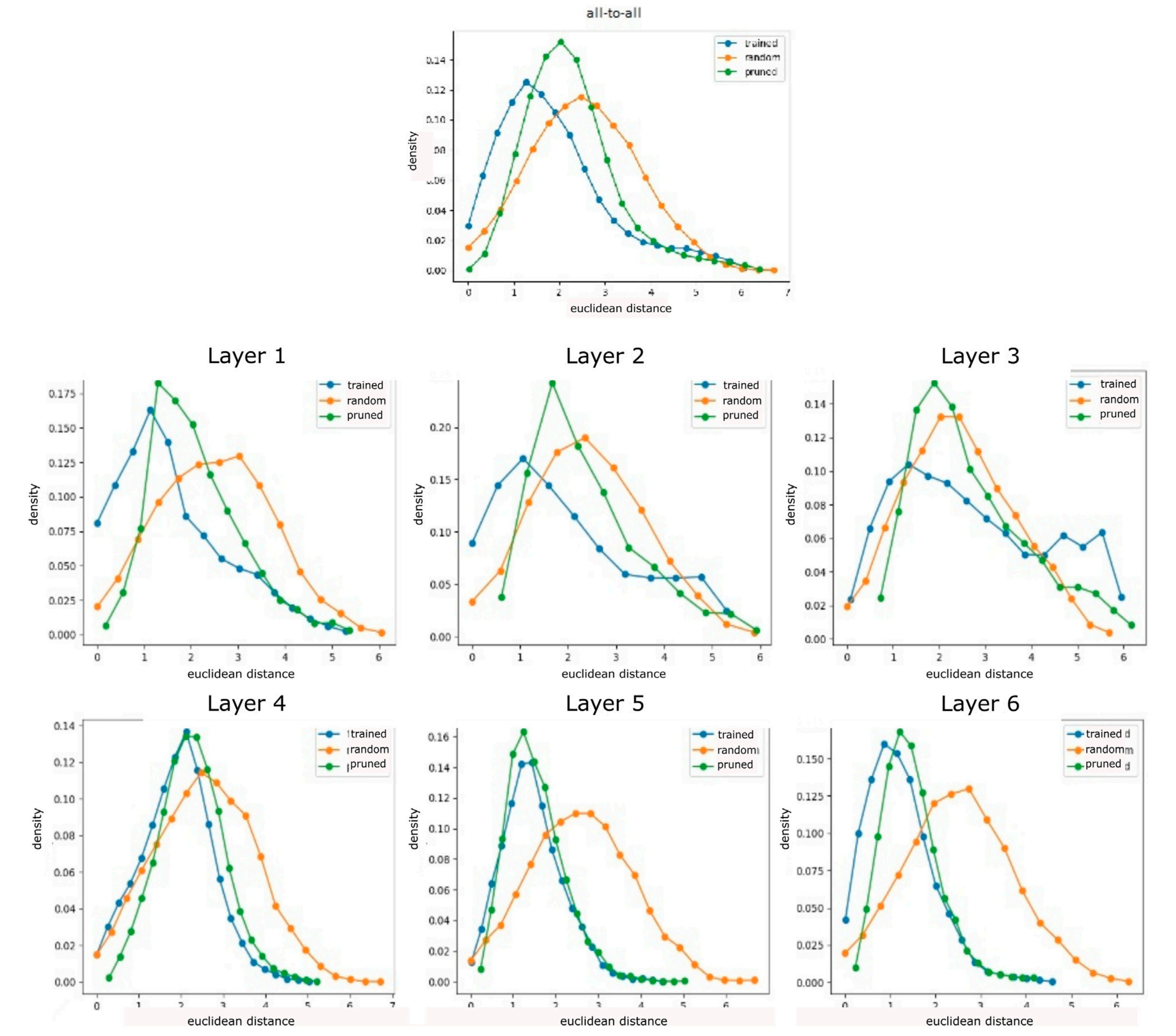

FIGURE 2 | Neuron activation analysis for randomly initialized, trained, and pruned networks on a three-joint manipulator (averages over 500 sample episodes and 20 trained agents). Distance measured between all-to-all (all neurons in a network are correlated among each other, left) neurons and layer-wise (for every neuron only neurons on this layer are considered for correlation, right) indicate a bell-shaped distribution with higher mean in the first and last layer. Pruning sharpens the bell shape, increasing the mean, but reducing very high distance scores.

$$d\left(n_i, n_j\right) := \frac{1}{|\mathcal{E}|} \sum_{E \in \mathcal{E}} \left| \frac{z_{n_i}^{(E)}}{Z_{n_i}} - \frac{z_{n_j}^{(E)}}{Z_{n_j}} \right|_{L2}, \tag{2}$$

with $z_{n_i}^{(E)} \in \mathbb{R}_{\geq 0}^{T}$ denoting the vector containing activation series of neuron n_i in episode E and $Z_{n_i} \in \mathbb{R}_{>0}$ the maximum activation of n_i in all episodes $\mathcal{E}$. For a layer-wise analysis **Eq. 2** is adapted accordingly, only considering distances to neurons that belong to the same layer. The upper triangular matrix of a distance matrix D holds all values $d(n_i, n_j)$ with $i \geq j$. The density distribution of neuron distances can be approximated by collecting all values in the upper triangular matrices of D.

In addition, hierarchical clustering as described in Hastie et al. (2009) is applied to individual network layers in order to reveal neuron groups that show similar activation behavior. We form groups that minimize the mean cluster distance $D(C_l)$ of neurons involved as:

$$D\left(C_l\right) := \frac{1}{|C_l|\left(|C_l| - 1\right)} \sum_{n_{il} \in C_l} \sum_{n_{jl} \in C_l \setminus \{n_{il}\}} d\left(n_{il}, n_{jl}\right), \tag{3}$$

for neuron cluster C of layer l. We conduct an experiment with a set of $M = 20$ networks and 48 neurons per hidden layer, for the three-joint manipulation task. A reference set of untrained

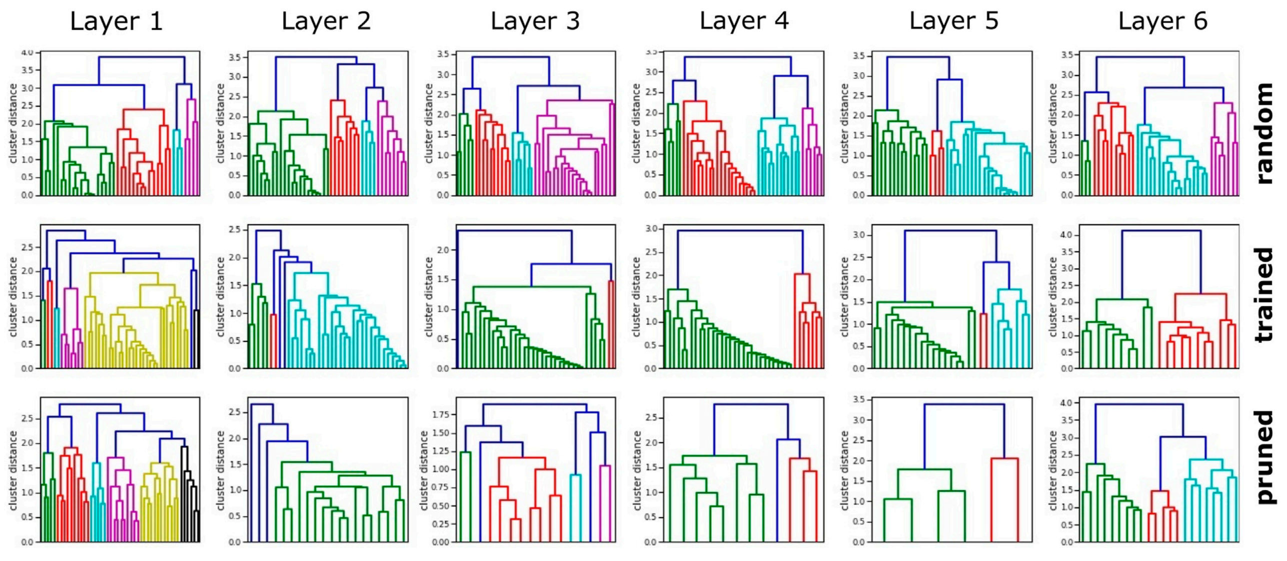

FIGURE 3 | Clustering dendrograms are generated based on the distance measures for an exemplary trained network. Untrained networks show very similar clusters, trained network highlights cluster groups, and pruning reduces neurons while increasing cluster distance. The first layer generally keeps the most distinct clusters, and the penultimate layer shows the strongest neuron reduction.

networks with identical structure is initialized by Xavier initialization (Glorot and Bengio, 2010). Neuron distances are averaged from a set of $m = 500$ sample episodes.

The distance distribution in randomized networks forms a bell-shaped distribution globally as well as layer-wise (**Figure 2**). However, the all-to-all distribution of trained networks primarily indicates a lower standard deviation and mean than that of random networks, with a slight distortion at high distances. The layer-wise analysis reveals that these higher distance scores occur increasingly on network layers closer to the output, in particular in the second half of layers. In contrast, lower layers demonstrate close to normal distributions. Clustering reveals a variety of distances for all layers in untrained randomly initialized networks (**Figure 3**) which is kept on the first layer only in trained networks, especially in middle layers, we observe clusters with low distances that have emerged during training.

An intrinsic network analysis reveals that a successful training changes visibly the neuron activation characteristics, highly depending on the neuron position in the network.

5 HEURISTIC NETWORK PRUNING

Non-uniform density distributions and low cluster differences in the inspected neuron activation indicate potential for network pruning. Dense information representation is a requirement for the comparison of different networks. For this purpose, we propose a pruning procedure that iteratively unifies neurons with similar activation, identified as small cluster distances, and retrains the network. Hereby, a trade-off between reduced network size and maintaining high-performance learning is aspired.

We apply Breadth First Search on the resulting cluster tree of every network layer. The first encountered clusters with distance **Eq. 3** below threshold $\bar{d}_\tau$, which is defined as a fraction τ of the maximum cluster distance, are selected to form the layer segmentation $\mathcal{C}$. Based on this neuron segmentation $\mathcal{C}^{(l)}$ of layer l, a reduced network is constructed that represents every cluster as a single neuron. Original network weights are reused for the initialization of the reduced network. We exploit the linearity of ReLU activation functions and assume identical neuron behavior which is altered only by linear scaling inside all clusters. Without loss of generality cluster activation, ζ_C is defined such that scaling factors $\gamma_n > 0$ (normalized magnitude of total cluster activation) of contained neurons sum to one and $\forall n \in C$: $\zeta_C = \frac{z_n}{\gamma_n}$, with z_n denoting the activation of neuron n, holds. For cluster $C \in \mathcal{C}^{(l)}$ and arbitrary neuron $n \in C$, the forward propagation of z_n can be rearranged to form the forward propagation of the cluster activation as:

$$\zeta_C = ReLU\left[\sum_{D\in\mathcal{C}^{(l-1)}} \zeta_D \frac{1}{\gamma_n} \sum_{m\in D} w_{nm}\gamma_m\right], \tag{4}$$

with w_{nm} denoting the weight from neuron m to n. **Eq. 4** acts as an approximation that in practice is only achieved by clusters of silent neurons that are not activated at all. Therefore, in order to improve stability all neurons of a cluster contribute to the reduced network weights ω as $\omega_{CD} = \frac{1}{\gamma_n}\sum_{m\in D} w_{nm}\gamma_m$. Scaling factors γ_n are generated from the maximum activation Z_n of **Eq. 2** of the respective neuron n.

In order to evaluate the introduced pruning procedure, we conduct experiments with a set of $M = 20$ neural networks (six hidden layers and 48 neurons each) trained for the three-joint manipulation task. Network reduction is applied with a set of $m =$

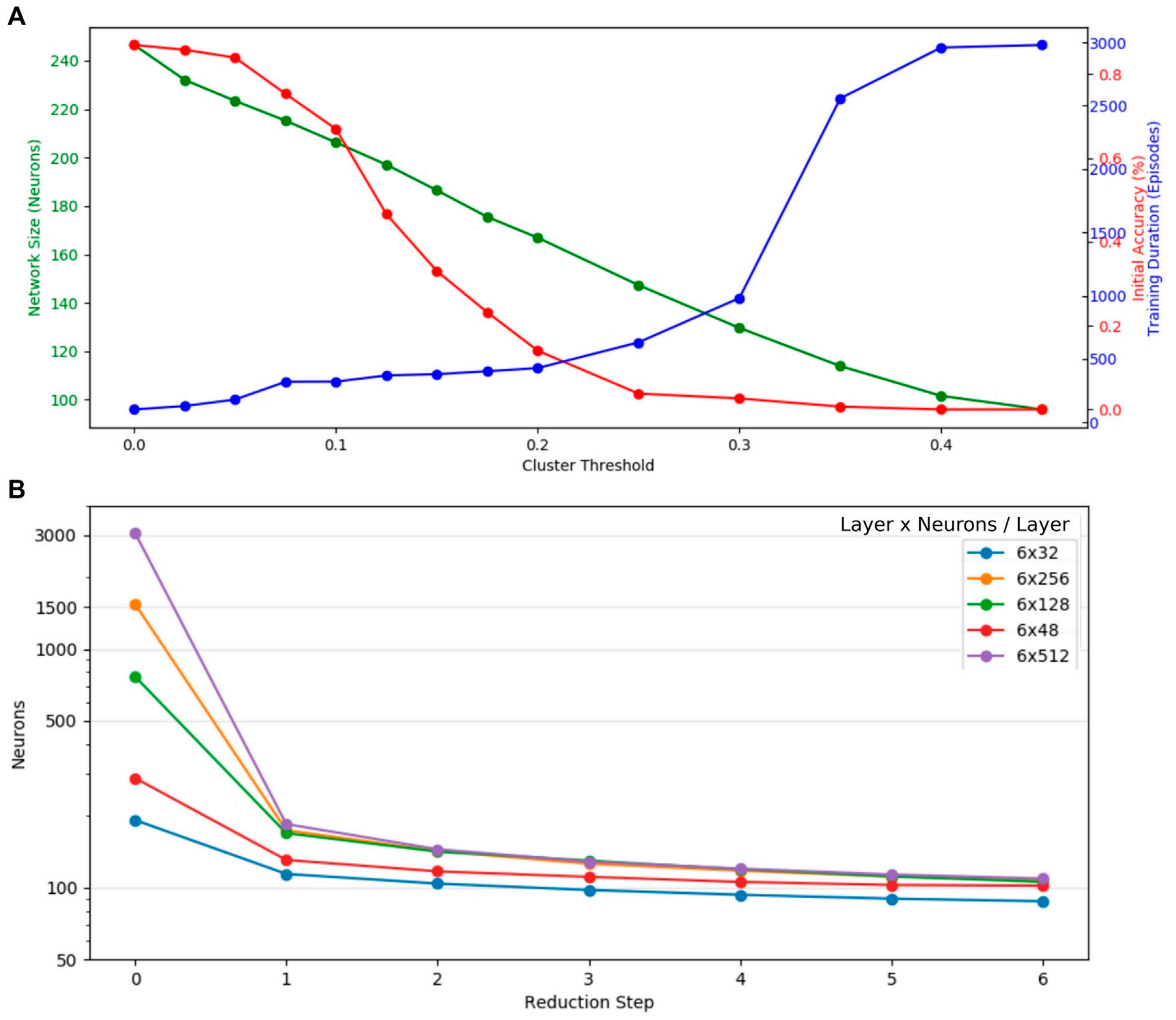

FIGURE 4 | Evaluation of heuristic cluster-based network pruning on the example of a three-joint manipulator. **(A)** Even though the initial network accuracy decreases rapidly, the training duration (number of episodes executed until the validation set is passed) only increases significantly with cluster thresholds larger than 0.3. As a trade-off between minimal network size after pruning and efficient training $\tau = 0.2$ has been picked as the optimal cluster threshold and was applied for all further experiments. **(B)** The first reduction step shows the strongest reduction for all networks. They have been initialized with a different neuron count per layer. Layers with more than 128 neurons are reduced to a very similar neuron count in the first pruning step.

300 sample episodes, and presented results are averaged over the set of networks which reached sufficient performance.

Results presented in **Figure 4A** show a nearly linear correlation between cluster threshold and resulting pruned network size if networks had an identical initial layer size of 48 neurons. In case of $\tau = 0$ only silent neurons are reduced, which does not affect the performance of the network, though reduces the network size significantly (initial size of all networks: 323 neurons). For values of $\tau \in (0, 0.1]$, the network is reduced, but no strong effect is apparent on initial accuracy [%] and training duration (number of episodes executed until the validation set is passed). We observe interesting behavior in range of $\tau \in (0.1, 0.22]$, as the initial accuracy decreases significantly, whereas the duration for retraining the networks barely increases. This implicates that the main processes in the network remain intact after reduction, whereas for $\tau > 0.2$ a strong increase in training duration indicates a loss of relevant information. As a trade-off between minimal network size and efficient training $\tau = 0.2$ has been selected as cluster threshold and was applied in all further experiments. As the pruning process highly depends on the initial network size, we analyze networks of initial hidden layer sizes of 32, 48, 128, 256, and 512 within the same test setup. The results shown in **Figure 4B** emphasize the first reduction step as the most dominant. Noticeably, large networks of initial layer neuron count of 128, 256, and 512 reach the similarly pruned network size already in the first iteration step. For subsequent reduction steps, the network size plateaus. Inspection of neuron-per-layer counts reveal that small initial networks (32, 48) taper with depth, compared to bigger initial networks that form an hourglass shape. The average

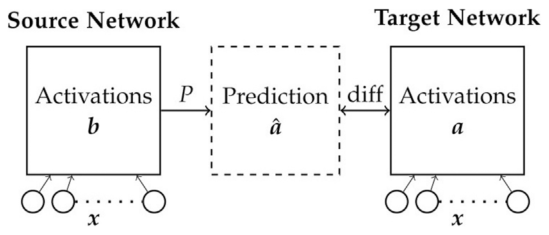

FIGURE 5 | Analysis of inter network mappings. Sets of two networks are trained on robots with different number of joints. A projection matrix P that reflects the network similarity is calculated to compute $\hat{\boldsymbol{a}}$ from the source network neuron activation $\boldsymbol{b}$ with minimal difference to $\boldsymbol{a}$.

network shape of 256 and 512 neuron networks after three reduction steps turns out as $\bar{s} = [\,51.6 \;\; 21. \;\; 15.9 \;\; 12.6 \;\; 10.2 \;\; 16.7\,]$. Network intrinsic neuron distance densities of pruned networks (**Figure 2**) implicate an increased homogeneous information representation compared to networks trained straight away. The bell-shaped distribution with higher mean shows lower variance. In addition, outliers with high distance scores are reduced. While clusters remain rather similar on the first and last layer, in particular the cluster distances on middle layers are drastically increased along with the reduced cluster number. Overall, we find that our pruning process reduces network size efficiently and hereby shows a visible effect on neuron activation toward a rather uniform distribution and distinct cluster structure.

6 CORRELATIONS IN NETWORKS TRAINED FOR MULTI-JOINT ROBOTS

Based on both, the analysis of individual neuron activation and the heuristic network pruning, we investigate mappings of neuron activation between different networks learned on robot manipulators with 2–4 joints. Here, the goal is to estimate whether activation patterns are similar in networks trained for the different robot kinematics. For this purpose, we construct a unidirectional linear projection between source and target network and analyze its accuracy and structure. Based on the source network neuron activation $\boldsymbol{b} \in \mathbb{R}^K_{\geq 0}$, resulting from input $\boldsymbol{x}$, a prediction $\hat{\boldsymbol{a}} = \boldsymbol{b}^T P$ of the target activation $\boldsymbol{a} \in \mathbb{R}^M_{\geq 0}$ for the same input $\boldsymbol{x}$ is given by the projection matrix $P \in \mathbb{R}^{K \times M}_{\geq 0}$ (**Figure 5**). The projection is constructed based on a set of N training inputs X that yield activation matrices $A \in \mathbb{R}^{N \times M}_{\geq 0}$ and $B \in \mathbb{R}^{N \times K}_{\geq 0}$ of the target and source network, respectively. In order to obtain a procedure invariant to neuron scaling, individual columns of A and B are normalized to the interval [0, 1] dividing by the maximal values contained. The resulting projection $\bar{P}$ can be adjusted to fit the original training data by $P_{km} = \frac{\alpha_m}{\beta_k} \bar{P}_{km}$.

Two approaches for projection construction are considered. `Greedy mapping` predicts each target neuron from the source neuron with minimal distance **Eq. 2**, and every entry of the greedy projection matrix $\bar{P}^g_{km}$ is 1 if $k = \arg\min_{i \in [K]} \{d(m, i)\}$ and 0 otherwise. `Linear mapping` incorporates all source neurons into the prediction of a target neuron by linear combination. Projection vectors $\boldsymbol{p}_m$, predicting the behavior of neuron m, are given by the solution of quadratic optimization with linear boundary constraints for each target neuron individually. Hereby, the mean squared error plus lasso regularization, to enforce sparsity of solution vectors, is minimized finding the best projection $\boldsymbol{p}$, that is,

$$\begin{aligned} &\text{minimize } \frac{1}{2} \left| \bar{B} \boldsymbol{p} - \frac{\boldsymbol{a}^{\downarrow}_m}{\alpha_m} \right|^2_{L2} + \lambda |\boldsymbol{p}|_{L1}, \\ &\text{subject to } \boldsymbol{p} \geq \boldsymbol{0} \end{aligned} \tag{5}$$

where $\bar{B}$ denotes the matrix of source activations scaled by β_k, $\boldsymbol{a}_m$ the target activations, and $\lambda \in \mathbb{R}_{\geq 0}$ the regularization strength. As mapping of two networks should be invariant to neuron scaling, all individual neuron activations are projected into the interval [0, 1] with neuron specific scaling factors β_k and α_m for the source and target network neurons, respectively. The solution vectors $\bar{\boldsymbol{p}}^\star_m$ are stacked to form the linear projection matrix $\bar{P}^l := [\,\bar{\boldsymbol{p}}^\star_1 \;\ldots\; \bar{\boldsymbol{p}}^\star_M\,]$. Input samples X are deduced from a set of sample episodes of the target network without duplicates. Input vectors of robot manipulators with different joint count are transformed by either duplicating best aligning joints or unspecified joints being set to zero, for a more or less complex source network, respectively (**Figure 6** middle right).

6.1 Evaluation Metrics

Projections are evaluated with regard to their goodness to fit a set of validation samples $X^{\mathcal{V}}$ and according to heuristic metrics that directly analyze a projection structure. The mean absolute prediction error is normalized by the prediction error of the zero projection $P_0 \in \{0\}^{K \times M}$ to construct the normalized error $E(P, X)$ that is invariant to weight scaling and adding silent neurons:

$$\bar{E}(P, X) := \frac{E(P, X)}{E(P_0, X)} = \frac{1}{|A|_1} \sum_{m=1}^{M} \left| \boldsymbol{a}^{\downarrow}_m - B \boldsymbol{p}_m \right|_1. \tag{6}$$

The entropy of a target neuron's projection $\boldsymbol{p}_m$ is referred to as the saturation of neuron m and projection P is the mean of all neuron saturations. A low saturation implies that few neurons suffice to describe the behavior of m. We calculate the overall projection saturation $S(P)$ according to **Eq. 7**:

$$\mathcal{S}(P) := -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} P_{km} \log_K (P_{km}) \in [0, 1]. \tag{7}$$

The usage of source network neurons to describe the target network is indicated by the coverage $\mathcal{C}$. It is defined as the entropy of the stochastic process that picks a target neuron m uniformly at random and passes it on to the source network according to the distribution $\frac{\boldsymbol{p}_m}{|\boldsymbol{p}_m|_{L1}}$. A low coverage value implies low utilization of the source network.

$$\begin{aligned} &\mathcal{C}(P) := -\frac{1}{K} \sum_{k=1}^{K} \kappa_k \, log_K (\kappa_k), \\ &\text{with } \kappa_k = \frac{1}{M} \sum_{m=1}^{M} \frac{P_{km}}{|\boldsymbol{p}_m|_{L1}}. \end{aligned} \tag{8}$$

The same statistical process is applied to construct a layer-wise projection $\mathcal{P}_{ij}$. It describes the probability of reaching the ith layer

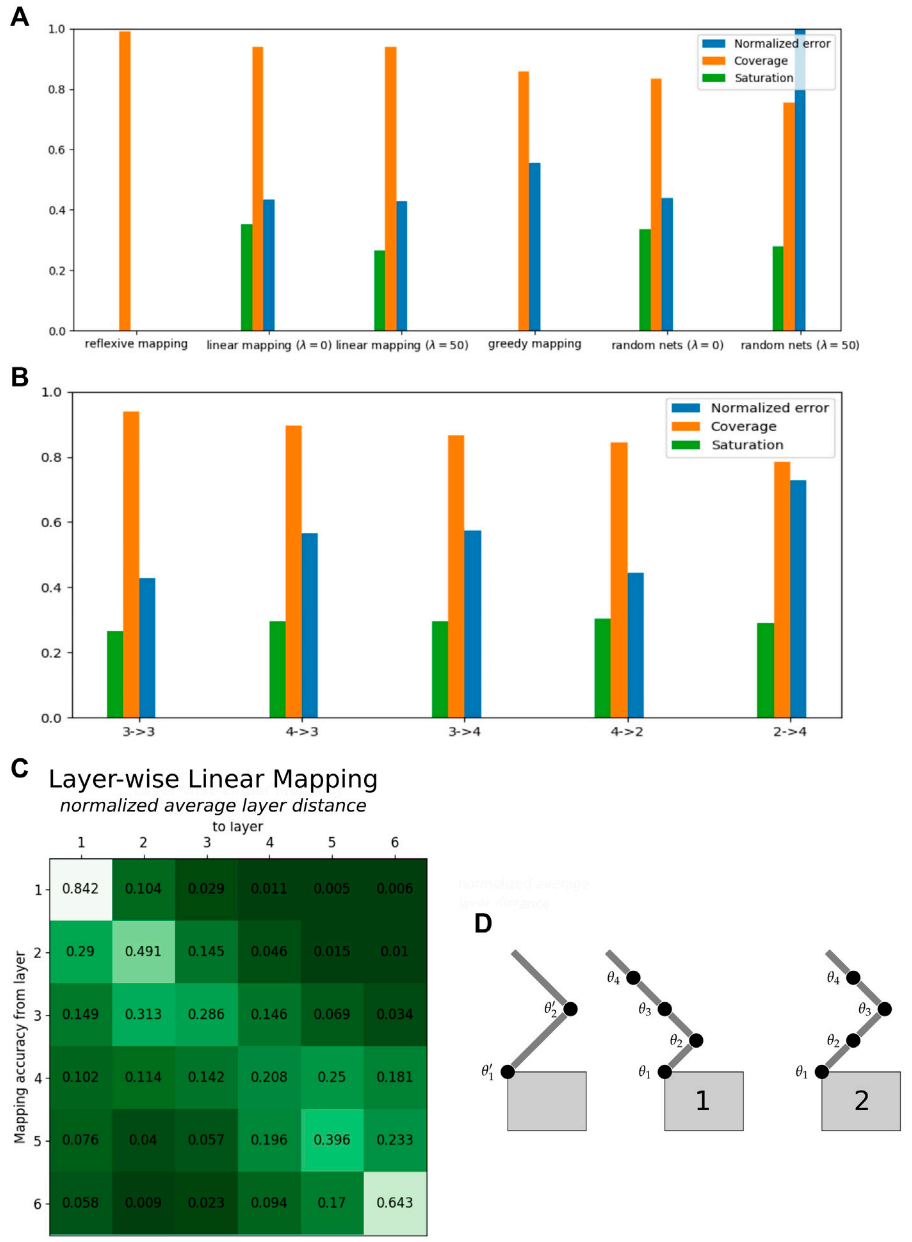

FIGURE 6 | Projection of neuron activation between networks trained for variable joint robot manipulators (data averaged with five networks each and three pruning steps). **(A)** Benchmark of mapping technique and evaluation metrics: on the example of multiple three-joint manipulator networks, we find linear mapping with $\lambda = 50$, the superior mapping approach in contrast to greedy mapping. In particular, coverage and normalized errors indicate mapping quality well in comparison to untrained networks. **(B)** Neuron activation correlations of networks trained on robots with different joint count (2–4 refers to a mapping from networks trained for a two-joint to a four-joint robotic arm): the mapping error gets higher with increased difference in joint numbers, and the coverage accordingly decreases. Mapping a network with higher complexity into lower complex ones performs slightly better than vice versa. In this study, the mappings 4-2 are closest to the performance of the native 3-3 mappings. This mapping is influenced by a proper transformation of sensory inputs to the increased number of input neurons on the first layer (bottom right). The results are demonstrated for balanced mapping in **(D2)** as $\theta_1' = \theta_1$, $\theta_2' = \theta_3$, $\theta_2 = 0$, $\theta_4 = 0$ which outperform the naive mapping $\theta_1' = \theta_1$, $\theta_2' = \theta_2$, $\theta_3 = 0$, $\theta_4 = 0$ depicted in **(D1)**. All results are the mean of 25 mappings. **(C)** A layer-wise linear mapping with $\lambda = 50$ is not optimal, but strongest correlations can be found between corresponding layers. This is represented in the higher diagonal values in the table of normalized average layer distances on bottom left. Here, layers 1 and 6 show best mapping (initialization with six layers each 256 neurons before pruning and random nets with average pruned network size of s = [46 22 16 13 8 20] neurons per layer).

$L_i^{(K)}$ of the source network when starting in some uniformly random neuron in the jth layer $L_j^{(M)}$ of the target network.

$$\mathcal{P}_{ij} := \frac{1}{|L_j^{(M)}|} \sum_{k \in L_i^{(K)}} \sum_{m \in L_j^{(M)}} \frac{P_{km}}{|\boldsymbol{p}_m|_{L1}}. \tag{9}$$

6.2 Results

For each robot manipulator with two, three, and four joints and $M = 5$ networks are first trained and then pruned in three steps before we analyze all possible mappings "a-b" between the respective sets. A set of validation inputs $X^{\mathcal{V}}$ is generated for $m = 300$ sample episodes of the target network and metrics evaluated. As a baseline, we map all three joint manipulator agent networks with an initial neuron count of 256 for each of the six hidden fully connected layers, among each other. As expected, as baseline mappings of networks to themselves (referred to as reflexive mapping) show zero error and saturation and coverage of 1 (**Figure 6**). However, greedy mapping shows a high normalized error and low coverage when compared to the linear mapping and thus is considered an inferior approach. In this baseline, we extract linear mapping with regularization strength of $\lambda = 50$ as the best metric as it indicates coverage and normalized error most significant on trained in contrast to random networks. Layer-wise linear projection ($\lambda = 50$) is not optimal, but we observe the best mapping to the respective layers, shown on the diagonal axis in the table of **Figure 6** (middle left). Here, layers one and six show the strongest correlation presumably due to increasingly specialized neurons at the input and output of the network.

Linear mapping ($\lambda = 50$) has been applied between sets of two-, three-, and four-joint robot manipulators (**Figure 6** middle right); random networks are initialized by the average network size of the respective joint count as evaluated with pruning. Scenarios 3-4 and 4-3 show similar prediction errors but indicate a higher mean error than 4-2 mappings. Latter mapping performs similar to the baseline, which might be induced by the fact that we transform inputs in a balanced way so that the four-joint arm can act like a two-joint arm (figure on the right, we choose the transformation 4b). It shows lower coverage of the source network, which is partially related to the fixed input channels for the source networks after input transformation. The worst performance in terms of prediction error results from scenario 2-4 where the two-joint manipulator networks are barely able to replicate the behavior of the four joint networks. Generally, the more distinct the robots the worse the mapping, except input transformation is implemented in a meaningful way. More complex networks map slightly better into less complex one, as compared to the opposite way round.

A deeper insight into how the source network is used can be drawn from mean layer-wise projections (**Figure 7**). The baseline scenario 3-3 shows more significant correlation to its respective layer, the closer it is to the input or output. The first layers of 3-4 and 4-3 mappings seem to follow the behavior of the baseline, whereas the deeper layers show no significant correlation. Contrary to the performance of the overall metrics, scenario 4-2 shows no strong

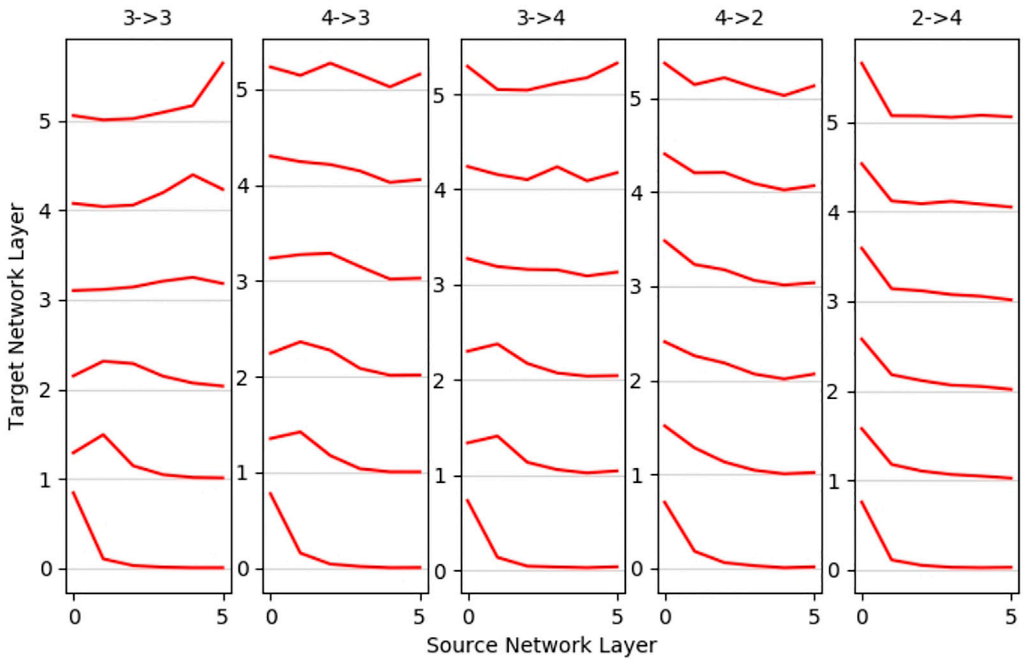

FIGURE 7 | Mean layer-to-layer projections: networks trained for robots with similar degrees of freedom show better layer-to-layer mappings. The first layer of the source network shows high utilization for mappings to all other layers, and the penultimate layer is the most unlikely to be utilized. The middle layer maps reasonably well for 4-3 and 3-4 mappings, and more distinct robots as 4-2 and strongest in 2-4 mostly utilize first layer neurons only.

layer-wise correlation, which is even worse in the inverted 2-4 mapping. If layers do not map well, all target layers tend to map to the lower layers, especially the first layer (most prominent in 2-4 mappings) of the source network; only a small tendency is visible of the output layer mapping to other output layers. We hypothesize that this phenomenon is credited to first layers having the highest neuron count and activation variance. Overall, we do find that a good mapping correlation when the source network is able to imitate the behavior of the target network, a suitable input transformation turned out to be crucial here. 4-2 mappings showed the lowest error, but networks trained on three- and four-joint networks map better into their respective layer.

7 CONCLUSION

In this study, we analyzed individual neuron activation within and correlations between neural networks trained for goal reaching of vertical space robot manipulators with two, three, and four joints. We analyzed and classified the activation in order to implement a pruning algorithm that removes redundant neurons and increases information density in the network. Finally, we analyzed correlations between the overall and layer-wise neuron activation of networks trained on robots with different joint number by projection mapping. Our results demonstrate that networks develop distinct activation patterns on individual neuron layers with bell-shaped distribution of activation densities. This distribution is compressed by our pruning algorithm that merges similar neuron activation classes mostly on the inner network layers. Networks trained for robots with only small joint number difference show a good layer-wise correlation of neuron activation. The more distinct the robot kinematic is in terms of joint number, the more important is a proper input transformation that fits the different network input layers. Here, mapping among equivalent network layers turns out less strong and instead dominant mapping to the first network layer is revealed. All experiments are benchmarked by comparison against untrained networks and self-correlations for multiple networks trained for the same task. Our results help to improve explainability of reinforcement learning in neural networks for robot motion learning and highlight network structures that can be reused on similar tasks after pretraining. The experiments conducted are limited to robot manipulators of 2–4 joints acting in a vertical plane, and as mapping quality is decreasing with greater joint distance number, we would expect worse mapping quality when additional joints are added. However, the underlying methodologies and workflow we present incorporating network pruning and mapping could be transferred to other reinforcement learning tasks and other robotic configuration setups as well. Analysis of neuron activation has been introduced in other contexts before; here, we utilize it for the analysis of the specific use case of vertical space robot manipulation. In future work, our pruning algorithm will be extended to also reduce the overall number of layers, and we will analyze additional network parameters and examine experimentally the reuse of network structures with good correlation. While our work focuses on homogeneous robot kinematics, we may also extend and evaluate the introduced mapping procedure to non-homogeneous kinematics.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusion of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

BF, FR, and AK framed the research question. BF and HL executed the experiments and analyzed the data. BF, HL, FR, and AK interpreted the data and wrote the manuscript.

FUNDING

This research received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (HBP SGA2) and No. 945539 (HBP SGA3).

REFERENCES

Ali, A., Xie, X., and Mark, W. (2021). Literature Review of Deep Network Compression. *Informatics* 8 (4), 1–12.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. *J. Artif. Intell. Res.* 47, 253–279. doi:10.1613/jair.3912

Carlo, D. E., Tateo, D., Bonarini, A., Restelli, M., Milano, P., and Peters, J. (2020). *Sharing Knowledge in Multi-Task Deep Reinforcement Learning*. New Orleans, United States: ICLR.

Eickenberg, M., Gramfort, A., Varoquaux, G., and Thirion, B. (2017). Seeing it all: Convolutional Network Layers Map the Function of the Human Visual System. *NeuroImage* 152 (9), 184–194. doi:10.1016/j.neuroimage.2016.10.001

Gaier, A., and Ha, D. (2019). *Weight Agnostic Neural Networks*. Vancouver, Canada: Advances in Neural Information Processing Systems, 32.

Glorot, X., and Bengio, Y. (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (IEEE), 249–256.

Hassibi, B., and Stork, D. (1992). "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in *Advances in Neural Information Processing Systems, Volume 5*. Editors S. Hanson, J. Cowan, and C. Giles (San Francisco, United States: Morgan-Kaufmann).

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (IEEE), 770–778. doi:10.1109/cvpr.2016.90

He, T., Fan, Y., Qian, Y., Tan, T., and Yu, K. (2014). "Reshaping Deep Neural Network for Fast Decoding by Node-Pruning," in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (IEEE), 245–249. doi:10.1109/icassp.2014.6853595

Jacob, A., Rohrbach, M., Darrell, T., and Klein, D. (2016). "Learning to Compose Neural Networks for Question Answering," in 2016 Conference of the North

American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference (IEEE), 1545–1554.

Kidziński, L., Mohanty, S. P., Ong, C. F., Huang, Z., Zhou, S., Anton, P., et al. (2018). "Learning to Run Challenge Solutions: Adapting Reinforcement Learning Methods for Neuromusculoskeletal Environments," in *The NIPS'17 Competition: Building Intelligent Systems* (Springer), 121–153.

Kingma, D. P., and Ba, J. L. (2015). "Adam: A Method for Stochastic Optimization," in 3rd International Conference on Learning Representations (San Diego, United States: ICLR 2015-Conference Track Proceedings), 1–15.

Kopuklu, O., Babaee, M., Hormann, S., and Rigoll, G. (2019). "Convolutional Neural Networks with Layer Reuse," in Proceedings - International Conference on Image Processing, ICIP (IEEE), 345–349. doi:10.1109/icip.2019.8802998

LeCun, Y., Denker, J. S., and Sara, A. (1990). "Solla. Optimal Brain Damage (Pruning)," in *Advances in Neural Information Processing Systems* (Mitpress), 598–605.

Lillicrap, T. P., Hunt, J. J., Alexander, P., Heess, N., Tom, E., Tassa, Y., et al. (2015). *Continuous Control with Deep Reinforcement Learning*. Ithaca, United States: arXiv preprint arXiv:1509.02971.

Livne, D., and Cohen, K. (2020). *PoPS: Policy Pruning and Shrinking for Deep Reinforcement Learning*. Toronto, Canada: IEEE Journal of Selected Topics in Signal Processing. 14 (4), 789–801.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level Control through Deep Reinforcement Learning. *Nature* 518 (7540), 529–533. doi:10.1038/nature14236

Parisotto, E., Ba, J. L., and Salakhutdinov, R. (2016). "Actor-Mimic Deep Multitask and Transfer Reinforcement Learning," in 4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings (IEEE), 1–16.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). "Grad-cam: Visual Explanations from Deep Networks via Gradient-Based Localization," in Proceedings of the IEEE International Conference on Computer Vision (IEEE), 618–626. doi:10.1109/iccv.2017.74

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. Ithaca, United States: arXiv preprint arXiv:1712.01815.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529 (7587), 484–489. doi:10.1038/nature16961

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). *Striving for Simplicity: The All Convolutional Net*. Ithaca, United States: arXiv preprint arXiv: 1412.6806.

Srinivas, S., and Babu, R. V. (2015). *Data-Free Parameter Pruning for Deep Neural Networks*. Norwich, England: BMVC.

Vuong, T. L., Van Nguyen, D., Nguyen, T. L., Bui, C. M., Kieu, H. D., Ta, V. C., et al. (2019). "Sharing Experience in Multitask Reinforcement Learning," in IJCAI International Joint Conference on Artificial Intelligence (Macao, China: IJCAI), 3642–3648. doi:10.24963/ijcai.2019/505

Zeiler, M. D., and Fergus, R. (2014). "Visualizing and Understanding Convolutional Networks," in European Conference on Computer Vision (Springer), 818–833. doi:10.1007/978-3-319-10590-1_53

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.