TPTF: LEVERAGING GLOBAL RECEPTIVE FIELDS AND SPECTRAL FILTERS FOR VISUAL ROBOTIC MANIPULATION WITH TRANSPORTING TRANSFORMER NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Barış Bilgin Şenol
September 2025

TpTf: Leveraging Global Receptive Fields and Spectral Filters for Visual Robotic Manipulation with Transporting Transformer Networks
By Barış Bilgin Şenol
September 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Salih Özgür Öğüz(Advisor)

Can Alkan

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

TPTF: LEVERAGING GLOBAL RECEPTIVE FIELDS AND SPECTRAL FILTERS FOR VISUAL ROBOTIC MANIPULATION WITH TRANSPORTING TRANSFORMER NETWORKS

Barış Bilgin Şenol
M.S. in Computer Engineering
Advisor: Salih Özgür Öğüz
September 2025

Transformers have recently emerged as a powerful and versatile tool capable of capturing complex interactions among long-distance features, making them highly suitable for learning visual representations for robotic manipulation tasks. However, existing density estimation models, such as Transporter networks [1], rely on convolutional backbones that primarily process local information, requiring multiple convolutional stems to learn task-specific policies. We explore the potential of Transformer networks and alternative token mixing mechanisms for categorical density estimation and propose the Transporting Transformer networks for complex robotic pick-and-place tasks. Our approach employs a single encoder stem to leverage global features for learning both *pick* and *pick-conditioned place* policies. Building upon its enhanced capacity, we also introduce a novel training scheme for multi-task learning on the Ravens benchmark. The Transporting Transformer learns manipulation policies directly from visual observations without object-level assumptions, achieving improved performance through effective modeling of long-range spatial relationships. It maintains sample efficiency comparable to existing methods while demonstrating superior performance in both single-task and multi-task learning settings.

Keywords: Visual Robotic Manipulation, Categorical Density Estimation, Transporter Networks, Transformer Networks, Self-Attention, Object-Agnostic.

ÖZET

TPTF: TAŞIYICI DÖNÜŞTÜRÜCÜ AĞLARIYLA GÖRSEL ROBOTİK MANİPÜLASYON İÇİN KÜRESEL ALGI ALANLARI VE SPEKTRAL FİLTRELERİN KULLANILMASI

Barış Bilgin Şenol
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Salih Özgür Öğüz
Eylül 2025

Dönüştürücü ağlar, uzun mesafeli özellikler arasındaki karmaşık etkileşimleri yakalayabilen güçlü ve çok yönlü bir araç olarak son zamanlarda ortaya çıkmış ve robotik manipülasyon görevleri için görsel temsil öğrenimine son derece uygun hâle gelmiştir. Bununla birlikte, Taşıyıcı ağları [1] gibi mevcut yoğunluk tahmin modelleri, öncelikle yerel bilgiyi işleyen konvolüsyonel omurga yapılarına dayanmakta ve görev-özel politikaları öğrenmek için birden çok konvolüsyonel gövde gerektirmektedir. Bu çalışmada, Dönüştürücü ağlarının ve alternatif token karıştırma mekanizmalarının kategorik yoğunluk tahmini üzerindeki potansiyelini inceleyerek karmaşık robotik tut-ve-yerleştir görevleri için Taşıyıcı Dönüştürücü ağlarını önermekteyiz. Yaklaşımımız, hem *tut* hem de *tut-bağımlı yerleştir* politikalarını öğrenmek amacıyla küresel özelliklerden yararlanmak üzere tek bir kodlayıcı gövdesi kullanır. Artırılmış kapasitesine dayanarak, Ravens veri kümesi üzerinde çok-görevli öğrenme için yeni bir eğitim şeması da tanıtmaktayız. Taşıyıcı Dönüştürücü ağları, nesne-seviyesi varsayımlara ihtiyaç duymadan görsel gözlemlerden doğrudan manipülasyon politikalarını öğrenir ve uzun menzilli uzaysal ilişkilerin etkili modellenmesi sayesinde performansı artırır. Örnek verimliliğini mevcut yöntemlerle karşılaştırılabilir düzeyde tutarken, tek-görevli ve çok-görevli öğrenme ortamlarında üstün performans sergiler.

Anahtar sözcükler: Görsel Robotik Manipülasyon, Kategorik Yoğunluk Tahmini, Taşıyıcı Ağlar, Dönüştürücü Ağlar, Öz-Dikkat, Nesne-Agnostik.

Acknowledgement

I would like to express my deepest gratitude to Assistant Professor Özgür S. Öğüz, whose insightful guidance, patient mentorship, and constructive feedback have been indispensable throughout the course of this thesis.

My sincere thanks also go to Can Alkan and Gökberk Cinbiş, members of my thesis jury, for their valuable comments and suggestions that helped to refine and improve this work.

I am profoundly grateful to the Scientific and Technological Research Council of Turkey (TÜBİTAK) for their financial support, which made this research possible.

I owe an immeasurable debt of thanks to my mother, whose unconditional love, encouragement, and countless sacrifices have shaped every step of my life journey.

Finally, I owe an immeasurable debt of thanks to my partner and companion. Her unwavering love, encouragement, and constant presence have been a source of strength and joy from the very beginning of this journey. Thank you for always being there and for bringing a smile to even the most challenging days.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning manipulation policies from expert demonstrations can be achieved by generating action value maps directly from visual observations, offering the potential for generalizability as it requires no prior assumptions for task completion. Models employing transport mechanisms [1] exemplify this approach where they tackle the *pick-and-place* problem through a two-stage procedure: first, identifying an appropriate *picking* position; second, determining an optimal *placing* location based on the initial picking location. This *pick-conditioned placing* procedure enables the model to adapt its actions according to densely encoded environmental features directly extracted from perceptual data. Furthermore, cross-correlation layers allow the model to estimate the optimal placement angle, effectively "imagining" different placement outcomes in various rotations and selecting the angle that maximizes the likelihood of a successful *placing* action. By representing each action as a sequence of *picking* and *placing* primitives, the model can learn complex multi-step tasks through visual feedback about the current environmental state[1].

[1]This work has been submitted for publication to the Transactions on Machine Learning Research (TMLR) journal and is currently under review.

However, the efficacy of learning from demonstrations is limited by their reliance on high-quality demonstrations, which are often noisy, suboptimal, and conditioned on high-dimensional observations. Consequently, this limitation necessitates the development of robust imitation learning models that can accurately capture the underlying distribution in their policy, accommodating a diverse range of possible actions. Meanwhile, the Transformer architecture [2] emerged with its ability to capture long-range dependencies through self-attention and cross-attention mechanisms. Its versatility is further demonstrated by its successful adaptation from natural language processing [3] to other areas, including computer vision tasks such as image classification [4, 5], object detection and segmentation [6], and denoising [7].

The self-attention mechanism in the Transformer architecture is the key factor ensuring the success of these models. In this mechanism, each token's representation is updated by attending to all other tokens in the previous layer, effectively mixing tokens in each layer. This process is essential for retaining long-term information and gives Transformers an advantage over traditional models such as convolutional ones which rely on deep cascaded layers to increase its receptive field. However, attending to all tokens at each layer results in a complexity of $\mathcal{O}(N^2)$ with respect to the sequence length.

Two major challenges presented by the Transformer architecture are the computationally intensive multi-head attention modules and the need for a considerably larger dataset to surpass competing convolutional models. These factors significantly elevate the resources necessary for experimenting with these novel models within a reasonable time-frame, limiting the ability of many researchers to experiment with Transformers in a timely manner.

The increasing complexity and dataset requirements motivated numerous studies to explore approximate and alternative token mixing mechanisms for Transformers. Nyströmformer [8], Linformer [9], and Performer [10] propose methods to

approximate self-attention to decrease quadratic costs, while CvT [11] introduces convolutions in Transformers. On the other hand, MLP-Mixer [12], ResMLP [13], and gMLP [14] approaches replace self-attention layers with multi-layer perceptrons. Another line of work investigates using linear functions termed Lambda layers [15], and Fourier Transform [16, 17, 18] directly in place of self-attention layers. These approaches achieve comparable or improved results at reduced computational costs. Given Transformer's capacity to process noisy data through attention or its alternatives, it is a compelling candidate for perception-driven robotic manipulation applications, where data is typically scarce and noisy.

Building on these advances, we introduce the Transporting Transformer (TpTf) and its variant, the Spectral Transporting Transformer (SpTpTf). TpTf embeds an attention mechanism at its core, while SpTpTf incorporates a Fourier transform — both designed to tackle complex robotic manipulation tasks. We combine RGB-D images captured from the scene through orthographic projection and patchify them before feeding into our singular Transformer-based encoder to extract features. We further refine these features through upsampling encoder layers to generate value maps for *pick*, *place*, and rotation actions. This procedure eliminates redundant encoding of the scene with separate U-Net models as used in transport-based works. Our streamlined approach enables us to alleviate the $\mathcal{O}(N^2)$ computational complexity associated with self-attention layers, and create an end-to-end transporting model capable of handling *picking* and *pick-conditioned placing* simultaneously.

Overall, the key contributions of our work can be summarized as follows:

- We present a novel Transformer-based architecture specifically designed to process RGB-D orthographic projections. Leveraging a unified encoder backbone, the model simultaneously generates pixel-wise action-value maps for robotic pick and pick-conditioned place tasks.

- We develop a new training protocol that enables efficient simultaneous learning of multiple tasks, significantly reducing the total number of necessary training iterations.
- We examine the extent to which frequency-domain features extracted from spatial data can be effectively exploited for robotic manipulation using the SpTpTf variant.
- We provide comprehensive and qualitative analysis on various *pick-and-place* tasks to demonstrate that TpTf utilizes encoded features for effective long-range interactions between patches while remaining extensible.

We evaluate our models against baseline results on the Ravens benchmark (1.1) and conduct an ablation study to assess the effect of model size. The results demonstrate that our novel, unified single-encoder backbone enables TpTf and SpTpTf to surpass larger baseline models in both single-task and multi-task settings, while offering greater computational efficiency. These findings indicate that incorporating vanilla attention—or alternative token mixing mechanisms—enhances the transporting model's capacity to learn underlying policies and to generate accurate action-value maps.

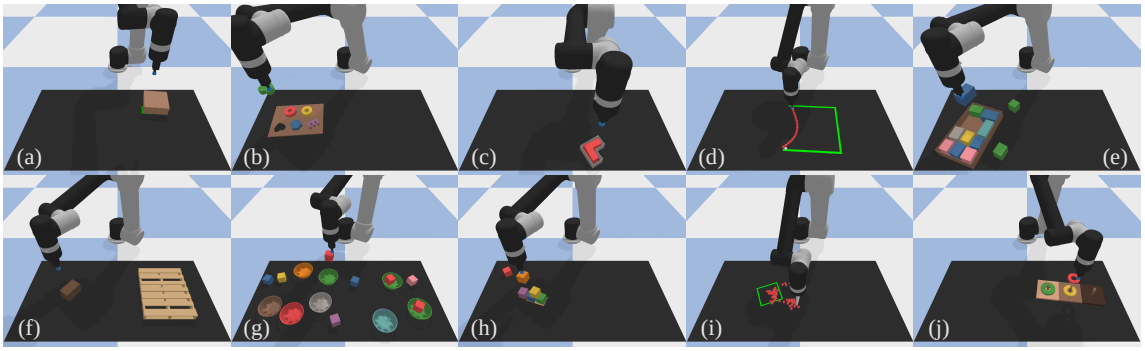

Figure 1.1: Transporting model performing robotic manipulation on the Ravens benchmark using *pick-and-place* primitives, where it predicts the most useful pick and place positions solely from visual input without relying on object-level assumptions. Target tasks include (a) aligning a box corner, (b) assembling kits, (c) inserting blocks, (d) manipulating rope, (e) packing boxes, (f) palletizing boxes, (g) placing red blocks into green bowls, (h) stacking blocks into a pyramid, (i) sweeping piles, and (j) playing Towers of Hanoi. The model infers the current state of the environment entirely from visual feedback.

Chapter 2

Related Work

2.1 Vision for Robotic Manipulation

The ability to manipulate the environment using sensory data is crucial for effectively leveraging environmental features. Traditional approaches in vision-based manipulation have relied on comprehensive object models, aligning segmented point cloud data gathered from the environment with these models [19, 20, 21, 22]. In contrast, advancements in deep learning have enabled novel object representations such as 6D object poses, segmentation maps, and class labels [23, 24, 25, 26, 27, 28], enhancing manipulation techniques. Recent progress has improved robustness and sample efficiency by incorporating object-centric assumptions, integrating techniques like object keypoints [29, 30, 31, 32], object detectors and pose estimators [33, 34, 35], feature embeddings [36, 37, 38], or dense descriptors [39, 40, 41]. Meanwhile, end-to-end models [42, 43, 44, 45] that directly map input observations to actions learn quickly and generalize well, they often require extensive datasets for training, which is limited and expensive in the context of robotic manipulation tasks.

2.2 Transporter Networks

Transporter Networks [1] propose an end-to-end architecture that preserves the spatial structure obtained from perceptual data by generating orthographic projections from point clouds and producing pick and pick-conditioned place confidence maps without relying on object-centric assumptions. This convolutional network leverages the inductive biases inherent in the preserved spatial structure, leading to significant improvements in sample efficiency. The authors introduced a new benchmark, Ravens-10, consisting of 10 distinct pick-and-place manipulation tasks. Subsequent studies quickly adapted and expanded this benchmark; for instance, by extending it with deformable object manipulation [46]. These adaptations demonstrated that Transporter Networks can seamlessly handle these additional tasks due to their architecture's lack of reliance on object-centric assumptions. Equivariant Transporter Networks [47] enhanced the model's sample efficiency by incorporating rotational equivariance into its pick and place networks using equivariant convolution layers. Meanwhile, FourTran [48] employed fiber space Fourier transformations to efficiently process rotational permutations, effectively extending the pick-and-place problem to 3D scenarios. CLIPort [49] combined Transporter Networks with a pretrained vision-language model, CLIP [50], for language-guided pick and place tasks. Building on this, LoHoRavens [51] utilized large language models to generate simpler step-by-step instructions that guide the CLIPort model in achieving long-horizon tasks.

2.3 Transformer Networks

2.3.1 Transformers

Transformer plays a pivotal role in modern state-of-the-art models, leveraging their multi-head attention mechanisms to capture long-range dependencies between sequences. Initially designed for machine translation tasks [2], transformers were quickly adapted for vision tasks, with Vision Transformer (ViT) [4] marking a significant milestone. ViT modifies the original transformer architecture to handle image data by dividing images into non-overlapping patches, flattening these patches, and then feeding them as input sequences into multi-head attention modules. The scaled dot-product attention mechanism is central to transformers' functionality.

2.3.2 Transformers in Vision

Building on the initial success of Vision Transformers, numerous models have emerged with innovative approaches to enhance its performance. DeiT [5] , for instance, introduces a transformer-specific distillation technique utilizing attention mechanisms, enabling the model to learn from a CNN-based teacher and acquire missing inductive biases. Another notable model is the Swin Transformer [6], which constructs a hierarchical architecture using multiple local windows for self-attention operations. Additionally, T2T-ViT [52] proposes to progressively tokenize input sequences before applying self-attention by restructuring them with overlapping regions to better capture local information between tokens. Moreover, the architecture was quickly adapted sub-fields of computer vision including image classification [53, 5], object detection and segmentation [54, 6, 55], denoising [7, 56], and super-resolution [57, 58]. Unlike traditional convolutional layers in ResNet [59], which is also utilized

in Transporter Networks, these networks employ attention-based processing for perceptual data.

2.3.3 Efficient Transformers

Another line of research focused on the self-attention mechanism as the limiting factor of Transformer's efficiency. These models aim to optimize the attention mechanism to reduce its computational complexity and memory requirements while still maintaining high accuracy. For instance, Linformer [9] applies projection to obtain a low-dimension representation of the keys and values matrices. This new low-rank decomposition alleviates the memory complexity issue. Performers [10] and Linear Transformers [60] reduce complexity through efficiently kernelizable attention mechanisms. Reformer [61] introduces locality sensitive hashing similarity to learn token clustering that allows better view of the input sequence while benefiting from the fixed hashing method. Sparse Transformers [62] make use of stride or dilation to reduce the number of attended features. CvT [11] introduces convolutional projections in self-attention layers.

2.3.4 Alternative Architectures

On the other hand, some studies proposed to completely replace the self-attention mechanism with different neural network architectures. ResMLP [13], gMLP [14], and MLP-Mixer [12] substitute multi-head attention with MLPs applied across feature or spatial channels. ConvMixer [63] adopts traditional convolutional layers. Fnet [16] employs Fourier Transform for natural language processing (NLP) tasks and demonstrates great efficiency benefits while maintaining competitive performance. GFNet [17] proposes global filter layers to learn in the frequency domain

by continuous forward and inverse transforms for visual recognition tasks. FrIT [18] applies Fractional Fourier Transform in the Transformer architecture for the hyperspectral imaging task. Another line of research has examined the direct replacement of self-attention with purely linear mechanisms, such as Lambda layers [15]. Other studies have examined replacing the self-attention mechanism with linear modules called Lambda layers [15]. These alternatives match or even surpass the original self-attention mechanism in performance, yet do so with significantly lower computational cost.

2.3.5 Transformers in Robotic Manipulation

The success of transformers in natural language and vision processing has extended to robotic manipulation tasks, including pick-and-place operations. PerAct [64] segments the environment into voxels and employs a Perceiver Transformer [65] for multi-task learning from demonstrations in robotic manipulation. In contrast, RVT [66] leverages dense feature descriptors of the explicit 3D representation of the environment to generate actions within that same space. Models like BC-Z [67] and BeT [68] aim to create generalizable imitation learning models. Additionally, Rt-1 [69] and Rt-2 [70] train their Robotic Transformer networks with not only manipulation demonstrations, but also general vision-language tasks such as visual question answering, incorporating large-scale web data for enhanced training.

Although Transformers achieve state-of-the-art results in many domains, they often require large models and extensive data to reach such performance. This raises the research question of whether the self-attention mechanism is truly indispensable, or whether simpler, parameter-efficient alternatives can achieve comparable results. Consequently, there remains ample room for improvement when the target task is constrained to limited training samples, as is typical in visual robotic manipulation.

Chapter 3

Methodology

3.1 Problem Statement

We investigate planar object rearrangement using a robotic arm by learning a policy π to determine subsequent actions in a series of motion primitives, based solely on visual observations $o_t \in \mathbb{R}^{H \times W \times C}$. The learned policy models two probability distributions: $p(\, a_{\text{pick}} \mid o_t \,)$ and $p(\, a_{\text{place}} \mid o_t,\, a_{\text{pick}} \,)$, using sequences of expert demonstrations such that $\pi(o_t \in \mathcal{O}) \rightarrow a_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}}) \in \mathcal{A}$. The action at step t comprises two end effector positions: $\mathcal{T}_{\text{pick}}$, the pixel coordinates (u, v) during the pick action a_{pick}, and $\mathcal{T}_{\text{place}}$, the pixel coordinates and rotation around the z-axis (u, v, θ) during the place action a_{place}. The sequence of consecutive actions a_t enables the robotic arm to solve a manipulation task by performing a series of pick and place actions.

3.2 Background

3.2.1 Transporter Network Overview

Transporter Networks propose a model architecture comprising three primary components: an attention arm that identifies a suitable region for grasping, a transport arm with key and query components that relocate the object to the target position by predicting the end effector position for placement (Figure 3.1). To obtain the pixel coordinates (u, v) for actions a_t, the attention arm models an action-value function $Q_{\text{pick}}((u, v) \mid o_t)$, which predicts the success rate of picking for individual pixels given the observation frame o_t. Conversely, the two components of the transport arm model an action-value function $Q_{\text{place}}(\mathcal{T}_i \mid o_t, \mathcal{T}_{\text{pick}}) = \Phi_{\text{query}}(o_t[\mathcal{T}_{\text{pick}}]) * \Phi_{\text{key}}(o_t)$, where the query and key components collectively generate dense features conducive to transportation. In a sense, these functions allow the network to model the underlying probability distributions required for transporting the target between two points. Specifically, the key component consumes the entirety of the observation frame o_t, while the query component processes a cropped and rotated region $(o_t[\mathcal{T}_{\text{pick}}])$ surrounding the $\mathcal{T}_{\text{pick}}$ pixel coordinates to produce suitable features for object relocation. The cross-correlation between these two dense features yields action-value maps that predict the success rate of placement corresponding to equally-spaced discrete rotations of the end-effector. Ultimately, the exact pixel coordinates for the actions $a_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}})$ are determined by:

$$\begin{aligned} \mathcal{T}_{\text{pick}} &= \arg\max_{(u,v)} Q_{\text{pick}}((u, v) \mid o_t), \\ \mathcal{T}_{\text{place}} &= \arg\max_{\mathcal{T}_i} Q_{\text{place}}(\mathcal{T}_i \mid o_t, \mathcal{T}_{\text{pick}}) = \Phi_{\text{query}}(o_t[\mathcal{T}_{\text{pick}}]) * \Phi_{\text{key}}(o_t). \end{aligned} \tag{3.1}$$

Here, the pick position solely determines the pixel coordinates (u, v), whereas the place position selects the maximum action-value across each discrete rotation index i. All components – attention, query, and key– utilize Fully Convolutional Networks with variations in input size and channel dimensions (see A.1 for details). These networks process orthographic projections of the tabletop scene, merged from multiple top-down RGB-D cameras, as the observation o_t.

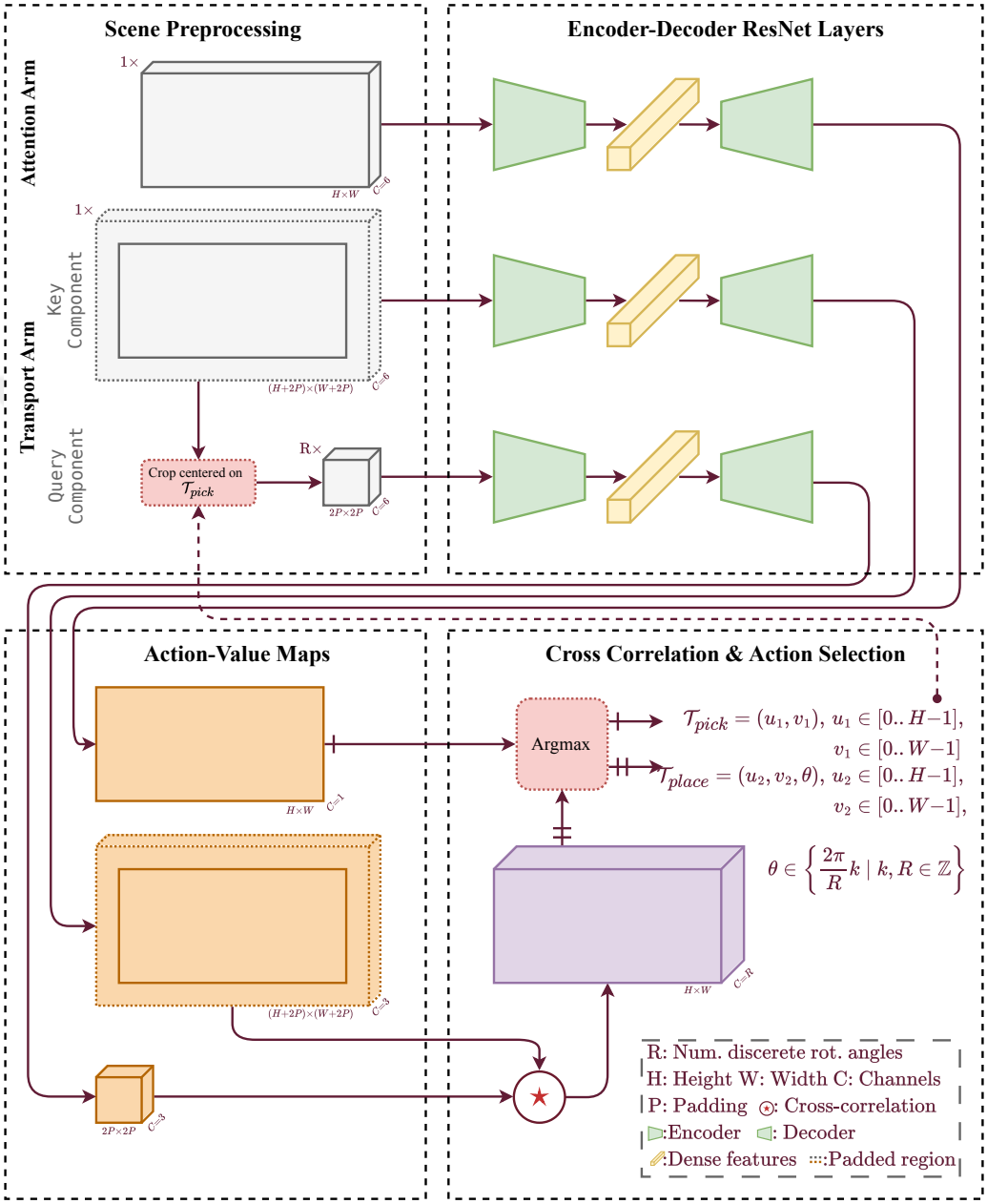

Figure 3.1: The architecture of Transporter Network. Each input observation frame undergoes preprocessing to be consumed by three identically structured ResNet models. These ResNet models incorporate an encoder-decoder structure to generate action-value maps correlating with picking and placing successes. Query and key maps are cross-correlated to determine the discrete angle that best fits the placing action, while the attention arm directly produces the pick map correlating with picking success.

3.2.2 Attention Mechanism in Transformers

The Transformer architecture plays a crucial role in state-of-the-art models, leveraging its multi-head attention mechanism to capture long-range relationships between sequences. This is achieved by dividing the input sequence into non-overlapping parts called the tokens and establishing relations between them through multi-head attention modules. The scaled dot-product attention is the key mechanism utilized in Transformers, where it computes a weighted sum across the elements of the values matrix V. The weights are determined by comparing the pairwise similarities between the query matrix Q and the key matrix K. The multi-head attention function can be expressed as:

$$\mathrm{MHA}(Q, K, V) = \mathrm{concat}(h_1, \ldots, h_h)W^O \tag{3.2}$$

$$\text{where } \mathrm{h}_i = \mathcal{A}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.3}$$

$$= \mathrm{softmax}\left[\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d}_k}\right]VW_i^V \tag{3.4}$$

where $Q, K, V \in R^{n \times d_m}$ are embedding matrices, n is the sequence length, d_m is the embedding dimension, h_i is the i^{th} head, and h is the total number of heads.

3.2.3 Discrete Fourier Transform

The Fourier Transform is a powerful tool used in signal processing and spectral analysis to examine the frequency components present within an arbitrary sequence. The discrete version of this algorithm applied to a sequence $x[n]$ with length N is defined by:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}, \ k = 0, 1, \ldots, N-1 \tag{3.5}$$

The resulting Discrete Fourier Transform (DFT) representation $X(k)$ is a weighted superposition of orthogonal basis vectors, which are complex sinusoidals. This new

representation can be used as the token mixing mechanism in place of multi-head attention. There are two primary approaches to compute the Fourier Transform: matrix multiplication or using the Fast Fourier Transform (FFT) algorithm [71]. In the matrix multiplication approach, an appropriately sized DFT matrix is cached and applied when needed. The matrix multiplication operation has a time complexity of $\mathcal{O}(N^2)$, but it can be faster on special hardware optimized for matrix operations. For general-purpose GPUs, FFT algorithm is an efficient implementation of the DFT that reduces computation time to $O(N \log N)$ by decomposing an N-point FFT into two $N/2$-point FFTs.

The Convolution Theorem [72] of Fourier Transform states that if a function $f(x)$ is element-wise multiplied with another function $g(x)$, then their Fourier Transforms are convolved in frequency domain. Conversely, convolving two arbitrary functions in the time domain results in the element-wise multiplication of their Fourier Transforms in the frequency domain. In other words, the convolution of two functions in the time domain corresponds to the product of their Fourier Transforms in the frequency domain. This understanding allows us to conceptualize cascaded Fourier Transform and MLP blocks as applying convolutions to an alternative representation of the original data.

Mathematically, if we have two sequences $x[n]$ and $h[n]$ with Fourier Transforms $X[k]$ and $H[k]$, respectively, then:

$$\mathcal{F}\{x[n] * h[n]\} = X[k] \cdot H[k] \tag{3.6}$$

where $\mathcal{F}$ denotes the Fourier Transform, $*$ represents convolution operation, and $X[k]$ and $H[k]$ are the Fourier Transforms of $x[n]$ and $h[n]$ respectively. The Fourier Transform of the convolution of the two sequences is the product of $X[k]$ and $H[k]$. This theorem allows us to manipulate signals in the frequency domain without having to perform complex calculations in the time domain.

The DFT seamlessly extends to the two-dimensional case. As a linear transform,

the 2D DFT can be obtained by applying two 1D DFTs along two dimensions on spatial data:

$$X[k,l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m,n] e^{-j2\pi(km/M+ln/N)} \\ \text{where } k = 0, 1, \ldots, M-1,\ l = 0, 1, \ldots, N-1. \tag{3.7}$$

The same properties observed in the one dimensional DFT hold true for the two dimensional case, including the convolution theorem. This theorem enables the SpTpTf architecture to efficiently manipulate multi-channel image signals without the need for complex calculations in the spatial domain.

3.3 Transporting Transformer Architecture

The Transporting Transformer (TpTf) model utilizes a self-attention based architecture to generate pick and place actions by processing RGB-D environment data (Figure 3.2). The proposed model consists of a convolutional stem that divides the input into Transformer-style tokens, which are then processed by the TpTf encoder backbone. This encoder is composed of alternating layers of self-attention and multilayer perceptrons, with residual connections between them, following the conventional architecture of a Transformer encoder. Specifically, the attention layer employs vanilla self-attention, while the MLP layer incorporates depthwise convolutions to facilitate close proximity interactions across channels without excessive computational overhead.

A single transformer encoder processes a single scene copy to produce dense features, which are subsequently processed by three smaller upsampling encoder layers with patch expansion. Two of these layers restructure the features into a grid, pad and reflatten them back into tokens to account for k discrete rotational bins. In contrast to the main TpTf encoder, the upsampling encoder layers iteratively expand

patches while reducing embedding dimensionality, effectively refining extracted features into action-value maps. This process culminates in bilinear upsampling followed by a linear layer, which strikes a balance between accuracy and parameter count. To select the optimal action, one of the padded action-value maps is rotated according to discrete rotation bins and partially cropped around the pick coordinates, after the pick coordinates are computed as the argument that maximizes the pick action-value map. This cropped map is then correlated with the full padded action-value map through cross-correlation, effectively merging maps where the object is transported at different angular positions. The coordinate maximizing successful placing probability across all discrete rotation bins is selected as the place action.

The TpTf adheres to the rules defined by the Transporter Network, however, we make improvements to the underlying architecture. By utilizing self-attention, we aim to boost performance across different tasks. Although using self-attention has a computational cost trade-off compared to the original convolutional model, namely, as tokens grow to resemble the final output map shape, attention layers compute interactions between each token, resulting in increasing computational costs that scale quadratically with the number of tokens. To alleviate this, we employ patch expansion encoder layers, as described previously, to iteratively reduce the embedding dimension and conclude with a bilinear upsampling layer. Moreover, we take partial crops for discrete rotational bins after dense features are computed by the encoder backbone, allowing for a single shared backbone for all downstream computations, as opposed to three convolutional arms used in the original Transporter architecture, thus further reducing the amount of required computations.

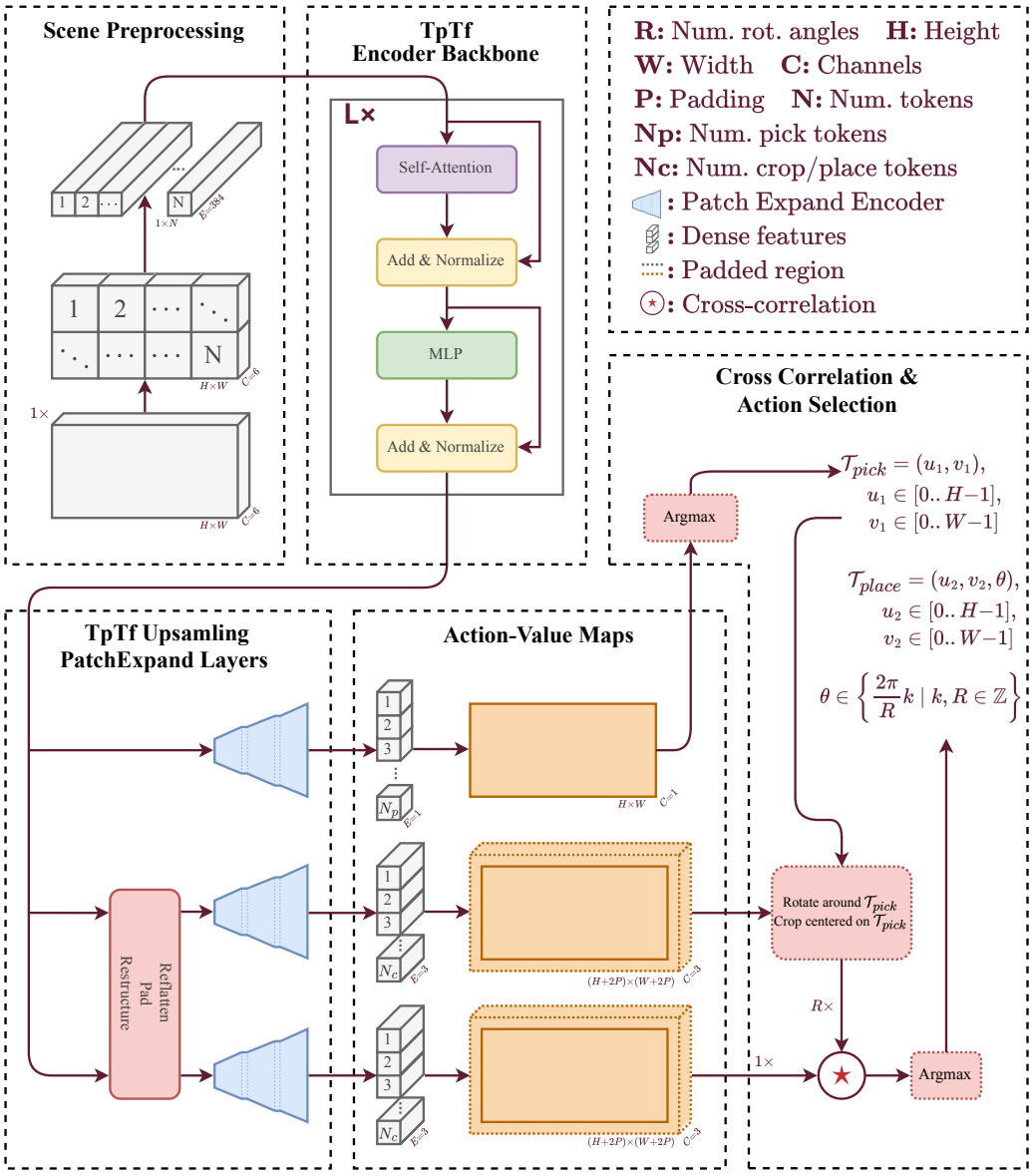

Figure 3.2: The architecture of the Transporting Transformer (TpTf) Network. The input observation is split into non-overlapping patches and flattened before processed by the encoder. The shared encoded features are then refined by patch expand encoder layers, which reduce embedding depth and increase token count, ultimately generating tokens that represent pick and place action-value maps. Rotational bins are derived by taking partial crops around pick coordinates obtained from the corresponding map, followed by cross-correlation to obtain a map that correlates with placing coordinates and discrete rotation bin.

3.4 Spectral Transporting Transformer Architecture

The Spectral Transporting Transformer (SpTpTf) extends the TpTf architecture by altering the token-mixing mechanism employed in its encoder blocks (Figure 3.3). Like TpTf, SpTpTf processes RGB-D observations to produce pick-and-place action-value maps. However, instead of the multi-head attention layers used in TpTf, SpTpTf introduces an $O(N \log N)$ spectral mixer that applies frequency-domain filters to enable long-range token interactions.

The spectral mixer operates by performing discrete Fourier transforms (DFTs) along both the feature and the spatial-grid (sequence) dimensions, thereby converting real-valued inputs into complex-valued representations. These complex features are multiplied element-wise (Hadamard product) with learnable complex-valued filter weights, and the result is transformed back to the spatial domain via inverse DFTs. For an input tensor x, the computation performed by the spectral mixer can be expressed as

$$X = \mathcal{F}_{\text{seq}}(\mathcal{F}_{\text{feature}}(x)) \tag{3.8}$$

$$z = X \odot \mathcal{H} \tag{3.9}$$

$$y = \mathcal{F}^{-1}_{\text{feature}}(\mathcal{F}^{-1}_{\text{seq}}(z)) \tag{3.10}$$

where $\mathcal{F}_{\text{seq}}$ denotes the DFT along the sequence (grid) dimension, $\mathcal{F}_{\text{feature}}$ denotes the DFT along the feature dimension, X is the complex-valued tensor obtained after the two forward transforms, $\mathcal{H}$ is a learnable complex weight tensor representing the spectral filters, z is the filtered complex tensor, and y is the real-valued output obtained after the inverse transforms.

Thus, the final output y constitutes a filtered version of the original input x, where the filtering is performed by learnable spectral kernels in the Fourier domain.

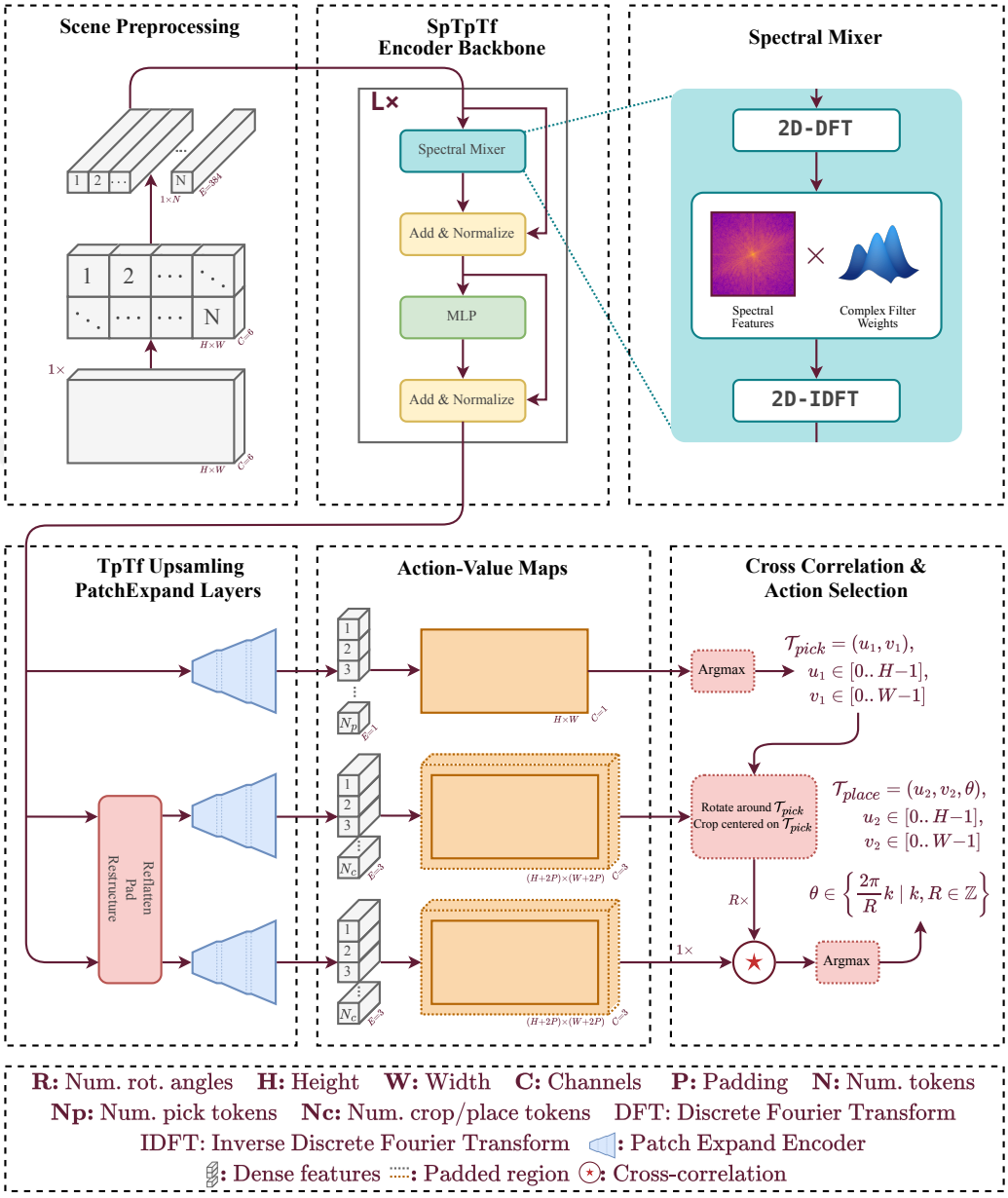

Figure 3.3: The architecture of the Spectral Transporting Transformer (SpTpTf) Network. The input observation is split into non-overlapping patches and flattened before processed by the encoder. The encoder block consists of 2D-DFT based spectral mixer and MLP layers with residual connections in between. The shared encoded features are then refined by patch expand encoder layers, which reduce embedding depth and increase token count, ultimately generating tokens that represent pick and place action-value maps. Rotational bins are derived by taking partial crops around pick coordinates obtained from the corresponding map, followed by cross-correlation to obtain a map that correlates with placing coordinates and discrete rotation bin.

3.5 Multi-task Learning Scheme for Ravens

Although stateless, TpTf can infer its current state for multi-step objectives through visual feedback from the environment. To showcase the enhanced capabilities of our model, we introduce a novel learning procedure (Algorithm 1) that enables the simultaneous learning of multiple tasks from a single model, thus eliminating the need to train individual models for each task as was done in the original Transporter work.

We formulate multi-task learning with a single model by assuming access to a collection of datasets $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$, where each dataset $\phi_i = \{\eta_{i1}, \eta_{i2}, \ldots, \eta_{in}\}$ contains n expert demonstrations for a single task. Each demonstration is a sequence of observation-action pairs $\eta_{ij} = \{(o_0, a_0), (o_1, a_1), \ldots\}$, defining a trajectory. To formalize the sampling procedure, we define a sampling function $S(\Phi, \kappa)$, which takes as input the collection of datasets Φ and the number of samples to draw κ. The function operates as follows: (1) select a dataset ϕ_i from Φ uniformly at random; (2) draw κ samples without replacement from the chosen dataset, denoted as $\eta_{ij_1}, \eta_{ij_2}, \ldots, \eta_{ij_\kappa}$, where $j_1, j_2, \ldots, j_\kappa$ are distinct indices in $[1..n]$; and (3) use these κ samples for training purposes. The sampling process is repeated iteratively until all sub-datasets ϕ_k in Φ have been exhausted. Specifically, after training on κ samples from the first chosen dataset, the function $S(\Phi, \kappa)$ selects another dataset ϕ_k (where $k \neq i$) from Φ uniformly at random and repeats the sampling procedure. After the model has been trained on κ samples from each task, the process continues by repeating the sampling procedure over all datasets Φ until the maximum number of training iterations is reached, ensuring that every sample across all tasks has been utilized for training at least once. Consequently, the sampling strategy can be represented as a sequence of sampling operations:

$$S(\Phi, \kappa) = \{(\eta_{i_1 j_1}, \ldots, \eta_{i_1 j_\kappa}), (\eta_{i_2 j_1}, \ldots, \eta_{i_2 j_\kappa}), \ldots\} \tag{3.11}$$

Algorithm 1 Sampling procedure $S(\Phi, \kappa)$ for multi-task training

Require: Set of task datasets $\Phi = \{\phi_1, \ldots, \phi_{|\Phi|}\}$, number of samples per draw κ, total training iterations T

$t \leftarrow 0$ ▷ global iteration counter
while $t < T$ **do**
  Shuffle Φ to obtain an ordering $(\phi_{i_1}, \ldots, \phi_{i_{|\Phi|}})$
  for $j = 1$ **to** $|\Phi|$ **do**
    $i \leftarrow i_j$ ▷ current dataset index
    Draw κ samples from ϕ_i and train network
    $t \leftarrow t + \kappa$
    if $t \geq T$ **then**
      break
    end if
  end for
end while

where $i_1, i_2, \ldots$ represent distinct dataset indices in $[1..m]$ and $j_1, j_2, \ldots$ represent unique sample indices within those datasets, each sampled uniformly. This approach ensures that training is performed in a round-robin fashion across all available datasets and their respective samples, maximizing the utilization of diverse expert demonstrations for multi-task learning purposes. Regardless of the current task, training loss is averaged pixel-wise cross-entropy loss between generated and one-hot encoded expert demonstration action-value maps as the model requires no task-specific input during training.

Chapter 4

Experiments

4.1 Setup

We utilize the exact same expert demonstrations used in the original Transporter paper, generated by the Ravens benchmark. We train using all datasets present in the benchmark:

(a) `align-box-corner`: lift the box of arbitrary dimensions and align one of its corners with the L-shaped marker on the tabletop.

(b) `assembling-kits`: collect heterogeneous objects and position each onto a board bearing its matching silhouette.

(c) `block-insertion`: grasp the L-shaped red block and insert it into the corresponding L-shaped receptacle.

(d) `manipulating-rope`: reconfigure a deformable rope so that it spans the three sides of a square, connecting the designated endpoints.

(e) `packing-boxes`: grasp boxes of varying sizes and pack them tightly within a container.

(f) `palletizing-boxes`: pick up uniform, fixed-size boxes and assemble them in alternating (transposed) layers on a pallet.

(g) `place-red-in-green`: retrieve the red blocks and deposit them into the green bowls, interspersed among other items.

(h) `stack-block-pyramid`: arrange six blocks in a 3-2-1 pyramidal configuration, adhering to a prescribed rainbow-color order.

(i) `sweeping-piles`: propel clusters of small objects toward a designated goal zone marked on the tabletop.

(j) `towers-of-hanoi`: transfer the disks sequentially from one peg to another, ensuring that no larger disk rests atop a smaller one.

Each task presents various challenges, some requiring precise placement, handling unseen objects, multi-step sequencing, deformable objects, and more. In each run, we train for 40,000 iterations where each episode's sample size is 10^{ε}, with $\varepsilon \in [0..3]$, to test for sample efficiency. We use Transporter Networks as the baseline to compare with and retrain it in our setup for fairness.

4.2 Single Task Learning

We compare TpTf and its variant SpTpTf with the Transporter Networks by training on all tasks from scratch using the standard Ravens benchmark. The comprehensive results are presented in Table 4.1. We report two different metrics across all tasks: mean and standard deviation of success rate, averaged over all episodes (Table A.2)

and over the last 10% of training episodes (denoted with the "L4" suffix). This is important because the model is updated after each expert demonstration, resulting in oscillatory behavior at the beginning, which can make actual performance appear worse. As training stabilizes towards the end of episodes, we believe it is more informative to report this stabilized performance rather than just the maximum achieved or average performance over all episodes.

Across the board, the TpTf models outperforms Transporter Networks in terms of success rate by a margin of a few percentage points. When matched in terms of success rate, TpTf models generally display a lower standard deviation across test demonstrations, with one notable exception occurring when the number of demos is relatively low, particularly when 1 demo is used. Although attention mechanism-based transformer architectures typically require large-scale datasets to outperform conventional architectures, such as convolutional ones, in this case, the performance gap narrows quickly even with as few as 10 unique demos, indicating that sample efficiency is mostly maintained despite using an attention-based architecture. The spectral variant SpTpTf, together with its alternate token-mixing layers, exhibits a comparable pattern, suggesting that filtering features in the complex domain promotes efficient long-range interactions among tokens that are pertinent to transport-related tasks, while preserving sample efficiency.

However, it's worth acknowledging that the original Transporter architecture already achieves a close to maximum success rate in most tasks, leaving limited room for improvement. Nevertheless, when considering multi-task training scenarios, as discussed in the next section, the benefits of TpTf become even more pronounced. This is where the true advantages of our approach become apparent, particularly in handling diverse tasks simultaneously.

Table 4.1: Single-task training success rate results are reported as mean (standard deviation) percentage points. The "-L4" notation denotes results over the last four episodes, which correspond to checkpoints taken during the final 10% of training iterations where models converge closer to their final performance. All models undergo training in an identical environment. In this training scheme, one model is trained per task for each number of demonstrations, with each training session lasting 40,000 iterations, totaling 1,600,000 iterations for each architecture. Symbol † denotes our models. Form2Fit results are taken from [42].

Task	**align-box-corner**				**assembling-kits**			
Num-Demos	1	10	100	1000	1	10	100	1000
TpTf-L4 †	6.2 (24.1)	84.8 (35.9)	96.8 (16.6)	97.5 (15.2)	26.2 (21.3)	66.5 (26.4)	92.8 (16.1)	97.2 (10.2)
SpTpTf-L4 †	2.8 (15.9)	77.5 (40.9)	93.5 (23.0)	99.2 (6.0)	30.9 (20.8)	73.8 (24.3)	93.4 (15.3)	94.7 (14.3)
Transporter-L4	13.2 (33.8)	62.5 (47.9)	95.2 (20.8)	99.2 (6.0)	16.9 (17.9)	62.2 (23.0)	92.0 (17.4)	95.9 (11.8)
Form2Fit [42]	7.0	2.0	5.0	16.0	3.4	7.6	24.2	37.6
	block-insertion				**manipulating-rope**			
	1	10	100	1000	1	10	100	1000
TpTf-L4 †	41.0 (49.1)	99.8 (2.5)	99.5 (3.5)	100.0 (0.0)	0.6 (2.7)	78.2 (33.4)	85.5 (29.2)	94.3 (17.9)
SpTpTf-L4 †	23.0 (42.0)	99.5 (5.0)	99.2 (7.5)	100.0 (0.0)	1.2 (5.6)	79.0 (33.0)	93.4 (18.6)	94.2 (18.1)
Transporter-L4	92.8 (25.6)	97.0 (16.9)	100.0 (0.0)	100.0 (0.0)	15.2 (26.3)	69.3 (39.8)	90.2 (20.7)	83.4 (25.3)
Form2Fit [42]	17.0	19.0	23.0	29.0	11.9	38.8	36.7	47.7
	packing-boxes				**palletizing-boxes**			
	1	10	100	1000	1	10	100	1000
TpTf-L4 †	82.0 (21.8)	99.5 (4.0)	97.8 (11.4)	99.7 (2.5)	93.8 (11.1)	97.3 (7.8)	99.6 (2.3)	99.9 (1.1)
SpTpTf-L4 †	71.4 (26.1)	98.2 (8.1)	99.5 (3.6)	99.5 (4.6)	92.0 (11.8)	97.5 (6.3)	99.5 (2.3)	98.6 (5.2)
Transporter-L4	87.4 (23.1)	96.8 (12.1)	99.5 (4.5)	98.9 (7.9)	79.6 (12.2)	97.8 (5.5)	97.0 (8.8)	96.6 (7.1)
Form2Fit [42]	29.9	52.5	62.3	66.8	21.6	42.0	52.1	65.3
	place-red-in-green				**stack-block-pyramid**			
	1	10	100	1000	1	10	100	1000
TpTf-L4 †	63.8 (38.3)	98.5 (9.8)	99.0 (6.8)	100.0 (0.0)	7.2 (10.1)	77.9 (30.3)	94.6 (16.4)	98.9 (7.6)
SpTpTf-L4 †	43.9 (42.0)	98.2 (12.0)	100.0 (0.0)	100.0 (0.0)	2.9 (7.3)	46.0 (26.6)	90.7 (23.0)	96.7 (14.4)
Transporter-L4	73.9 (38.2)	100.0 (0.0)	99.8 (2.5)	100.0 (0.0)	11.2 (12.7)	65.9 (29.8)	95.3 (15.6)	97.5 (10.5)
Form2Fit [42]	83.4	100.0	100.0	100.0	19.7	17.5	18.5	32.5
	sweeping-piles				**towers-of-hanoi**			
	1	10	100	1000	1	10	100	1000
TpTf-L4 †	98.1 (4.2)	99.9 (0.9)	100.0 (0.2)	100.0 (0.1)	90.4 (20.2)	94.5 (16.6)	99.2 (6.2)	100.0 (0.0)
SpTpTf-L4 †	97.6 (5.5)	100.0 (0.3)	100.0 (0.1)	100.0 (0.2)	87.5 (21.5)	92.6 (20.3)	100.0 (0.0)	96.9 (9.0)
Transporter-L4	97.9 (7.0)	99.3 (3.8)	99.3 (3.9)	99.4 (3.3)	67.3 (38.0)	93.3 (15.8)	99.9 (1.1)	99.8 (2.1)
Form2Fit [42]	13.2	15.6	26.7	38.4	3.6	4.4	3.7	7.0

4.3 Multi Task Training

We implement the multi-task sampling procedure $S(\Phi, \kappa)$ on the Ravens benchmark and report the mean and standard deviation of success rate metrics across all tasks, averaged over the last 10% of training episodes (denoted as "L4") in Table 4.2 (for results averaged over all episodes see Table A.3). Additionally, we analyze the effect of the number of unique samples to train on before switching to a different task κ. A single model is trained using all tasks and evaluated separately for each task.

Our TpTf model outperforms Transporter Networks by a greater margin in terms of success rate with lower deviations in the multi-task training scheme. While Transporter Networks may surpass TpTf in a limited number of demonstrations (1 or 10), this trend is not observed when the number of demos exceeds 100, consistent with previous results and indicating that sample efficiency is largely preserved.

The enhanced capacity of our TpTf model becomes apparent in tasks like `assembling-kits`, `manipulating-rope`, `stack-block-pyramid`, and `towers-of-hanoi`. The latter three tasks require multi-step sequencing and proper adaptation to the task without forgetting previously learned tasks. While the Transporter Networks performed well for these tasks in the single-task case, they are bottlenecked by the model's capacity in the multi-task case. SpTpTf's performance is competitive with that of Transporter Networks, delivering comparable results across a variety of tasks and different numbers of demonstrations. Its primary advantage becomes evident in the complexity analysis section. Nevertheless, both models achieve nearly 100% success rates in some tasks, indicating that policy learning based on the original task-independent problem formulation is applicable in broader settings.

Additionally, we conduct experiments with varying sample sizes during the training interval for a selected dataset κ to assess its impact on success rates. The results show that rapid switching between different tasks with smaller κ values yields superior performance for both models. Conversely, higher κ values cause overfitting to the active task, leading models to eventually "forget" previously learned tasks.

Table 4.2: Multi-task training mean (SD) success rates. The "-L4" suffix denotes the mean over the last four episodes (the final 10% of training). Symbol † denotes our models.

Task		**align-box-corner**				**assembling-kits**			
κ	Num-Demos	1	10	100	1000	1	10	100	1000
1	TpTf-L4 †	17.0 (37.5)	75.8 (42.1)	96.8 (14.5)	97.2 (13.7)	8.6 (12.4)	60.2 (27.4)	74.4 (27.0)	71.6 (28.2)
	SpTpTf-L4 †	29.2 (45.3)	68.5 (45.6)	96.8 (15.6)	96.8 (17.3)	7.9 (11.9)	39.4 (26.4)	64.7 (28.1)	67.5 (26.4)
	Transporter-L4	46.5 (48.7)	80.5 (38.8)	98.5 (7.9)	98.8 (7.8)	14.3 (16.4)	54.2 (26.4)	61.2 (26.9)	63.1 (25.1)
10	TpTf-L4 †	49.0 (49.9)	61.0 (47.8)	98.2 (13.0)	98.8 (7.8)	10.0 (14.7)	55.3 (27.7)	76.9 (25.2)	74.1 (27.5)
	SpTpTf-L4 †	27.0 (44.1)	76.5 (41.9)	93.0 (25.0)	96.2 (17.7)	6.9 (11.8)	41.1 (27.4)	64.6 (27.4)	71.3 (25.1)
	Transporter-L4	45.2 (49.2)	70.8 (42.7)	96.5 (17.3)	94.0 (20.9)	12.2 (14.3)	41.7 (26.1)	46.5 (24.9)	50.1 (25.5)
100	TpTf-L4 †	17.2 (37.7)	77.5 (41.3)	97.5 (14.9)	98.5 (7.9)	8.5 (12.4)	48.0 (28.5)	70.9 (28.2)	72.3 (27.6)
	SpTpTf-L4 †	35.2 (47.3)	80.5 (37.0)	88.5 (29.0)	89.2 (28.4)	8.8 (12.5)	43.8 (25.5)	59.5 (29.4)	60.2 (28.5)
	Transporter-L4	41.2 (49.0)	72.2 (40.4)	87.0 (30.7)	67.8 (29.6)	4.9 (9.5)	32.4 (23.5)	30.6 (22.1)	40.3 (25.4)
		block-insertion				**manipulating-rope**			
		1	10	100	1000	1	10	100	1000
1	TpTf-L4 †	99.5 (3.5)	100.0 (0.0)	99.8 (2.5)	100.0 (0.0)	5.3 (16.4)	71.4 (36.6)	69.0 (35.9)	80.8 (28.5)
	SpTpTf-L4 †	94.8 (22.2)	97.8 (13.9)	99.5 (3.5)	98.8 (5.4)	0.6 (2.7)	37.6 (39.8)	63.6 (41.7)	65.5 (37.8)
	Transporter-L4	94.8 (21.0)	99.8 (2.5)	99.2 (6.0)	99.5 (5.0)	45.5 (40.6)	72.6 (37.5)	85.6 (28.2)	69.7 (34.9)
10	TpTf-L4 †	99.0 (7.0)	98.2 (10.4)	99.5 (5.0)	99.8 (2.5)	3.8 (14.2)	55.8 (40.2)	73.1 (30.5)	72.6 (35.5)
	SpTpTf-L4 †	93.0 (25.3)	99.5 (3.5)	99.8 (2.5)	99.5 (5.0)	1.2 (4.3)	54.1 (41.1)	57.3 (41.7)	75.8 (35.1)
	Transporter-L4	96.0 (15.6)	98.2 (12.7)	100.0 (0.0)	99.5 (5.0)	28.4 (37.3)	62.9 (36.7)	79.2 (32.4)	86.9 (26.9)
100	TpTf-L4 †	95.2 (20.5)	98.0 (12.0)	100.0 (0.0)	99.8 (2.5)	2.3 (9.0)	65.6 (40.8)	82.3 (31.6)	69.0 (36.1)
	SpTpTf-L4 †	96.5 (17.4)	99.2 (7.5)	99.8 (2.5)	99.2 (7.5)	1.8 (6.2)	26.8 (26.4)	45.2 (29.5)	41.3 (19.5)
	Transporter-L4	95.5 (19.8)	89.0 (20.7)	95.2 (16.6)	89.5 (24.6)	26.6 (35.0)	53.4 (39.9)	59.4 (25.5)	68.3 (27.4)
		packing-boxes				**palletizing-boxes**			
		1	10	100	1000	1	10	100	1000
1	TpTf-L4 †	96.5 (12.4)	97.2 (11.6)	99.8 (2.1)	99.7 (2.7)	84.8 (13.7)	92.7 (10.0)	92.3 (9.3)	95.9 (7.8)
	SpTpTf-L4 †	93.2 (16.1)	95.4 (14.5)	99.8 (1.2)	99.8 (1.4)	70.0 (12.8)	77.0 (12.5)	80.5 (11.4)	69.5 (14.9)
	Transporter-L4	83.1 (22.9)	96.6 (11.8)	99.9 (0.6)	99.9 (0.7)	65.8 (14.6)	91.8 (13.2)	93.9 (10.3)	94.7 (8.0)
10	TpTf-L4 †	94.6 (15.4)	96.0 (13.6)	100.0 (0.1)	99.7 (2.5)	92.8 (9.4)	93.3 (8.3)	91.5 (9.1)	94.6 (8.1)
	SpTpTf-L4 †	95.9 (11.8)	98.2 (7.6)	99.5 (2.8)	99.7 (2.7)	77.7 (13.4)	74.1 (13.1)	80.8 (13.1)	68.2 (12.7)
	Transporter-L4	82.5 (24.6)	97.8 (10.2)	99.5 (4.3)	99.4 (4.4)	74.5 (13.5)	79.9 (12.9)	70.9 (15.9)	94.0 (6.8)
100	TpTf-L4 †	94.0 (14.9)	99.0 (6.7)	99.9 (0.6)	99.6 (2.6)	72.8 (12.3)	95.6 (5.9)	91.1 (9.4)	94.1 (7.0)
	SpTpTf-L4 †	90.7 (18.5)	98.6 (5.8)	98.2 (6.0)	98.6 (4.8)	74.2 (13.5)	73.6 (13.7)	77.8 (12.2)	77.2 (15.3)
	Transporter-L4	75.9 (23.5)	89.1 (17.8)	98.0 (6.5)	98.2 (7.6)	68.6 (12.7)	71.1 (12.8)	61.2 (11.7)	79.8 (12.2)
		place-red-in-green				**stack-block-pyramid**			
		1	10	100	1000	1	10	100	1000
1	TpTf-L4 †	65.8 (40.8)	96.5 (14.2)	100.0 (0.0)	100.0 (0.0)	9.5 (13.7)	44.6 (33.0)	72.9 (32.9)	78.4 (31.0)
	SpTpTf-L4 †	47.8 (43.8)	69.5 (40.5)	99.2 (5.9)	90.6 (24.1)	6.0 (10.0)	13.0 (11.7)	15.3 (14.8)	29.0 (22.3)
	Transporter-L4	57.9 (44.5)	96.4 (15.6)	98.4 (5.3)	98.7 (8.5)	11.7 (14.9)	21.6 (16.9)	27.7 (18.8)	23.4 (16.6)
10	TpTf-L4 †	80.8 (34.2)	92.5 (22.1)	100.0 (0.0)	100.0 (0.0)	20.2 (19.3)	41.5 (33.0)	76.1 (31.0)	77.3 (28.8)
	SpTpTf-L4 †	44.0 (43.7)	73.2 (38.4)	94.0 (18.1)	93.3 (21.0)	7.3 (10.6)	15.3 (13.9)	13.7 (14.9)	19.2 (17.0)
	Transporter-L4	51.4 (44.4)	95.6 (12.2)	98.0 (5.9)	93.7 (11.4)	8.8 (12.5)	17.0 (16.0)	23.0 (16.4)	31.5 (15.5)
100	TpTf-L4 †	58.5 (43.7)	96.2 (15.1)	100.0 (0.0)	99.9 (0.8)	2.7 (6.8)	29.0 (25.3)	52.0 (30.5)	38.2 (28.6)
	SpTpTf-L4 †	46.2 (44.5)	66.3 (36.3)	92.5 (19.4)	83.6 (26.8)	5.5 (9.2)	12.1 (12.8)	9.7 (10.9)	11.5 (12.1)
	Transporter-L4	58.6 (37.0)	79.1 (26.3)	95.1 (16.6)	88.0 (21.4)	0.4 (2.5)	22.2 (15.1)	8.4 (10.9)	15.1 (9.5)
		sweeping-piles				**towers-of-hanoi**			
		1	10	100	1000	1	10	100	1000
1	TpTf-L4 †	94.3 (10.5)	96.5 (9.4)	96.8 (8.9)	96.6 (9.9)	61.6 (34.7)	88.6 (22.9)	84.7 (24.8)	89.5 (20.8)
	SpTpTf-L4 †	91.5 (12.6)	99.8 (1.2)	99.8 (0.6)	97.1 (7.6)	28.7 (27.5)	71.9 (30.7)	64.3 (34.3)	45.0 (22.4)
	Transporter-L4	92.0 (16.1)	93.4 (14.4)	95.5 (11.7)	90.4 (18.7)	42.0 (30.8)	81.7 (23.2)	57.4 (31.5)	37.7 (21.0)
10	TpTf-L4 †	94.5 (9.4)	88.8 (13.5)	96.9 (8.7)	96.3 (10.2)	49.5 (35.9)	92.8 (16.1)	96.1 (13.7)	91.8 (16.8)
	SpTpTf-L4 †	85.5 (16.8)	99.7 (1.1)	99.6 (1.2)	98.6 (6.2)	21.7 (18.6)	56.1 (34.7)	48.3 (18.3)	75.7 (27.2)
	Transporter-L4	92.9 (15.9)	95.6 (10.8)	95.5 (10.6)	92.9 (19.4)	44.7 (31.6)	50.6 (26.5)	47.8 (22.2)	52.1 (28.8)
100	TpTf-L4 †	72.6 (27.7)	95.2 (13.4)	80.5 (29.4)	66.9 (33.2)	38.7 (31.6)	92.2 (19.1)	93.1 (16.1)	90.3 (15.6)
	SpTpTf-L4 †	81.3 (21.4)	63.3 (19.1)	74.5 (22.2)	45.1 (23.3)	32.4 (28.6)	45.6 (23.3)	51.0 (26.7)	43.4 (22.3)
	Transporter-L4	64.5 (21.2)	54.5 (30.7)	42.4 (18.1)	53.7 (20.1)	17.2 (15.6)	18.0 (11.8)	17.8 (6.6)	15.2 (4.8)

4.4 Complexity Analysis and Model Size Ablation

This section presents a comparative analysis of model complexity and a thorough ablation study of our proposed TpTf models and another smaller variant, TpTf-xs.

4.4.1 Complexity Analysis

Our complexity analysis reveals that all three TpTf variants require significantly fewer computational resources than the Transporter model. As shown in Table 4.3, the full-size TpTf consumes 46.88 and 140.6 GFLOPs during forward and total passes, respectively, with 23.05M parameters and 1542MB memory usage. The compact TpTf-xs reduces this further to 21.14 GFLOPs (forward) and 63.41 GFLOPs (total), 10.41M parameters, and 1060MB memory. The spectral variant strikes a middle ground: it requires only 14.39 GFLOPs (forward) and 43.17 GFLOPs (total), 15.15M parameters, and 1176MB of memory, while still achieving a 57.68MB model size. In contrast, the Transporter demands 138.8 and 416.0 GFLOPs, 29.50M parameters, and 1874MB memory. These results underscore the effectiveness of our design choices in minimizing both arithmetic operations and memory footprint across all TpTf configurations.

4.4.2 Ablation Study

To investigate the impact of model size on performance, we performed an ablation study that varied the embedding dimension of the self-attention layers in TpTf and its compact variant TpTf-xs, and we additionally evaluated the spectral version SpTpTf (Table 4.4). Across the three κ settings, the full-size TpTf achieves the highest multi-task success rates, while the reduced-capacity TpTf-xs incurs only a

Table 4.3: Efficiency comparison of the TpTf variants with the Transporter during Ravens training. To evaluate efficiency, we consider metrics such as Floating Point Operations (FLOPS), Multiply-Accumulate Operations (MACs) during inference and training, parameter counts, GPU RAM usage during training, and model (32-bit float) storage size. All models are trained with the same settings on identical software and hardware setups to ensure a valid comparison. Unified architecture of TpTf results lower operations and parameter counts across the board. Symbol † denotes our models.

Model	Forward FLOPs (G)	Total FLOPs (G)	Forward MACs (G)	Total MACs (G)	Params (M)	Memory (MB)	Size (MB)
TpTf †	46.88	140.6	23.33	70.00	23.05	1542	87.93
TpTf-xs †	21.14	63.41	10.50	31.50	10.41	1060	39.72
SpTpTf †	14.39	43.17	7.12	21.37	15.12	1176	57.68
Transporter	138.8	416.4	69.23	207.7	29.50	1874	112.5

modest drop. SpTpTf performs slightly below both TpTf variants but nevertheless matches or even exceeds the original Transporter baseline in several cases.

In the final-training ("-L4") checkpoints, the gap widens: the TpTf variants receive $+12.4, +14.2, +10.7$ performance boosts versus Transporter's $+9.8$ percentage points in the $\kappa = 1$ setting. All three TpTf variants retain strong single-task performance, surpassing Transporter's results. Thus, even the smallest SpTpTf preserves the core mechanisms of the full model and remains competitively performant despite requiring substantially fewer floating point operations. Moreover, the self-attention-based variants outperform the standard Transporter, demonstrating that effective policy learning can be achieved with far fewer parameters.

Table 4.4: Model size comparison of TpTf variants with Transporter task success rates. Success rates from multi-task and single-task training are expressed as mean ± standard deviation percentage points, averaged over all episodes and the last four episodes (denoted by "-L4"), which correspond to checkpoints taken during the final 10% of training iterations where models converge closer to their final performance; all models are trained in an identical environment. Results are provided for the case of 1000 demo instances and averaged across *all* tasks. Per-task mean success rates are reported in Table 4.2 (results for the final 10% of checkpoints) and Table A.3 (results for all checkpoints), respectively. Symbol † denotes our models.

	κ			
Model	**1**	**10**	**100**	**Single Task**
TpTf †	78.6±15.7	74.3±16.2	61.7±15.4	93.0±8.8
TpTf-xs †	75.4±15.6	70.5±15.5	61.6±16.4	93.7±8.8
SpTpTf †	65.3±17.6	62.8±17.0	49.3±16.7	95.1±7.7
Transporter	67.8±15.5	61.5±16.1	38.9±16.3	91.9±10.7
TpTf/L4 †	91.0±14.3	90.5±14.0	82.9±16.2	98.8±5.5
TpTf-xs/L4 †	89.6±13.8	89.1±15.3	84.7±17.2	98.3±6.9
SpTpTf/L4 †	76.0±18.0	79.8±17.0	64.9±18.9	98.0±7.2
Transporter/L4	77.6±14.6	79.4±16.5	61.6±18.3	97.1±7.4
Form2Fit[42]	-	-	-	44.0

Chapter 5

Discussion & Limitations

Although we have extended the baseline framework, several limitations remain.

First, our architectural changes are confined to merging the redundant feature-extraction streams into a single shared backbone. All higher-level components of the Transporter framework are left unchanged: the placement problem is still conditioned on the pick location, rotation is discretized into bins, and cross-correlation is used to produce the placement-action value map. It is therefore important to revisit these design choices. In particular, we should investigate whether a continuous rotation representation can replace the discrete bins, and whether the cross-correlation step is indispensable or if the placement map can be generated directly from the conditioned pick position.

Second, our evaluation is limited to the original ten tasks of the Ravens benchmark. Recent extensions such as the deformable-object dataset [46] have not been considered, nor have works that integrate Transporter as a module for natural-language grounding, long-horizon planning, or that modify the convolutional layers in-place. These orthogonal extensions could be combined with our backbone-sharing

and self-attention modification to assess their joint impact.

Third, the Spectral Transporting Transformer (SpTpTf) component provides only an initial glimpse of the utility of spectral features for visual robotic manipulation. In more mature vision and language domains, spectral methods have matched and even surpassed attention-based baselines while requiring less computation. In our experiments, SpTpTf merely matches the original Transporter's performance, whereas the attention-based TpTf significantly exceeds it. This suggests that the full potential of spectral representations in manipulation remains untapped, although we cannot yet make definitive claims.

Future work will therefore pursue (i) more sophisticated spectral-feature processing pipelines, (ii) dynamic capacity allocation mechanisms such as conditional adapters or mixture-of-experts, and (iii) broader evaluations that include deformable-object tasks and multimodal extensions. These directions aim to narrow the performance gap between multi-task and single-task settings and to demonstrate the generality of the proposed architecture.

Our empirical results provide evidence for the following three conclusions.

5.1 Shared Backbone

Unified backbone models outperform models with multiple individual branches. Shared-backbone design in the Transporting Transformer consolidates redundant feature extraction of multi-branch architecture of the Transporter into a single encoder. By computing dense features once with the self-attention-augmented backbone and then re-using them for pick-generation, rotational-bin cropping, and placement scoring, the model eliminates redundant convolutions and dramatically cuts both FLOPs and memory usage. This unified representation also benefits from

the richer, globally contextualized embeddings that self-attention provides, improving task generalization and also alleviating the quadratic token-cost penalty because the subsequent sub-task specific patch-expansion and bilinear up-sampling stages keep the parameter count low (Table 4.3).

5.2 Multi Task Learning

Multi-task capabilities are limited by model capacity. The original Transporter architecture demonstrated that a single visual-perception backbone can be reused across many manipulation problems without any low-level redesign of the network. However, the baseline experiments in the Transporter Networks train a fresh copy of the model for every task, so the claim "one architecture for many tasks" is only proved partially—each model is still dedicated to a single objective. In practice, the amount of knowledge that a fixed-size network can retain across tasks is bounded by its capacity (number of parameters, depth of the feature hierarchy, and size of the latent space). When the same weights are asked to encode the policies for ten distinct pick-and-place variations, for example, the network must compress a larger and more diverse set of visual-action mappings into the same bottleneck. If the model is under-parameterized, interference between tasks will appear as degraded success rates on the harder or less-frequent tasks.

By sharing weights across tasks, we replace the 10 separate runs required by the original Transporter pipeline with a single training run, cutting total wall-clock training time by roughly a factor of 10. Moreover, the shared training process reduces redundant gradient computations on similar visual features across tasks. Additionally, we attain parameter economy by instead of deploying 10 copies of the network, we keep only one set of weights. This may especially be valuable for embedded robotic platforms where memory and power budgets are tight on the edge. The

same model can be loaded once and used for any of the learned tasks at inference time.

By introducing a principled multi-task sampling schedule, we demonstrate that a *single transporting model can learn multiple manipulation objectives simultaneously*, dramatically reducing training and deployment overhead (Table 4.4).

5.3 Self Attention and Spectral Token Mixing Mechanisms

Token mixers can effectively learn relevant features for transport tasks. The baseline Transporter Networks employ a purely convolutional architecture with ResNet backbones. Recent state-of-the-art vision models have largely transitioned from convolutional backbones to transformer-based networks. Our Transporting Transformer, which is built on self-attention, distinguishes itself from the fully-convolutional baseline in two fundamental ways.

First, it fuses information globally across all spatial tokens in a single pass, rather than relying on fixed-size convolutional kernels and separate branches that approximate long-range dependencies. By projecting the convolutional stem's patches into a shared embedding space and applying multi-head self-attention, each token can attend to every other token irrespective of spatial distance. This capability enables the model to capture holistic scene geometry—such as occluded object contours or distant targets—that would otherwise require deep stacks of dilated convolutions or handcrafted hierarchical branches.

Second, the attention weights are modulated adaptively by the current task context (e.g., pick vs. place, rotation bin). Consequently, the receptive field can expand

or contract on-the-fly, a flexibility unavailable in a static convolutional pipeline. The resulting attention layer therefore yields a more expressive, context-aware representation while preserving a modest parameter budget, thanks to the shared backbone described earlier. This provides a clear pathway for improving the quality of action-value maps beyond what the baseline's separate convolutional branches can achieve.

Spectral features for visual robotic manipulation.

Prior work on spectral representations has been confined largely to generic vision and natural-language processing domains. With SpTpTf, we directly investigate whether spectral features can benefit policy learning in robotic manipulation. By replacing the self-attention layers in TpTf with complex Fourier filters, we obtain performance that is competitive with the considerably larger Transporter Networks, while markedly reducing floating-point operations.

Thus, by integrating different token-mixing mechanisms—self-attention and spectral filtering—into a transporting model, we systematically assess how the choice of token mixer influences overall performance and computational requirements.

Chapter 6

Conclusion

In this work we introduced the Transporting Transformer (TpTf) — a transformer-based backbone that replaces the duplicated convolutional streams of the original Transporter Networks with a single, self-attention-augmented encoder. By generating pick- and place-action value maps directly from the shared representation, TpTf reduces redundant FLOPs, lowers memory consumption, and attains higher success rates across the ten Raven benchmark tasks. The unified backbone also enables a multi-task learning protocol in which a single model is trained once for all manipulation objectives, cutting wall-clock training time by an order of magnitude and simplifying possible deployment on resource-constrained robotic platforms.

Our experiments, however, reveal several open challenges that motivate future research. First, the current design still inherits legacy choices from the Transporter framework: placement conditioning is limited to a discrete set of rotation bins and the placement map is produced via cross-correlation. Exploring continuous rotation representations and direct placement-map decoders could further improve precision and eliminate the need for costly correlation operations. Second, while TpTf's

self-attention mixer proves effective, the spectral token-mixing variant SpTpTf only matches baseline performance; more sophisticated spectral pipelines, dynamic capacity mechanisms (e.g., conditional adapters or mixture-of-experts), and hybrid attention-spectral architectures may unlock the efficiency gains reported in vision and language domains. Third, our evaluation is confined to the original Ravens suite; extending the analysis to deformable-object benchmarks, language-grounded manipulation, and long-horizon planning will test the generality of the shared-backbone paradigm.

Beyond 2D orthographic projection image inputs, a natural next step is to generalize TpTf to 3D voxel or point-cloud representations, which would broaden its applicability to tasks such as cloth handling, granular media manipulation, and assembly in cluttered scenes. Finally, as Transformers scale, preserving sample efficiency becomes paramount for robotics, where data collection is expensive. Incorporating capacity-aware training schedules, or task-aware token routing could sustain performance gains without prohibitive data requirements.

In summary, TpTf demonstrates that a single, attention-driven backbone can serve as a versatile policy core for vision-based robotic manipulation, offering both computational savings and strong empirical results. By addressing the identified limitations—continuous action parametrization, richer token-mixing strategies, broader task suites, and 3D extensions—we anticipate that the approach has the potential to evolve into a robust, general-purpose manipulator capable of tackling the diverse challenges of real-world robotics.

Bibliography

[1] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*, pp. 726–747, PMLR, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers; distillation through attention," in *International Conference on Machine Learning*, vol. 139, pp. 10347–10357, July 2021.

[6] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

[7] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5728–5739, 2022.

[8] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, "Nyströmformer: A nyström-based algorithm for approximating self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 14138–14148, 2021.

[9] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[10] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[11] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," 2021.

[12] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24261–24272, 2021.

[13] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, and H. J'egou, "Resmlp: Feedforward

networks for image classification with data-efficient training," *arXiv preprint arXiv:2105.03404*, 2021.

[14] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay attention to mlps," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9204–9215, 2021.

[15] I. Bello, "Lambdanetworks: Modeling long-range interactions without attention," *arXiv preprint arXiv:2102.08602*, 2021.

[16] I. Eckstein, J. P. Lee-Thorp, J. Ainslie, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv preprint arXiv:2105.03824*, 2022.

[17] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[18] X. Zhao, M. Zhang, R. Tao, W. Li, W. Liao, L. Tian, and W. Philips, "Fractional fourier image transformer for multimodal remote sensing data classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[19] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[20] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[21] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[22] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, Spie, 1992.

[23] V. Narayanan and M. Likhachev, "Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances.," in *Robotics: Science and Systems*, 2016.

[24] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1386–1383, IEEE, 2017.

[25] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 international conference on 3D vision (3DV)*, pp. 728–737, IEEE, 2018.

[26] X. Chen, R. Chen, Z. Sui, Z. Ye, Y. Liu, R. I. Bahar, and O. C. Jenkins, "Grip: Generative robust inference and perception for semantic robot manipulation in adversarial environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3988–3995, IEEE, 2019.

[27] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation," in *Conference on robot learning*, pp. 1369–1378, PMLR, 2020.

[28] M. Gualtieri and R. Platt, "Robotic pick-and-place with uncertain object instance segmentation and shape completion," *IEEE robotics and automation letters*, vol. 6, no. 2, pp. 1753–1760, 2021.

[29] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*, pp. 1101–1112, PMLR, 2020.

[30] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, "Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11602–11610, 2020.

[31] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," in *The International Symposium of Robotics Research*, pp. 132–157, Springer, 2019.

[32] T. D. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih, "Unsupervised learning of object keypoints for perception and control," *Advances in neural information processing systems*, vol. 32, 2019.

[33] Y. Yoon, G. N. DeSouza, and A. C. Kak, "Real-time tracking and pose estimation for industrial objects using geometric features," in *2003 IEEE International conference on robotics and automation (cat. no. 03CH37422)*, vol. 3, pp. 3473–3478, IEEE, 2003.

[34] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3936–3943, IEEE, 2014.

[35] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3665–3671, IEEE, 2020.

[36] O. Mees, N. Abdo, M. Mazuran, and W. Burgard, "Metric learning for generalizing spatial relations to new objects," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3175–3182, IEEE, 2017.

[37] P. Jund, A. Eitel, N. Abdo, and W. Burgard, "Optimization beyond the convolution: Generalizing spatial relations with end-to-end metric learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4510–4516, IEEE, 2018.

[38] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2642–2651, 2019.

[39] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *arXiv preprint arXiv:1806.08756*, 2018.

[40] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2019.

[41] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg, "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9411–9418, IEEE, 2020.

[42] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9404–9410, IEEE, 2020.

[43] M. Khansari, D. Kappler, J. Luo, J. Bingham, and M. Kalakrishnan, "Action image representation: Learning scalable deep grasping policies with zero real world data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3597–3603, IEEE, 2020.

[44] C. Devin, P. Rowghanian, C. Vigorito, W. Richards, and K. Rohanimanesh, "Self-supervised goal-conditioned pick and place," *arXiv preprint arXiv:2008.11466*, 2020.

[45] L. Berscheid, P. Meißner, and T. Kröger, "Self-supervised learning for precise pick-and-place without object model," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4828–4835, 2020.

[46] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4568–4575, IEEE, 2021.

[47] H. Huang, D. Wang, R. Walters, and R. Platt, "Equivariant transporter network," *arXiv preprint arXiv:2202.09400*, 2022.

[48] H. Huang, O. Howell, X. Zhu, D. Wang, R. Walters, and R. Platt, "Fourier transporter: Bi-equivariant robotic manipulation in 3d," *arXiv preprint arXiv:2401.12046*, 2024.

[49] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on robot learning*, pp. 894–906, PMLR, 2022.

[50] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[51] S. Zhang, P. Wicke, L. K. Şenel, L. Figueredo, A. Naceri, S. Haddadin, B. Plank, and H. Schütze, "Lohoravens: A long-horizon language-conditioned benchmark for robotic tabletop manipulation," *arXiv preprint arXiv:2310.12020*, 2023.

[52] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 558–567, October 2021.

[53] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10076–10085, 2020.

[54] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, pp. 213–229, Springer, 2020.

[55] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6881–6890, 2021.

[56] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li, "Uformer: A general u-shaped transformer for image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 17683–17693, 2022.

[57] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning texture transformer network for image super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5791–5800, 2020.

[58] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12299–12310, 2021.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[60] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International conference on machine learning*, pp. 5156–5165, PMLR, 2020.

[61] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[62] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[63] A. Trockman and J. Z. Kolter, "Patches are all you need?," *arXiv preprint arXiv:2201.09792*, 2022.

[64] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proceedings of The 6th Conference on Robot Learning* (K. Liu, D. Kulic, and J. Ichnowski, eds.), vol. 205 of *Proceedings of Machine Learning Research*, pp. 785–799, PMLR, 14–18 Dec 2023.

[65] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[66] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "Rvt: Robotic view transformer for 3d object manipulation," in *Proceedings of The 7th Conference on Robot Learning* (J. Tan, M. Toussaint, and K. Darvish, eds.), vol. 229 of *Proceedings of Machine Learning Research*, pp. 694–710, PMLR, 06–09 Nov 2023.

[67] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-z: Zero-shot task generalization with robotic imitation learning," in *5th Annual Conference on Robot Learning*, 2021.

[68] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 22955–22968, Curran Associates, Inc., 2022.

[69] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[70] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*, pp. 2165–2183, PMLR, 2023.

[71] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[72] G. B. Arfken, H.-J. Weber, and F. E. Harris, *Mathematical Methods for Physicists*, ch. 15.5 Convolution Theorem, p. 810–814. Elsevier, 3rd ed., 1985.

Appendix A

Extended Results & Network Specifications

A.1 Transporter Networks Model Specification

Transporter Networks feature ResNet [59] backbones in an encoder-decoder architecture. The network comprises two arms, attention and transport, where the transport arm consists of two sub-components for key and query models. Details of the convolutional backbone model can be found in Table A.1. The scene o_t is a merged view of RGBD images from three different fixed viewpoints. To balance color and depth channels, the depth channel is duplicated twice, resulting in an input shape of $(H, W, C) = (160, 320, 6)$. The height dimension is padded with zeros to match the length of the width dimension, providing a square input. This square input is directly processed by the attention backbone. The key component of the transport arm processes a further padded input of shape $(H, W, C) = (384, 384, 6)$, allowing crops of $(64, 64)$ from this padded o_t to always contain valid values, with cross-correlation

matching the original scene dimensions. These $(64, 64)$ crops are then processed by the query component of the transport arm prior to cross-correlation.

Table A.1: The Transporter Network's architectural details, as utilized in the original work, involve several stages. The parameter S represents the size along a given dimension, while C denotes 2-dimensional convolutional layers. A numerical prefix indicates the number of times a layer is applied. Specifically, the CS layer consists of a 2D convolutional layer with a shortcut connection between the start and end layers, utilizing a convolutional operation in the shortcut, whereas the CI layer represents a convolutional layer with a shortcut identity connection linking the start and end layers. Through an iterative process, the input scene is encoded into progressively deeper yet smaller feature embeddings, then decoded into the target action-value map via alternating U upsampling layers and convolutional blocks.

Stage	Layer Configuration	Embedding Dimension	Feature Dimensions (height, width)
0	input	6	(S, S)
1	$1C$	64	(S, S)
2	$3CS + 3CI$	64	(S, S)
3	$3CS + 3CI$	128	$(S/2, S/2)$
4	$3CS + 3CI$	256	$(S/4, S/4)$
5	$3CS + 3CI$	512	$(S/8, S/8)$
6	$3CS + 3CI$	256	$(S/8, S/8)$
7	U	256	$(S/4, S/4)$
8	$3CS + 3CI$	128	$(S/4, S/4)$
9	U	128	$(S/2, S/2)$
10	$3CS + 3CI$	64	$(S/2, S/2)$
11	U	64	(S, S)
12	$3CS + 3CI$	1 or 3	(S, S)
13	output	1 or 3	(S, S)

A.2 Single Task Training Across All Checkpoints

The single-task training results for TpTf and SpTpTf, benchmarked against the baseline Transporter Networks, are presented in Table A.2. These figures represent averages over all training checkpoints and complement the final 10% checkpoint-specific results reported in Table 4.1.

Table A.2: Single-task training success rate results are reported as mean (standard deviation) percentage points. All models undergo training in an identical environment. In this training scheme, one model is trained per task for each number of demonstrations, with each training session lasting 40,000 iterations, totaling 1,600,000 iterations for each architecture. Symbol † denotes our models. Form2Fit results are taken from [42].

Task	**align-box-corner**				**assembling-kits**			
Num-Demos	1	10	100	1000	1	10	100	1000
SpTpTf †	3.9 (18.7)	70.3 (43.4)	93.4 (18.2)	98.2 (5.9)	24.6 (19.9)	71.0 (24.1)	86.2 (18.6)	89.5 (16.8)
TpTf †	6.4 (24.0)	78.2 (38.3)	94.7 (18.0)	97.2 (11.2)	21.5 (19.9)	67.4 (25.7)	84.8 (19.5)	87.3 (17.0)
Transporter	16.9 (36.7)	57.9 (47.9)	92.0 (24.8)	97.6 (10.6)	17.0 (17.9)	58.1 (23.5)	80.5 (20.7)	84.0 (17.9)
Form2Fit [42]	7.0	2.0	5.0	16.0	3.4	7.6	24.2	37.6
	block-insertion				**manipulating-rope**			
	1	10	100	1000	1	10	100	1000
SpTpTf †	22.6 (41.6)	99.3 (5.5)	99.7 (2.5)	100.0 (0.5)	0.7 (3.3)	66.3 (35.9)	83.8 (24.0)	84.3 (22.8)
TpTf †	41.5 (49.0)	99.5 (4.0)	99.8 (1.7)	99.8 (1.6)	0.7 (3.8)	68.0 (32.9)	80.7 (25.4)	81.6 (23.3)
Transporter	94.5 (21.3)	97.8 (12.9)	99.8 (1.4)	100.0 (0.5)	8.5 (19.9)	67.9 (37.8)	87.5 (24.5)	83.7 (25.1)
Form2Fit [42]	17.0	19.0	23.0	29.0	11.9	38.8	36.7	47.7
	packing-boxes				**palletizing-boxes**			
	1	10	100	1000	1	10	100	1000
SpTpTf †	70.7 (27.0)	97.2 (10.1)	99.6 (2.5)	99.7 (2.1)	86.7 (13.0)	96.2 (7.3)	97.2 (4.9)	97.4 (5.1)
TpTf †	81.7 (23.2)	97.3 (8.1)	98.9 (5.0)	99.3 (3.1)	89.8 (17.6)	97.1 (6.3)	97.0 (5.4)	98.1 (4.6)
Transporter	87.7 (21.4)	97.7 (9.1)	99.2 (4.0)	99.1 (5.6)	78.5 (12.3)	92.4 (7.3)	93.4 (8.9)	94.4 (8.3)
Form2Fit [42]	29.9	52.5	62.3	66.8	21.6	42.0	52.1	65.3
	place-red-in-green				**stack-block-pyramid**			
	1	10	100	1000	1	10	100	1000
SpTpTf †	46.6 (43.7)	97.6 (13.0)	99.9 (1.0)	100.0 (0.2)	2.4 (6.3)	50.7 (26.9)	79.8 (24.0)	82.7 (20.4)
TpTf †	62.4 (40.3)	97.5 (13.3)	99.3 (4.6)	99.9 (1.1)	7.7 (11.3)	71.2 (27.7)	70.7 (19.3)	73.1 (16.0)
Transporter	64.0 (41.8)	99.8 (1.6)	99.7 (2.2)	99.9 (1.2)	10.7 (12.6)	52.9 (27.7)	70.8 (19.6)	75.1 (17.7)
Form2Fit [42]	83.4	100.0	100.0	100.0	19.7	17.5	18.5	32.5
	sweeping-piles				**sweeping-piles**			
	1	10	100	1000	1	10	100	1000
SpTpTf †	97.3 (5.9)	99.6 (1.4)	99.7 (1.2)	99.4 (1.8)	97.3 (5.9)	99.6 (1.4)	99.7 (1.2)	99.4 (1.8)
TpTf †	98.3 (3.6)	99.0 (2.7)	99.0 (3.0)	98.1 (4.0)	98.3 (3.6)	99.0 (2.7)	99.0 (3.0)	98.1 (4.0)
Transporter	97.6 (5.8)	98.5 (5.4)	97.7 (7.3)	98.0 (7.0)	97.6 (5.8)	98.5 (5.4)	97.7 (7.3)	98.0 (7.0)
Form2Fit [42]	13.2	15.6	26.7	38.4	13.2	15.6	26.7	38.4

A.3 Multi Task Training Across All Checkpoints

The multi-task training results for TpTf and SpTpTf, compared with the baseline Transporter Networks, are presented in Table A.3. These values are averaged over all training checkpoints and serve as a complement to the final 10% checkpoint-specific analysis shown in Table 4.2.

Table A.3: Multi-task training success rate results are reported as mean (standard deviation) percentage points. Symbol † denotes our models.

Task		**align-box-corner**				**assembling-kits**			
κ	Num-Demos	1	10	100	1000	1	10	100	1000
1	SpTpTf †	26.0 (43.2)	65.9 (43.0)	83.1 (28.0)	83.8 (27.4)	6.7 (10.9)	29.0 (21.2)	45.5 (24.0)	45.7 (24.0)
	TpTf †	16.1 (35.9)	68.3 (42.1)	86.2 (24.1)	88.6 (21.6)	7.2 (11.6)	39.9 (24.5)	54.6 (25.4)	53.1 (26.2)
	Transporter	42.8 (47.1)	70.7 (38.2)	85.9 (20.2)	83.9 (21.7)	11.4 (14.1)	40.4 (23.1)	41.1 (22.3)	37.8 (21.5)
10	SpTpTf †	20.6 (38.9)	56.9 (43.2)	74.6 (29.6)	77.2 (29.5)	4.8 (9.3)	27.4 (20.9)	43.3 (23.6)	41.9 (23.1)
	TpTf †	40.6 (46.8)	58.5 (45.1)	83.8 (23.8)	84.7 (21.3)	6.9 (11.3)	38.9 (24.2)	53.1 (24.6)	51.1 (25.3)
	Transporter	36.4 (40.8)	62.1 (32.9)	76.7 (20.7)	73.5 (23.4)	8.2 (11.3)	24.8 (20.1)	22.7 (17.5)	24.8 (18.3)
100	SpTpTf †	24.5 (38.0)	48.4 (36.6)	50.0 (27.8)	54.5 (28.6)	5.0 (8.8)	23.1 (18.0)	27.4 (18.5)	29.4 (18.6)
	TpTf †	12.2 (30.0)	58.5 (38.5)	71.2 (25.1)	72.5 (23.6)	4.8 (8.7)	25.1 (18.6)	35.0 (21.0)	34.8 (19.1)
	Transporter	20.9 (33.6)	41.0 (31.8)	51.5 (32.0)	42.8 (28.1)	1.5 (4.0)	7.5 (7.0)	10.8 (10.4)	12.2 (9.8)
		block-insertion				**manipulating-rope**			
		1	10	100	1000	1	10	100	1000
1	SpTpTf †	89.6 (27.5)	98.2 (7.9)	98.8 (6.2)	99.0 (4.2)	0.9 (4.5)	26.9 (32.6)	45.4 (33.8)	45.8 (31.3)
	TpTf †	97.6 (9.6)	99.0 (4.5)	98.6 (4.2)	98.9 (4.8)	4.0 (12.5)	53.0 (34.4)	54.5 (29.5)	63.8 (28.0)
	Transporter	90.3 (23.0)	94.8 (13.4)	96.4 (9.0)	95.5 (9.8)	35.1 (37.6)	59.6 (34.8)	60.5 (31.2)	63.5 (30.8)
10	SpTpTf †	84.3 (33.2)	97.3 (9.1)	98.3 (4.9)	98.9 (4.2)	0.7 (3.6)	26.6 (27.5)	33.2 (28.6)	34.5 (25.5)
	TpTf †	95.9 (11.4)	97.5 (7.0)	98.0 (6.7)	98.0 (7.1)	3.2 (10.9)	42.4 (31.4)	49.6 (27.9)	50.4 (29.6)
	Transporter	81.2 (24.6)	84.5 (18.3)	90.0 (10.3)	92.0 (12.6)	25.7 (31.4)	46.7 (29.2)	46.8 (26.6)	48.1 (25.6)
100	SpTpTf †	77.5 (29.4)	93.3 (9.1)	96.6 (5.6)	97.3 (6.5)	1.0 (4.5)	11.4 (14.0)	14.4 (14.3)	17.7 (13.6)
	TpTf †	76.7 (31.6)	93.4 (12.9)	96.8 (6.3)	97.2 (9.4)	1.3 (5.2)	19.4 (16.3)	29.9 (16.4)	21.0 (12.2)
	Transporter	59.8 (28.4)	62.9 (21.9)	82.2 (18.2)	64.2 (20.3)	13.5 (19.3)	17.4 (15.2)	20.4 (13.3)	21.6 (16.5)
		packing-boxes				**palletizing-boxes**			
		1	10	100	1000	1	10	100	1000
1	SpTpTf †	91.6 (17.9)	96.6 (10.2)	98.9 (4.1)	98.9 (3.6)	61.2 (12.2)	61.6 (13.2)	60.0 (13.3)	57.7 (14.0)
	TpTf †	95.6 (12.3)	96.7 (10.2)	99.0 (4.3)	99.2 (4.4)	77.9 (12.9)	77.2 (10.2)	81.2 (10.4)	85.7 (9.7)
	Transporter	87.0 (20.8)	96.1 (11.3)	98.7 (4.3)	98.6 (4.5)	65.8 (13.0)	65.9 (14.0)	62.6 (12.7)	69.6 (12.5)
10	SpTpTf †	91.8 (15.5)	97.2 (8.1)	98.3 (4.6)	98.7 (4.7)	63.5 (12.3)	61.5 (13.3)	67.9 (13.1)	60.5 (13.4)
	TpTf †	92.6 (16.9)	96.7 (10.8)	99.3 (2.9)	99.1 (4.1)	76.1 (11.7)	79.4 (10.4)	78.5 (10.9)	82.8 (9.8)
	Transporter	86.2 (21.0)	96.1 (9.6)	96.6 (7.5)	94.9 (9.3)	60.0 (12.2)	63.5 (12.8)	60.3 (14.0)	68.9 (12.1)
100	SpTpTf †	80.6 (21.4)	91.8 (11.4)	89.0 (13.4)	86.5 (12.4)	59.0 (11.8)	51.3 (12.1)	57.7 (12.4)	59.4 (12.5)
	TpTf †	87.3 (15.9)	94.4 (10.9)	94.5 (5.1)	95.8 (6.4)	62.7 (11.8)	71.8 (10.7)	71.1 (9.8)	73.6 (10.3)
	Transporter	61.7 (27.0)	71.3 (20.0)	81.2 (17.8)	74.7 (18.3)	43.5 (11.5)	43.0 (11.3)	35.5 (9.9)	38.3 (10.9)
		place-red-in-green				**stack-block-pyramid**			
		1	10	100	1000	1	10	100	1000
1	SpTpTf †	47.5 (43.2)	76.5 (36.9)	91.0 (22.2)	87.0 (27.4)	5.0 (8.8)	9.4 (10.7)	10.4 (12.0)	12.9 (13.3)
	TpTf †	70.8 (38.7)	90.5 (19.5)	95.9 (6.8)	96.2 (5.6)	7.8 (11.5)	28.0 (24.4)	43.3 (26.0)	46.6 (26.0)
	Transporter	66.6 (41.3)	93.8 (17.3)	98.5 (6.2)	97.7 (8.2)	6.8 (10.8)	13.6 (14.2)	15.9 (13.9)	18.1 (13.9)
10	SpTpTf †	42.9 (42.3)	74.2 (36.6)	83.8 (27.7)	86.8 (25.9)	5.3 (9.0)	10.7 (11.9)	9.9 (11.9)	10.2 (11.4)
	TpTf †	71.3 (39.0)	89.5 (21.2)	92.4 (11.0)	92.3 (8.9)	13.8 (15.6)	26.7 (23.0)	39.9 (23.7)	42.3 (23.8)
	Transporter	63.1 (40.4)	88.3 (18.8)	94.1 (12.4)	92.6 (14.2)	4.8 (8.7)	12.1 (12.7)	14.6 (11.8)	13.8 (11.1)
100	SpTpTf †	42.8 (40.1)	57.5 (33.1)	64.9 (28.5)	64.3 (31.9)	4.3 (7.8)	10.3 (12.3)	10.2 (11.5)	10.1 (11.3)
	TpTf †	48.0 (41.3)	80.8 (24.3)	79.6 (16.1)	79.2 (17.4)	2.7 (6.4)	24.3 (21.6)	33.4 (23.0)	33.3 (22.9)
	Transporter	45.5 (31.2)	52.7 (27.7)	70.1 (22.4)	70.0 (25.3)	1.5 (3.9)	11.3 (9.8)	7.0 (8.7)	9.5 (8.6)
		sweeping-piles				**towers-of-hanoi**			
		1	10	100	1000	1	10	100	1000
1	SpTpTf †	85.5 (15.8)	86.6 (12.8)	93.8 (7.9)	87.3 (9.5)	21.0 (19.0)	35.2 (19.7)	33.3 (19.3)	34.8 (21.2)
	TpTf †	91.4 (11.9)	92.3 (10.7)	93.5 (10.5)	91.4 (10.5)	38.4 (26.3)	63.1 (21.4)	60.9 (19.2)	62.7 (20.1)
	Transporter	88.1 (20.6)	85.4 (24.6)	86.6 (23.0)	88.1 (20.8)	28.7 (23.5)	33.5 (15.1)	29.7 (15.8)	25.2 (10.8)
10	SpTpTf †	83.4 (16.2)	77.6 (15.8)	82.1 (16.7)	82.7 (13.3)	12.5 (13.2)	32.7 (21.4)	23.9 (14.4)	36.6 (18.6)
	TpTf †	91.1 (11.5)	84.6 (15.1)	88.5 (11.7)	87.0 (13.2)	31.3 (24.3)	60.2 (19.1)	63.6 (18.3)	55.6 (18.5)
	Transporter	84.3 (17.5)	79.8 (25.1)	82.8 (25.7)	85.5 (23.1)	28.2 (18.6)	21.5 (13.4)	20.0 (11.6)	20.8 (11.0)
100	SpTpTf †	69.6 (16.8)	56.3 (18.7)	57.8 (16.6)	53.7 (18.8)	12.9 (11.6)	22.3 (14.1)	25.5 (13.8)	20.4 (13.1)
	TpTf †	64.0 (21.5)	68.4 (17.2)	65.3 (19.2)	61.8 (18.9)	19.0 (16.1)	45.5 (16.4)	46.0 (14.3)	47.4 (13.9)
	Transporter	61.4 (18.2)	55.9 (25.6)	49.4 (23.7)	49.7 (22.8)	7.2 (6.5)	7.0 (4.6)	8.0 (4.0)	5.7 (2.8)

Table A.4: Success rates reported as mean (standard deviation) percentage points. The results are averaged across all tasks and are representative of the prevailing performance. Symbol † denotes our models.

Task	**All Tasks - All Checkpoints**			
Num-Demos	1	10	100	1000
TpTf †	50.8 (19.6)	87.4 (16.2)	92.4 (10.5)	93.3 (8.6)
SpTpTf †	45.3 (18.5)	84.8 (16.9)	93.9 (9.8)	95.1 (7.7)
Transporter	57.3 (19.6)	82.2 (17.9)	91.8 (12.1)	93.0 (10.1)
Form2Fit [42]	22.0	31.1	37.5	47.2

A.4 Summarized Training Results

Tables A.4 and A.5 summarize the single-task results (reported in percentage points) across all benchmarks. The first table aggregates performance over the final 10% of checkpoints, while the second uses the entire checkpoint set. In both cases, TpTf models achieve performance that is comparable to, and often exceeds, the baseline. The gap becomes more pronounced when we focus on the final checkpoints, where training has converged and the models are most stable.

Tables A.6 and A.7 report the multi-task results (in percentage points) for all tasks. The first table aggregates performance over the final 10% of checkpoints, whereas the second uses the entire checkpoint set. The TpTf model consistently outperforms the baseline, while the SpTpTf variant achieves results that are comparable to the baseline. The performance gap becomes more pronounced when we restrict the analysis to the final checkpoints, where training has converged and the models are most stable.

Table A.5: Success rates reported as mean (standard deviation) percentage points. The "-L4" notation denotes results over the last four episodes, which correspond to checkpoints taken during the final 10% of training iterations where models converge closer to their final performance. The results are averaged across all tasks and are representative of the prevailing performance.

Task	**All Tasks - 10% Checkpoints**			
Num-Demos	1	10	100	1000
TpTf-L4 †	50.9 (20.3)	89.7 (16.8)	96.5 (10.9)	98.7 (5.5)
SpTpTf-L4 †	45.3 (19.9)	86.2 (17.7)	96.9 (9.3)	98.0 (7.2)
Transporter-L4	55.5 (23.5)	84.4 (19.5)	96.8 (9.5)	97.1 (7.4)
Form2Fit [42]	21.1	29.9	35.2	44.0

Table A.6: Multi-task training success rate results are reported as mean (standard deviation) percentage points. The results are averaged over all tasks and is representative of prevailing performance. Symbol † denotes our models.

Task		**All Tasks - All Checkpoints**			
κ	Num-Demos	1	10	100	1000
1	TpTf †	50.7 (18.3)	70.8 (20.2)	76.8 (16.0)	78.6 (15.7)
	SpTpTf †	43.5 (20.3)	58.6 (20.8)	66.0 (17.1)	65.3 (17.6)
	Transporter	52.3 (25.2)	65.4 (20.6)	67.6 (15.9)	67.8 (15.5)
10	TpTf †	52.3 (19.9)	67.4 (20.7)	74.7 (16.1)	74.3 (16.2)
	SpTpTf †	41.0 (19.3)	56.2 (20.8)	61.5 (17.5)	62.8 (17.0)
	Transporter	47.8 (22.6)	57.9 (19.3)	60.5 (15.8)	61.5 (16.1)
100	TpTf †	37.9 (18.9)	58.2 (18.7)	62.3 (15.6)	61.7 (15.4)
	SpTpTf †	37.7 (19.0)	46.6 (17.9)	49.3 (16.2)	49.3 (16.7)
	Transporter	31.6 (18.4)	37.0 (17.5)	41.6 (16.0)	38.9 (16.3)

Table A.7: Multi-task training success rate results are reported as mean (standard deviation) percentage points. The "-L4" notation denotes results over the last four episodes, which correspond to checkpoints taken during the final 10% of training iterations where models converge closer to their final performance. The results are averaged over all tasks and is representative of prevailing performance. Symbol † denotes our models.

Task		**All Tasks - 10% Checkpoints**			
κ	Num-Demos	1	10	100	1000
1	TpTf †	54.3 (19.6)	82.4 (20.7)	88.7 (15.8)	91.0 (14.3)
	SpTpTf †	47.0 (20.5)	67.0 (23.7)	78.3 (15.7)	76.0 (18.0)
	Transporter	55.4 (27.1)	78.9 (20.0)	81.7 (14.7)	77.6 (14.6)
10	TpTf †	59.4 (20.9)	77.5 (23.3)	90.8 (13.6)	90.5 (14.0)
	SpTpTf †	46.0 (20.0)	68.8 (22.3)	75.1 (16.5)	79.8 (17.0)
	Transporter	53.7 (25.9)	71.0 (20.7)	75.7 (15.0)	79.4 (16.5)
100	TpTf †	46.2 (21.7)	79.6 (20.8)	86.7 (16.1)	82.9 (16.2)
	SpTpTf †	47.3 (21.9)	61.0 (20.7)	69.7 (18.8)	64.9 (18.9)
	Transporter	45.3 (22.6)	58.1 (23.9)	59.5 (16.5)	61.6 (18.3)