The Surprising Effectiveness of Representation Learning for Visual Imitation

Jyothish Pari*
New York University
jp5981@nyu.edu

Nur Muhammad
(Mahi) Shafiullah*
New York University
mahi@cs.nyu.edu

Sridhar Pandian
Arunachalam
New York University
sa5914@nyu.edu

Lerrel Pinto
New York University
lerrel@cs.nyu.edu

Abstract—While visual imitation learning offers one of the most effective ways of learning from visual demonstrations, generalizing from them requires either hundreds of diverse demonstrations, task specific priors, or large, hard-to-train parametric models. One reason such complexities arise is because standard visual imitation frameworks try to solve two coupled problems at once: learning a succinct but good representation from the diverse visual data, while simultaneously learning to associate the demonstrated actions with such representations. Such joint learning causes an interdependence between these two problems, which often results in needing large amounts of demonstrations for learning. To address this challenge, we instead propose to decouple representation learning from behavior learning for visual imitation. First, we learn a visual representation encoder from offline data using standard supervised and self-supervised learning methods. Once the representations are trained, we use non-parametric Locally Weighted Regression to predict the actions. We experimentally show that this simple decoupling improves the performance of visual imitation models on both offline demonstration datasets and real-robot door opening compared to prior work in visual imitation. All of our generated data, code, and robot videos are publicly available at https://jyopari.github.io/VINN/.

I. INTRODUCTION

Imitation learning serves as a powerful framework for getting robots to learn complex skills in visually rich environments [52, 40, 11, 53, 49]. Recent works in this area have shown promising results in generalization to previously unseen environments for robotic tasks such as pick and place, pushing, and rearrangement [49]. However, such generalization is often too narrow to be directly applied in the diverse real-world application. For instance, policies trained to open one door rarely generalize to opening different doors [44]. This lack of generalization is further exacerbated by the plethora of different options to achieve generalization: either needing hundreds of diverse demonstrations, task-specific priors, or large parametric models. This begs the question: What really matters for generalization in visual imitation?

An obvious answer is visual representation – generalizing to diverse visual environments should require powerful representation learning. Prior work in computer vision [16, 7, 8, 5, 4] have shown that better representations significantly improve downstream performance for tasks such as image classification. However, in the case of robotics, evaluating the performance of visual representations is quite

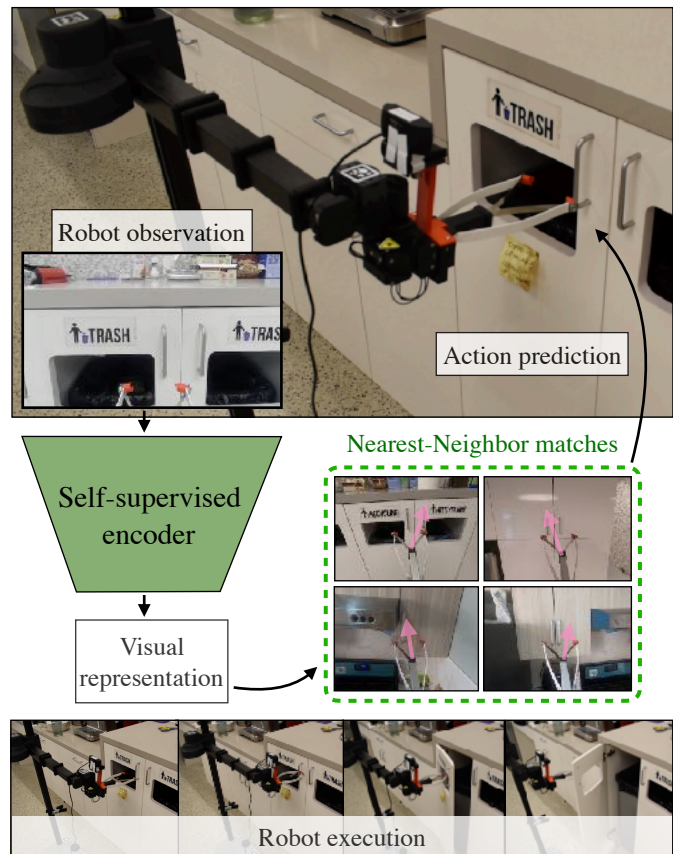*The first two authors contributed equally to this work.

Fig. 1. Consider the task of opening doors from visual observations. VINN, our visual imitation framework first learns visual representations through self-supervised learning. Given these representations, non-parametric weighted nearest neighbors from a handful of demonstrations is used to compute actions, which results in robust door-opening behavior.

complicated. Consider behavior cloning [43], one of the simplest methods of imitation. Standard approaches in behavior cloning fit convolutional neural networks on a large dataset of expert demonstrations using end-to-end gradient descent. Although powerful, such models conflate two fundamental problems in visual imitation: (a) representation learning, i.e. inferring information-preserving low-dimensional embeddings from high-dimensional observations and (b) behavior learning, i.e. generating actions given representations of the environment state. This joint learning often results in large dataset requirements for such techniques.

One way to achieve this decoupling is to use representation

modules pre-trained through standard proxy tasks such as image classification, detection, or segmentation [35]. However, this relies on large amounts of labelled human data on datasets that are often significantly out of distribution to robot data [6]. A more scalable approach is to take inspiration from recent work in computer vision, where visual encoders are trained using self-supervised losses [8, 7, 16]. These methods allow the encoders to learn useful features of the world without requiring human labelling. There has been recent progress in vision-based Reinforcement Learning (RL) that improves performance by creating this explicit decoupling [41, 48]. Visual imitation has a significant advantage over RL settings: learning visual representations in RL is further coupled with challenges in exploration [47], which has limited its application in real-world settings due to poor sample complexity.

In this work we present a new and simple framework for visual imitation that decouples representation learning from behavior learning. First, given an offline dataset of experience, we train visual encoders that can embed high-dimensional visual observations to low-dimensional representations. Next, given a handful of demonstrations, for a new observation, we find its associated nearest neighbors in the representation space. For our agent's behavior on that new observation, we use a weighted average of the nearest neighbors' actions. This technique is inspired by Locally Weighted Regression [3], where instead of operating on state estimates, we operate on self-supervised visual representations. Intuitively, this allows the behavior to roughly correspond to a Mixture-of-Experts model trained on the visual demonstrations. Since nearest neighbors is non-parametric, this technique requires no additional training for behavior learning. We will refer to our framework as Visual Imitation through Nearest Neighbors (VINN).

Our experimental analysis demonstrates that VINN can successfully learn powerful representations and behaviors across three manipulation tasks: Pushing, Stacking, and Door Opening. Surprisingly, we find that non-parametric behavior learning on top of learned representations is competitive with end-to-end behavior cloning methods. On offline MSE metrics, we report results on par with competitive baselines, while being significantly simpler. To further test the real-world applicability of VINN, we run robot experiments on opening doors using 71 visual demonstrations. Across a suite of generalization experiments, VINN succeeds 80% on doors present in the demonstration dataset and 40% on opening the door in novel scenes. In contrast, our strongest baselines have success rates of 53.3% and 3.3% respectively.

To summarize, this paper presents the following contributions. First, we present VINN, a novel yet simple to implement visual imitation framework that derives non-parametric behaviors from learned visual representations. Second, we show that VINN is competitive to standard parametric behavior cloning and can outperform it on a suite of manipulation tasks. Third, we demonstrate that VINN can be used on real robots for opening doors and can achieve high generalization performance on novel doors. Finally, we extensively ablate

over and analyze different representations, amount of training data, and other hyperparameters to demonstrate the robustness of VINN.

II. RELATED WORK

A. Imitation via Cloning

Imitation learning is frequently used to learn skills and behaviors from human demonstrations [31, 27, 26, 42]. In the context of manipulation, such techniques have successfully solved a variety of problems in pushing, stacking, and grasping [52, 54, 2, 19]. Behavioral Cloning (BC) [43] is one of the most common techniques. If the agent's morphology or viewpoint is different than the demonstrations', the model needs to involve techniques such as transfer learning to resolve this domain gap [40, 36]. To close this unintended domain gap, [52] has used tele-operation methods, while [39, 49] have used assistive tools. Using assistive tools provides us the benefit of being a able to scalably collect diverse demonstrations. In this paper, we follow the DemoAT [49] framework to collect expert demonstrations.

B. Visual Representation Learning

In computer vision, interest in learning a good representation has been longstanding, especially when labelled data is rare or difficult to collect [7, 8, 16, 5]. This large class of representation learning techniques aim to extract features that can help other models improve their performance in some downstream learning tasks, without needing to explicitly learn a label. In such tasks, first a model is trained on one or more pretext tasks with this unlabeled dataset to learn a representation. Such tasks generally include instance invariance, or predicting some image transformation parameters (e.g. rotation and distortion), patches, or frame sequence [15, 10, 9, 28, 7, 8, 46]. In representation learning, the performance of the model on the pretext task is usually disregarded. Instead, the focus is on the input domain to representation mapping that these models have learned. Ideally, to solve such pretext tasks, the pretrained model may have learned some useful structural meaning and encoded it in the representation. Thus, intuitively, such a model can be used in downstream tasks where there is not enough data to learn this structural meaning directly from the available task-relevant data. Unsupervised representation learning, in works such as [7, 8, 16, 5, 4, 12], has shown impressive performance gains on difficult benchmarks since they can harness a large amounts of unlabelled data unavailable in task-specific datasets.

Recently, interest in unsupervised or semi-supervised representation learning technique has grown within robotics [24] due to the availability of unlabeled data and its effectiveness in visual imitation tasks [50, 51]. We follow a BYOL-style [16] self-supervised representation learning framework in our experiments.

C. Non-parametric Control

Non-parametric models are those, which instead of modeling some parameters about the data distribution, tries to
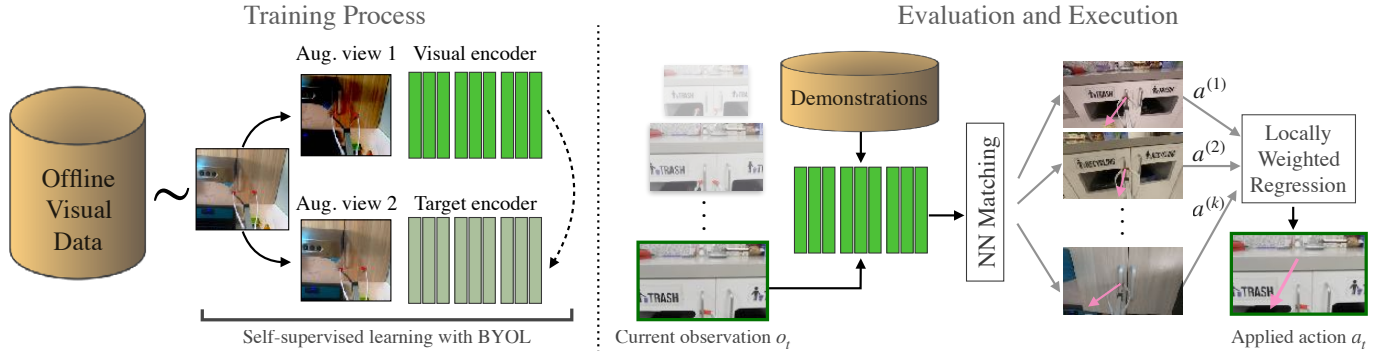
Fig. 2. Overview of our VINN algorithm. During training, we use offline visual data to train a BYOL-style self-supervised model as our encoder. During evaluation, we compare the encoded input against the encodings of our demonstration frames to find the nearest examples to our query. Then, our model's predicted action is just a weighted average of the associated actions from the nearest images.

express it in terms of previously observed training data. Non-parametric models are significantly more expressive, but as a downside to this, they usually require a large number of training examples to generalize well. A popular and simple example of non-parametric models is Locally Weighted Learning (LWL) [3]. LWL is a form of instance-based, non-parametric learning that refers to algorithms whose response to any query is a weighted aggregate of similar examples. Simple nearest neighbor models are an example of such learning, where all weight is put on the closest neighbor to the input point. Nearest neighbor methods have been successfully used in previous works for control tasks [23] More sophisticated, k-NN algorithms base their predictions on an aggregate of the nearest k points [1].

Uses of LWL based methods in supervised learning, robotics, and reinforcement learning is quite old. In works like [38, 45], effectiveness of LWL algorithms like k-nearest neighbor has shown competitive success in difficult, high dimensional tasks like classifying the miniImageNet. LWL has also shown success for robotic control problems [3], although it requires an accurate state-estimator to obtain low-dimensional states. In [22, 32, 33], elements of non-parametric learning is weaved into the reinforcement learning algorithms to create models which can adjust their complexity based on the amount of available data. Finally, in works like [37] non-parametric k-Nearest Neighbor regression based Q-functions are shown to give a good approximation of the true Q function under some theoretical assumptions. Our work, VINN, draws inspiration from the simplicity of LWL and demonstrates the usefulness of this idea by using Locally Weighted Regression in challenging visual robotic tasks.

III. APPROACH

In this section, we describe the components of our algorithms and how they fit together to create VINN. As seen in Fig. 2, VINN consists of two parts: (a) training an encoding network on offline visual data, and (b) querying against the provided demonstrations for a nearest-neighbor based action prediction.

A. Visual Representation Learning

Given an offline dataset of visual experience from the robot, we first learn a visual representation embedding function. In this work, we use two key insights for learning our visual representation: first, we can learn a good vision prior using existing large but unrelated real world datasets, and then, we can fine-tune starting from that prior using our demonstration dataset, which is small but relevant to the task at hand.

For the first insight, whenever possible, we initialize our models from an ImageNet-pretrained model. Such models are provided with the PyTorch [30] library that we use and can be achieved by simply adding a single parameter to the model initialization function call.

Then, we use self supervised learning and train this visual encoder on the all the frames in our offline training dataset. In this work, we use Bootstrap Your Own Latent (BYOL) [16] as the self-supervision objective. As illustrated in Fig. 2, BYOL uses two versions of the same encoder network: one normally updating online network, and a slow moving average of the online network called the target network. The BYOL self-supervised loss function tries to reduce the discrepancy in the two heads of the network when they are fed with differently augmented version of the same image. Although we use BYOL in this work, VINN can also work with other self-supervised representation learning methods [7, 8, 5, 4] (Table III).

In practice, we initialize both the BYOL online and target networks with an ImageNet-pretrained encoder. Then, using the BYOL objective, we finetune them to better fit our image distribution. Once the self-supervised training is done, we encode all our training demonstration frames with the encoder to obtain a set of their embeddings, E.

B. k-Nearest Neighbors Based Locally Weighted Regression

The set of embeddings E given by our encoder holds compact representations of the demonstration images. Thus, during test time, given an input we search for demonstration frames with similar features. We find the nearest neighbors of the encoded input e from the set of demonstration embeddings, E. In Fig. 3, we see that these nearest neighbors are visually
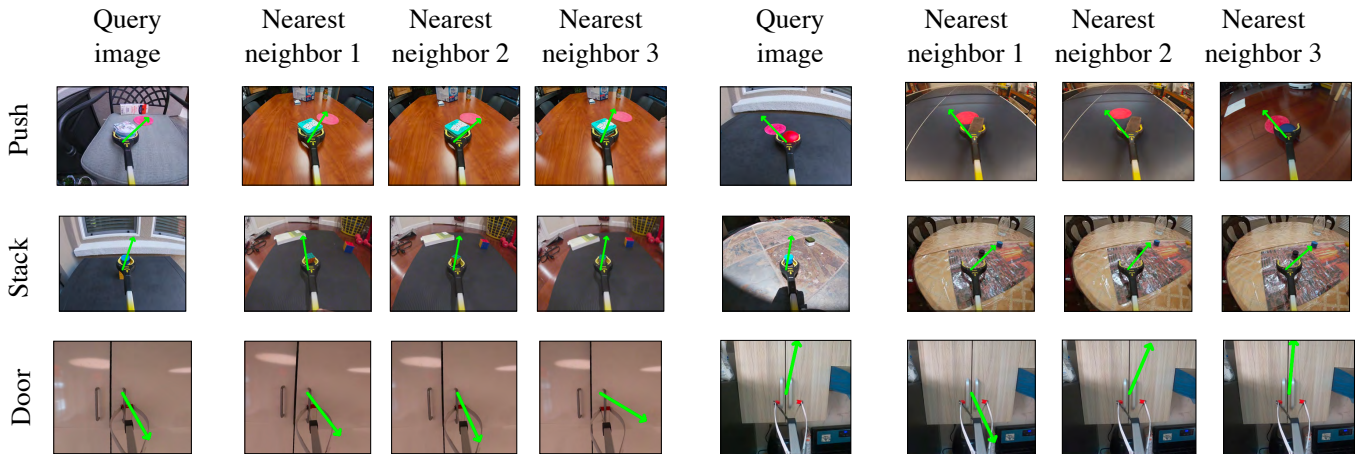
Fig. 3. Nearest neighbor queries on the encoded demonstration dataset; the query image is on the first column, and the found nearest neighbors are on the next three columns. The associated action is shown with a green arrow. The bottom right set of nearest neighbors demonstrates the advantage of performing a weighted average over nearest neighbors' actions instead of copying the nearest neighbor's action.

similar to the query image. Our algorithm implicitly assumes that a similar observation must result in a similar action. Thus, once we have found the k nearest neighbors of our query, we set the next action as an weighted average of the actions associated with those k nearest neighbors.

Concretely, this is done by performing nearest neighbors search based on the distance between embeddings: $\|e - e^{(i)}\|_2$, where $e^{(i)}$ is the ith nearest neighbor. Once we find the k nearest neighbors and their associated actions, namely $(e^{(1)}, a^{(1)}), (e^{(2)}, a^{(2)}), \cdots, (e^{(k)}, a^{(k)})$, we set the action as the Euclidean kernel weighted average [3] of those examples' associated actions:

$$\hat{a} = \frac{\sum_{i=1}^{k} \exp\left(-\|e - e^{(i)}\|_2\right) \cdot a^{(i)}}{\sum_{i=1}^{k} \exp\left(-\|e - e^{(i)}\|_2\right)}$$

In practice, this turns out to be the average of the observations' associated actions weighted by the SoftMin of their distance from the query image in the embedding space.

C. Deployment in real-robot door opening

For our robotic door opening task, we collect demonstrations using the DemoAT [49] tool. Here, a reacher-grabber is mounted with a GoPro camera to collect a video of each trajectory. We pass the series of frames into a structure from motion (SfM) method which outputs the camera's location in a fixed frame [29]. From the sequence of camera poses, which consist of coordinate and orientation, we extract translational motion which becomes our action. To extract the gripper state, we train a gripper network that outputs a distribution over four classes (open, almost open, almost closed, closed), which represent various stages of gripping. Then, we feed these images and their corresponding actions into our imitation learning method.

To train our visual encoders, we train ImageNet-pretrained BYOL encoders on individual frames in our demonstration dataset without action information. This same dataset with

action information serves as the demonstration dataset for the k-NN based action prediction. Note that although we use task-specific demonstrations for representation learning, our framework is compatible with using other forms of unlabelled data such offline datasets [17, 14] or task-agnostic play data [50].

To execute our door-opening skill on the robot, we run our model on a closed loop manner. After resetting the robot and the environment, on every step, we retrieve the robot observation and query the model with it. The model returns a translational action $\hat{a}$ as well as the gripper state g, and the robot moves $c \odot \hat{a}$ where the vector c is a hyper-parameter with each element < 1 to mitigate our SfM model's inaccuracies and improve transfer from human demonstrations to robot execution. In addition, for nearest neighbor based methods, we have hyper-parameters that map the floating value g into a gripper state which was tuned per experiment.

IV. EXPERIMENTAL EVALUATION

In the previous sections we have described our framework for visual imitation, VINN. In this section, we seek to answer our key question: how well does VINN imitate human demonstrations? To answer this question, we will evaluate both on offline datasets and in closed-loop real-robot evaluation settings. Additionally, we will probe into the generalization with few demonstrations ability of VINN in settings where imitation algorithms usually suffer.

A. Experimental Setup

We conduct two different set of experiments: the first on the offline datasets for Pushing, Stacking and Door-Opening and the second on real-robot door opening.

a) Offline Visual Imitation Datasets: Data for Pushing and Stacking tasks are taken from [49]. The goal in the pushing task is to slide an object on a surface into a red circle. In the stacking task, the goal is to grasp an object present in the scene and move it on top of another object also in the scene,
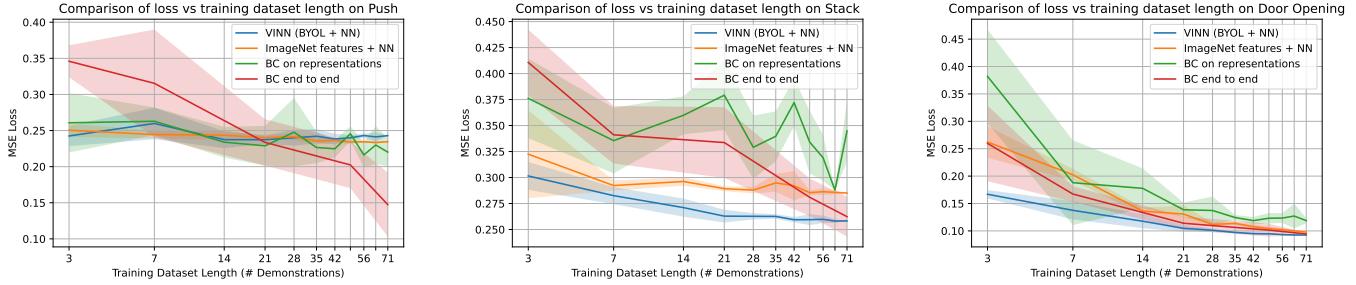
Fig. 4. Mean Squared Error for the Pushing, Stacking and Door Opening (left to right) datasets of different algorithms trained on subsamples of the original dataset. End-to-end behavior cloning initialized with ImageNet-trained features perform as well as VINN for larger datasets, but fixed representation based methods outperforms it largely on small datasets.

and release. To avoid confusion, in the expert demonstrations for stacking, the closest object is always placed on top of the distant object. The action labels are end-effector movements, which in this case is the translation vector in between the current frame and the subsequent one. In each case, there are a diverse set of backgrounds and objects that make up the scene and the task, making the tasks difficult.

For Door Opening, data is collected by 3 data-collectors in their kitchens. This amounts to a total of 71 demonstrations for training and 21 demonstrations for testing. We normalize all actions from the dataset to account for scale ambiguity from SfM. For all three tasks, we calculate the MSE loss between the ground truth actions and the actions predicted by each of the methods. Note that the number of demonstrations collected for this Door Opening task is an order of magnitude smaller than the ones used for Stacking and Pushing, which contain around 750 and 930 demonstrations respectively. To understand the performance on the various model in low data settings, we create subsampled Pushing and Stacking datasets containing 71 demonstrations on each for training and 21 for testing. This subsampling makes all three our datasets have the same size.

b) Closed-loop control: We conduct our robot experiments on a loaded cabinet door opening task (see Fig. 1), where the goal of the robot is to grab hold of the cabinet handle and pull open the cabinet door. We use the Hello-Robot Stretch [21] for this experiment. When evaluations start, the arm resets to ≈ 0.15 meters away from the cabinet door, with a random lateral translation within 0.05 meters parallel to the cabinet to evaluate generalization to varying starting states.

B. Baselines

We run our experiments for baseline comparison using the following methods:
- *Random Action:* In this baseline, we sample a random action from the action space.
- *Open Loop:* We find the maximum-likelihood open loop policy given all our demonstration, which is the average action $\bar{a}(t)$ over all actions $a_i(t)$ seen in the dataset at timestep t. In a Bayesian sense, if standard behavioral cloning is trying to approximate $p(a \mid s)$, this model is trying to approximate $p(a \mid t)$.

- *Behavioral Cloning (BC) end to end:* We train a ResNet-50 model with augmentated demonstration frames similar to [43, 49]. We initialize the model with weights derived from ImageNet pretraining.
- *BC on Representations (BC-rep):* We use a self-supervised BYOL model to extract the encoding of each of our demonstration frames, and perform behavioral cloning on top of the representations. This baseline is similar to [50] and performs better than end-to-end BC on the real robot (Table I).
- *Implicit Behavioral Cloning:* We train Implicit BC [13] models on the tasks, modifying the official code.
- *ImageNet features + NN:* Instead of self-supervision, here we use the image representation generated by a pretrained ImageNet encoder akin to [6]. The difference between this baseline and our method is simply forgoing the finetuning step on our dataset. This baseline highlights the importance of self-supervised pre-training on the domain related dataset.
- *Self-supervised learning method + NN:* This is our method; we compare three different ways of learning self-supervised representations features from our dataset – BYOL [16], SimCLR [7], and VICReg [4], starting from an ImageNet pretrained ResNet-50, and then we use locally weighted regression to find the action.

C. Training Details

Each encoder network used in this paper follows the ResNet-50 architecture [18] with the final linear layer removed. Unless specified otherwise, we always initialize the weights of the ResNet-50 encoder with a pretrained model on ImageNet dataset. For VINN, we train our self-supervised encodings with the BYOL [16] loss. For standard end-to-end BC, we replace the last linear layer with a three-layer MLP and train it with the MSE loss. For BC-rep, we freeze the encoding network to the weights trained by BYOL on our dataset, and train just the final layers with the MSE loss. Additionally, for all visual learning, we use random crop, random color jitter, random grayscale augmentations and random blurring. We trained the self-supervised finetuning methods for 100 epochs on all three datasets.

D. How does VINN Perform on Offline Datasets?

For our first evaluation, we compare our method against the baselines on their Mean-Squared Error loss for the Pushing, Stacking, and Door-Opening tasks in Fig. 4. To understand the impact of the training dataset size on the algorithms, we train models on multiple subsamples of different sizes from each dataset. We see that while end-to-end Behavioral Cloning starting from pretrained ImageNet representations can be better with a large amounts of training demonstrations, Nearest Neighbor methods are either competitive or better performing in low data settings.

On the Stacking and Door-Opening tasks, VINN is significantly better when the number of training demonstrations are small (< 20). While on the Pushing task, we notice that the task might be too difficult to solve with small number of demonstrations. One reason for this is that BYOL might not be able to extract the most relevant representations for this task. Further experiments in Table III show that using other forms of self-supervision such as VICReg can significantly improve performance on this task. Overall, these experiments supports our hypothesis that provided with good representations, nearest-neighbor techniques can provide a competitive alternative to end-to-end behavior cloning.

E. How does VINN Perform on Robotic Evaluation?

Next, we run VINN and the baselines on our real robot environment. In this setting, our test environment comprises of the same three cabinets where training demonstrations were collected presented without any visual modifications. For each of our models, we run 30 rollouts with the robot in the real world with three different cabinets. On each rollout, the starting position of the robot is randomized as detailed in (Sec. IV-A). In Table I, we show the percentage of success from the 30 rollouts of each model, where we record both the number of time the robot successfully grasped the handle, as well as the number of time it fully opened the door.

TABLE I
SUCCESS RATE OVER 30 TRIALS (10 TRIALS ON THREE CABINETS EACH) ON THE ROBOTIC DOOR OPENING TASK.

Method	Handle grasped	Door opened
BC (end to end)	0%	0%
BC on representations	56.7%	53.3%
Imagenet features + NN	20%	0%
VINN (BYOL + NN)	**80%**	**80%**

As we see from Table I, VINN does better than all BC variants in successfully opening the cabinet door when there is minimal difference between the test and the train environments. Noticeably, it shows that depending on self-supervised features on augmented data make the models much more robust. BC, as an end-to-end parameteric model, does not have a strong prior on the actions if the robot makes a wrong move causing the visual observations to quickly goes out-of-distribution [34]. On the other hand, VINN can recover up to certain degree of deviation using the nearest neighbor prior,

since the translation actions typically tend to re-center the robot instead of pushing it further out of distribution.

F. To What Extent does VINN Generalize to Novel Scenes?

To test generalization of our robot algorithms to novel scenes in the real world, we modified one of our test cabinets with various levels of occlusion. We show frames from a sample rollouts in each environment in Fig. 5, which also shows the cabinet modifications.

TABLE II
SUCCESS RATE OVER 10 TRIALS ON ROBOTIC DOOR OPENING WITH VISUAL MODIFICATIONS ON ONE CABINET DOOR.

Modification	BC-rep	VINN (ours)
Baseline (no modifications)	**90**%	80%
Covered signs and handle	10%	**70%**
Covered signs, handle, and one bin	0%	**50%**
Covered signs, handle, and both bins	0%	0%

In Table II, we see that VINN only completely fails when all the visual landscape on the cabinet is occluded. This failure is expected, because without coherent visual markers, the encoder fails to convey information, and thus the k-NN part also fails. Even then, we see that VINN succeeds at a higher rate even with significant modifications to the cabinet while BC-rep fails completely.

Over all the real robot experiments, we find the following phenomenon: while a good MSE loss is not sufficient for a good performance in the real world, the two are still correlated, and a low MSE loss seems to be necessary for good real world performance. This observation let us test hypotheses offline before deploying and testing them in a real robot, which can be time-consuming and expensive. We hypothesize that this gap between performance on the MSE metric (Table III) and real world performance (Table I, II) comes from variability in different models' ability to perform well in situations off the training manifold, where they may need to correct previous errors.

G. How Important are the Design Choices Made in VINN for Success?

VINN comprises of two primary components, the visual encoder and the nearest-neighbor based action modules. In this section, we consider some major design choices that we made for each of them.

a) Choosing the Right Self-supervision: While we use a BYOL-based self-supervised encoding in our algorithm, there are multiple other self-supervised methods such as SimCLR and VICReg [7, 4]. On a small set of experiments we noticed similar MSE losses compared to SimCLR [7] and VICReg [4]. From Table III, we see that BYOL does the best in Door-Opening and Stacking, while VICReg does better in Pushing. However, we choose BYOL for our robot experiments since it requires less tuning overall.
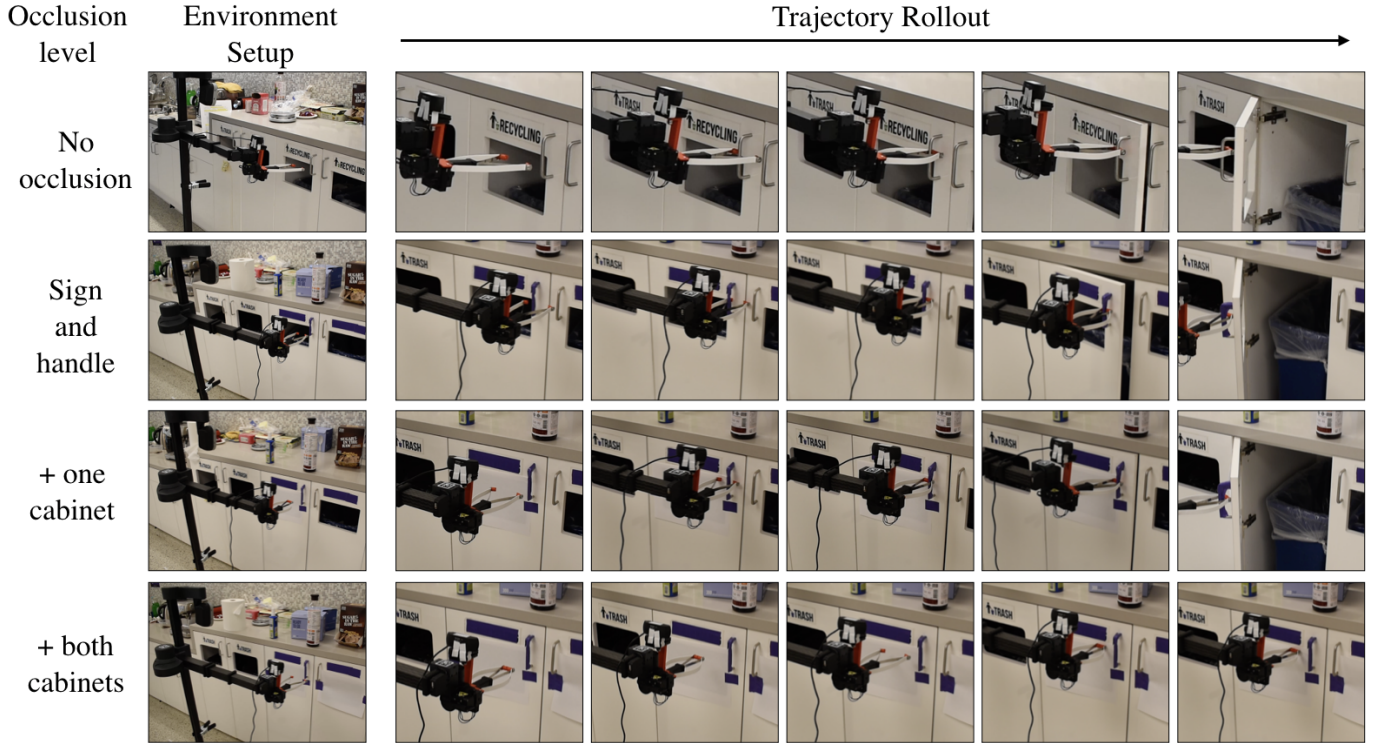
Fig. 5. Sample frames from the rollouts from our model on the real robot experiments, with artificial occlusions added to the cabinet to test generalization. Under the maximum occlusion, our model fails to ever open the cabinet door, while in all other cases, the robot is able to succeed (Table II.)

TABLE III

TEST MSE ($\times 10^{-1}$) ON PREDICTED ACTIONS FOR A SET OF BASELINE METHODS AND ABLATIONS. STANDARD DEVIATIONS, WHEN REPORTED, ARE OVER THREE RANDOMLY INITIALIZED RUNS.

			No Pretraining		With ImageNet Pretraining				
Tasks	Random	Open Loop	Implicit BC	BYOL + NN	BC-Rep	VINN (BYOL + NN)	VICREG + NN	SimCLR + NN	ImageNet + NN
Door Opening	6.34	2.27	1.8	1.52	1.19 ± 0.05	0.92	1.05	0.95	0.98
Stacking	6.13	2.83	7.1	2.82	3.45 ± 0.29	2.58	2.74	2.63	2.85
Pushing	6.15	2.12	5.6	2.43	2.20 ± 0.20	2.43	1.50	2.21	2.35

b) Ablating Pretraining and Fine-tuning: Another large gain in our algorithm is achieved by initializing our visual encoders with a network trained on ImageNet. In Table III, we also show MSE losses from models that resulted from ablating this components of VINN. Removing this component achieves the column BYOL + NN (No Pretraining), which performs much worse than VINN. Similarly, the success of VINN depends on the self-supervised fine-tuning on our dataset, ablating which results in the model shown in ImageNet + NN column of Table III. This model performs only slightly worse than VINN on the MSE metric. However, in Table I, we see that this model performs poorly on the real world. These ablations show that the performance of our locally weighted regression based policy depends on the quality of the representation, where a good representation leads to better nearest neighbors, which in turn lead to a better policy both offline and online.

c) Performing Implicit instead of Explicit Imitation: Moving away from the explicit forms of imitation where the models try to predict the actions directly, we run baselines with Implicit Behavioral Cloning (IBC) [13]. As we see on Table III, this baseline fails to learn behaviors significantly better than the random or open loop baselines. We believe this is caused by two reasons. First, the implicit models have to model the energy for the full space (action space $\times$ observation space), which requires more data than the few demonstrations that we have in our datasets. Second, the official implementation of IBC supports $[-1, 1]^3$ as the action space instead of its much smaller subspace of normalized 3d vectors S^2. This much larger action space, over which IBC tried to model the action, might have resulted in worse performance for IBC. While VINN makes the implicit assumption that the locally-weighted average of valid actions also yield a valid action, it can be freely projected to any relevant space without further

processing, which makes it more flexible.

d) Learning a Parametric Policy on Representations: Our Behavioral Cloning on representations (BC-Rep) baseline in all our experiments (Sec. IV) show the performance of a baseline where we use learned representations to learn a parametric behavioral policy. In the MSE losses (Table III) and real world experiments (Table I, II.) This is the baseline that achieves the closest performance to VINN. However, the difference between BC-rep and VINN becomes more pronounced as the gap between training and test domain or the policy horizon grows. These experimental results indicate that using a non-parametric policy may be enabling us to be robust to out-of-distribution samples.
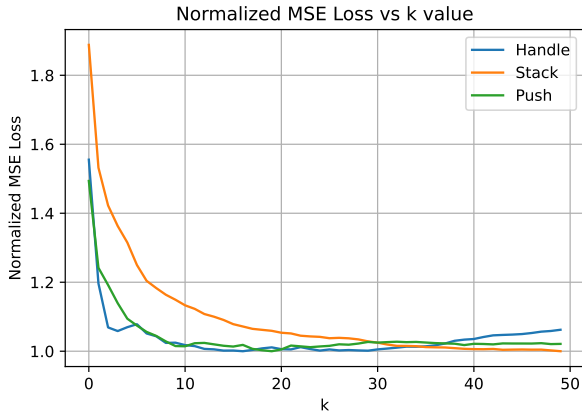

Fig. 6. Value of k in the k-nearest neighbor weighted regression in VINN vs normalized MSE loss achieved by the model.

e) Choosing the Right k for k-Nearest Neighbors: Finally, in VINN, we study the effect of different values of k for the k-NN based locally weighted controller. This parameter is important because with too small of a k, the predicted action may stop being smooth. On the other hand, with too large of a k, unrelated examples may start influencing the predicted action. By plotting our model's normalized MSE loss in the validation set against the value of k in Fig. 6, we find that around 10, k seems ideal for achieving low validation loss while averaging over only a few actions. Beyond $k = 20$, we didn't notice any significant improvement to our model from increasing k.

H. Computational Considerations

While the datasets we used for our experiments were not large, we recognize that our current nearest neighbor implementation is a $O(n)$ algorithm dependant linearly on the size of the training dataset with a naive algorithm. However, we believe VINN to be practical, since firstly, it was designed mostly for the small demonstration dataset regime where $O(n)$ is quite small, and secondly, this search can be sped up with a compiled index beyond the naive method using open-source libraries such as FAISS [20] which were optimized to run nearest neighbor search on the order of billion examples [25].

Currently, our algorithm takes ≈ 0.074 seconds to encode an image, and ≈ 0.038 seconds to perform nearest neighbors regression, which is only a small speed penalty for the robotic tasks we consider.

V. LIMITATIONS AND FUTURE WORK

In this work we proposed VINN, a new visual imitation framework that decouples visual representation learning from behavior learning. Although this decoupling improves over standard visual imitation methods, there are several avenues for future work. First, there is still some remaining hurdles to generalizing to a new scene, as seen in Sec. IV-F, where our model fails when all large, recognizable markers are removed from the scene. While our NN-based action estimation lets us add new demonstrations easily, we cannot easily adapt our representation to such drastic changes in scene. An incremental representation learning algorithm has great potential to improve upon that. Second, our self-supervised learning is currently done on task related data, while ideally, if the dataset is expansive enough, task agnostic pre-training should also give us good performance [50]. Finally, although our framework focuses on a single-task setting, we believe that learning a joint representation for multiple tasks could reduce the overall training overhead while being just as accurate.

REFERENCES

[1] David W. Aha and Steven L. Salzberg. Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks. In P. Cheeseman and R. W. Oldford, editors, *Selecting Models from Data*, pages 321–328, New York, NY, 1994. Springer New York. ISBN 978-1-4612-2660-4.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009.

[3] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. *Lazy learning*, pages 11–73, 1997.

[4] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[6] Bryan Chen, Alexander Sax, Gene Lewis, Iro Armeni, Silvio Savarese, Amir Zamir, Jitendra Malik, and Lerrel Pinto. Robust policies via mid-level visual representations: An experimental study in manipulation and navigation. *arXiv preprint arXiv:2011.06698*, 2020.

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[8] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[9] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2016.

[10] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks, 2015.

[11] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[12] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. *arXiv preprint arXiv:2104.14548*, 2021.

[13] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *arXiv preprint arXiv:2109.00137*, 2021.

[14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[17] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv e-prints*, pages arXiv–2006, 2020.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[19] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50 (2):1–35, 2017.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[21] Charles C Kemp, Aaron Edsinger, Henry M Clever, and Blaine Matulevich. The design of stretch: A compact, lightweight mobile manipulator for indoor human environments. *arXiv preprint arXiv:2109.10892*, 2021.

[22] Minwoo Lee and Charles W Anderson. Robust reinforcement learning with relevance vector machines. *Robot Learning and Planning (RLP 2016)*, page 5, 2016.

[23] Elman Mansimov and Kyunghyun Cho. Simple nearest neighbor policy method for continuous control tasks, 2018. URL https://openreview.net/forum?id=ByL48G-AW.

[24] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020.

[25] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. A survey of product quantization. *ITE Transactions on Media Technology and Applications*, 6 (1):2–10, 2018.

[26] Andrew N Meltzoff and Keith Moore. Newborn infants imitate adult facial gestures. *Child development*, 1983.

[27] Andrew N Meltzoff and M Keith Moore. Imitation of facial and manual gestures by human neonates. *Science*, 1977.

[28] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification, 2016.

[29] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion, 2017.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[31] Jean Piaget. *Play, dreams and imitation in childhood*, volume 25. Routledge, 2013.

[32] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control, 2017.

[33] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control, 2018.

[34] Stephane Ross, Geoffrey Gordan, and Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *arXiv preprint arXiv:1011.0686*, 2010.

[35] Alexander Sax, Jeffrey O Zhang, Bradley Emi, Amir

Zamir, Silvio Savarese, Leonidas Guibas, and Jitendra Malik. Learning to navigate using mid-level visual priors. *arXiv preprint arXiv:1912.11121*, 2019.

[36] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[37] Devavrat Shah and Qiaomin Xie. Q-learning with nearest neighbors, 2018.

[38] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning, 2017.

[39] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

[40] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *ICLR*, 2017.

[41] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[42] Michael Tomasello, Sue Savage-Rumbaugh, and Ann Cale Kruger. Imitative learning of actions on objects by children, chimpanzees, and enculturated chimpanzees. *Child development*, 64(6):1688–1705, 1993.

[43] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[44] Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. Doorgym: A scalable door opening environment and baseline agent. *CoRR*, abs/1908.01887, 2019. URL http://arxiv.org/abs/1908.01887.

[45] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning, 2019.

[46] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018.

[47] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[48] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

[49] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv e-prints*, pages arXiv–2008, 2020.

[50] Sarah Young, Jyothish Pari, Pieter Abbeel, and Lerrel Pinto. Playful interactions for representation learning. *arXiv preprint arXiv:2107.09046*, 2021.

[51] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[52] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

[53] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[54] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei A. Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and imitation learning for diverse visuomotor skills. *CoRR*, abs/1802.09564, 2018. URL http://arxiv.org/abs/1802.09564.

A. VINN Pytorch Pseudocode

```python
def dist_metric(x,y):
    return(torch.norm(x-y).item())

def calculate_action(dist_list,k):
    action = torch.tensor([0.0,0.0,0.0])
    top_k_weights = torch.zeros((k,))
    for i in range(k):
        top_k_weights[i] = dist_list[i][0]
    top_k_weights = softmax(-1*top_k_weights)
    for i in range(k):
        action = torch.add(top_k_weights[i]
            * dist_list[i][1], action)
    return(action)

def calculate_nearest_neighbors(query_img,
    dataset, k):
    query_embedding = encoder(query_img)
    for dataset_index in
        range(len(dataset)):
        dataset_embedding,
            dataset_translation =
            dataset[dataset_index]
        distance =
            dist_metric(query_embedding,
            dataset_embedding)
        dist_list.append((distance,
            dataset_translation,
            dataset_path))
    dist_list = sorted(dist_list, key =
        lambda tup: tup[0])
pred_action = calculate_action(dist_list, k)
return pred_action
```

B. Network Architectures and Training Details

In this section, we will go over our implementation, network architectures, and training details for our various baselines.

a) Random Action: We sampled a 3-d vector from $[-1, 1]^3$, normalized it, and used it as our action for this baseline.

b) Open Loop: We computed the average action at frame t over all demonstrations from our dataset for this baseline for each frame number t.

c) Behavioral Cloning (end to end or from representations): For our parameterized model experiments, our encoding network is always a ResNet50, and our translation neural network is a three-layer MLP whose layer dimensions are $2048, 1024, 3$. Our gripper model is a linear layer that predicts four gripper states from the encoder network output. We train both models for 8000 epochs with a learning rate of 0.001 using the Adam optimizer. On an RTX8000, given the learned representations, it takes 12 minutes on the Door Opening dataset to train both MLPs for BC from representations. For training the BC end to end model until convergence, it takes us three hours in total.

d) Implicit Behavioral Cloning (IBC): We used the official Github repo for Implicit Behavioral Cloning [13] offline experiments. We modified their Push from Pixels task to fit 3-d vectors bounded within $[-1, 1]^3$. Unfortunately, we could not use the space of normal vectors since the current published version of the IBC code does not support constrained action spaces.

We trained the standard Dense ResNet model provided with the IBC repo for encoders, and IBC-with-DFO framework for sampling actions. It took us about 6 hours to train the models end-to-end on our datasets on an RTX 8000 GPU for 10,000 steps. For every hyperparameter, we use the defaults for the learning to push from pixels task that is included in the IBC repository.

We computed the MSE loss from this model by first sampling 256 actions with DFO optimization, as it's done in IBC, and choosing the action with the highest assigned value out of it.

e) VINN: For our BYOL-trained encoding network, we use a ResNet50 architecture, with the final linear for ImageNet classification replaced with an identity layer. We use the representation vector of size 2048. We fine-tune this network using BYOL for a 100 epochs on our demonstration datasets with the ADAM optimizer and a 3×10^{-4} learning rate. To train this BYOL on the Door Opening dataset for 100 epochs it took approximately 3.5 hours on a workstation with one Nvidia RTX8000.

C. Robot details

Fig. 7.   Hello Robot's Stretch [21], the robot model used in our experiments

We run all of our robots in the Hello Robot's Stretch [21]. This robot has a dexterous wrist with 3 DoF, a telescopic 1 DoF arm on which it is mounted, and an 1-DoF lift on which the arm is mounted. The base of the robot is also capable of rotation and lateral motion, which gives the robot's end-effector a full 6-DoF capability.

On each step, the translation model predicts $\Delta(x, y, z)$ for the gripper, which is converted to the movement in the robot's joints with an inverse kinematics model. This model takes into account simpler objectives like avoiding self-collisions, but does not model avoiding issues like environment collisions.

For the robot observations, we use a standard webcam mounted on top of the robot wrist using a custom 3-d printed mount. The image captured by the robot is streamed over the network to a machine running the VINN algorithm, which responds with the predicted robot action.

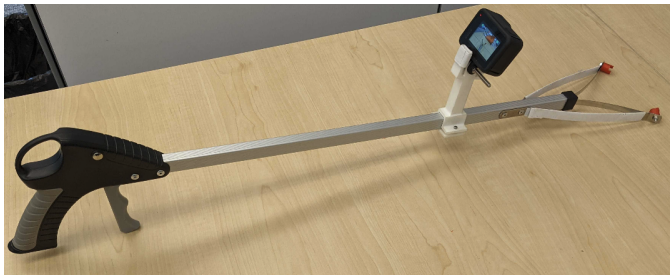*D. Demonstration Collection Details*

Fig. 8. Reacher grabber tool used for our demonstrations.

We use the DemoAT [49] framework for collecting our demonstrations. We use a simple reacher-grabber tool availabe at hardware shops or online, fitted with a GoPro camera to do capture our observation frames. An image of this is shown in Fig. 8,

Fig. 9. Modified grip on the robot and the reacher grabber.

We replaced the pads at the end of the robot gripper with simple 3-d printed nubs for easy resets of the robot, and we do the same on the reacher-grabber tool, as seen in Fig. 9.

To get visual observations, we mount a GoPro on top of the reacher grabber tool with a custom 3-d printed mount. We linearize the GoPro video in post-processing using `ffmpeg` to get rid of the wide-angle distortions, and extract the frames at one frame per second speed. Finally, using the extracted frames and the OpenSfM library, we reconstruct the 3-d movements between frames. We take the delta position changes between consecutive frames, and normalize them to get our actions.