Visuomotor Control for Single Task Robotic Manipulation via Text-Conditioned Flow Matching

Somdeb Saha
TCS Research
Tata Consultancy Services
Bangalore,KA, India
somdeb.saha@tcs.com

Vighnesh Vatsal
TCS Research
Tata Consultancy Services
Bangalore,KA, India
vighnesh.vatsal@tcs.com

Kaushik Das
TCS Research
Tata Consultancy Services
Bangalore,KA, India
kaushik.da@tcs.com

***Abstract*—This paper introduces Text-Conditioned Flow Matching Policy (TC-FMP), a continuous-time generative imitation learning framework that leverages natural language as a lightweight yet effective guidance signal for visuomotor control. TC-FMP learns an instruction-conditioned velocity field that guarantees the smooth evolution of trajectories from noise toward the demonstrated action distribution. To mitigate mode collapse and improve conditional fidelity, classifier-free guidance is incorporated by randomly dropping textual conditioning during training. At inference time, guided and unguided vector fields are combined to produce a scaled velocity field, which is integrated using a second-order Heun solver to generate action sequences. The resulting action trajectories are executed in a receding-horizon control scheme, enabling closed-loop execution over long horizons. Language is fused with visual and proprioceptive inputs via FiLM modulation, introducing negligible architectural overhead while allowing natural language prompts to steer task-relevant behaviors. TC-FMP is first evaluated on the planar PushT benchmark, and then to demonstrate its robustness and practical utility; on a long-horizon, contact-rich bimanual peg-in-hole insertion task under varying noise conditions. Experimental results show that increasingly specific language instructions consistently improve both average and best-case returns compared to diffusion-based and flow-matching baselines. These findings highlight that even single-task visuomotor control can substantially benefit from language-driven guidance, improving robustness, controllability, and performance.**

***Index Terms*—Flow Matching, Classifier free Guidance, Bimanual Manipulation**

I. Introduction

Learning policies for contact-rich robotic manipulation from demonstrations have advanced rapidly with the use of large-capacity sequence models and generative objectives. Diffusion-based policies [1] have proven particularly effective for visuomotor imitation learning, allowing expressive multimodal action distributions conditioned on high-dimensional sensory input [2], [3]. However, their reliance on many denoising steps for inference introduces significant rollout latency, limiting suitability for closed-loop control. This was addressed using deterministic sampling, proposed by [4].

Flow matching [5] has recently emerged as a compelling alternative, learning a continuous-time vector field that transports samples from an initial distribution to the data distribution with fewer integration steps and a simpler training objective. For robotic control, this continuous-time formulation naturally supports trajectory-level prediction and receding-horizon execution, where policies repeatedly replan actions based on updated observations [6]. Flow-based policies could also be used in assistive robot manipulation tasks taking affordance into consideration [7]. Flow matching also serves as the backbone for generalist Vision Language Action (VLA) models such as $\pi_{0.5}$ [8]. Although more numerically stable than diffusion, it suffers from mode collapse. This was solved by conditioning the model on the sampling step size along with the timestep [9]. Streaming Flow Policy [10] enables faster policy execution times by modeling action trajectories as flow trajectories.

Concurrently, natural language has emerged as a practical interface for specifying tasks and constraints. Although language-conditioned policies are typically explored in multi-task settings [8], even single-task manipulation can benefit from language [11] as a lightweight, high-level specification of desired behavior, such as the target, motion style, or success criterion, without modifying the underlying low-level control interface. [12] generates a flow conditioned policy using visual features from a simulated data based on task description. [13] uses semantic scene descriptions generated from language models for robotic manipulation in cluttered spaces while [14] uses language instructions as the first level of a hierarchal diffusion based planner.

The velocity field that is learned by the neural network must be conditioned on input images and the language instructions [15]. Images can be RGB [15] or RGB-D [16] from single [17] or multiple camera viewpoints [18]. The conditioning method most commonly used is FiLM [19] (Feature-wise Linear Modulation) due to its simplicity in [2] and [11]. However, in order to capture high dimensional data and long range sequences, cross attention was preferred by [20] and [21]. The most common choice of architecture for the target neural network is U-Net [22], used by [7] and [10]. Alternatively, Diffusion Transformers (DiT) are also used as they do not suffer from inductive biases [15], [16].

Classifier-Free Guidance (CFG), a popular method for conditioning inspired by conditional generative modeling, has been shown to improve sample quality during sampling [23]. During training, some conditions are randomly dropped to null, to induce unconditional behavior. At inference time

the conditional and unconditional vector fields are combined linearly to produce a resultant vector field. CFG has been used to improve constrained manipulation by conditioning on state, start and end goals [24]. [25] used CFG conditioned on goals to outperform several baselines on manipulation benchmarks.

Bimanual manipulation is a fundamental motor capability underlying human tool use. However, replicating even seemingly simple tasks, such as knot tying or package unboxing, remains a major challenge for robotic systems due to the need for precise contact-rich perception, long-horizon planning, and tightly coupled control. By enabling simultaneous grasping and coordinated interaction, bimanual robotic systems offer substantial advantages over single-arm platforms in terms of task efficiency, dexterity, and object-handling flexibility. Consequently, bimanual manipulation has emerged as a particularly demanding and important problem in robotics research. Recent advances in Imitation Learning (IL), especially approaches based on diffusion models [2] and flow matching [26], have demonstrated improved performance over classical methods such as Dynamic Movement Primitives [27], particularly in complex, high-dimensional manipulation tasks. Building on these developments, this paper proposes a novel IL framework designed to address contact-rich bimanual manipulation through the integration of flow matching, natural language instructions, and classifier-free guidance. In the proposed approach, bimanual action sequences are generated by integrating a learned velocity field trained using flow matching, with stochastic dropping of text conditioning to enable classifier-free guidance at inference time. Natural language task descriptions are encoded using a compact sentence encoder (SentenceBERT) [28], while visual observations are processed through standard vision encoders [29]. The resulting multimodal representations are fused via Feature-wise Linear Modulation (FiLM) conditioning [19], allowing language to modulate perception and control with minimal additional computational overhead. This design enables flexible, instruction-conditioned planning and execution for complex bimanual manipulation tasks involving rich physical interactions. The contributions of this paper can be summarized as follows:

- A text-conditioned flow matching policy (TC-FMP) for action sequence generation, predicting in a closed-loop, receding horizon manner.
- A CFG-style guidance mechanism that uses language instructions for tasks, that improves trajectory quality under the same base neural network architecture.
- Empirical evaluation on a planar PushT benchmark task and a high-dimensional bimanual robotic manipulation task, with TC-FMP showing better performance in terms of average and maximum scores compared to other flow matching and diffusion-based approaches.

Data Availability Statement: The authors are governed by institutional policies regarding non-disclosure of intellectual property. The source code from this work, including trained model weights, are prohibited from being made publicly available. However, the data and the evaluation environment are open source, and sufficient details of the methodology are provided to make the work reproducible.

II. System Description

While flow-based generative models have traditionally been applied to image synthesis [5], they can be repurposed for action sequence prediction in an imitation learning context for robotics. In this section, we describe the robotic environments used for evaluating and benchmarking our text-conditioned flow matching policy (TC-FMP) against other similar methods for imitation learning.

Generative models aim to learn an underlying data distribution p_{data} such that, after training, samples can be drawn from this distribution. In flow models, this is achieved by learning a continuous transformation that maps samples from a simple initial distribution p_{init} to p_{data} over a time interval $t \in [0, 1]$. The initial distribution is typically chosen as an isotropic Gaussian, $p_{\text{init}} = \mathcal{N}(0, I_d)$. Assuming that human teleoperation data defines the target distribution p_{data}, the learned flow model can be used to generate action sequences consistent with expert behavior. To improve robustness, visual conditioning is enabled which allows for extraction of pixel-based environmental features that inform the action generation process. Given the robotics setting, a closed-loop, receding-horizon control formulation is considered, operating over action sequences rather than individual time steps. Specifically, all action sequences x lie in the action space

$$x \in \mathcal{A}^{\mathcal{T}} \subseteq \mathbb{R}^{H_p \times A_d},$$

where H_p denotes the prediction horizon and A_d the dimensionality of the action space. Similarly, all observation sequences(features) lie in the observation space

$$o \in \mathcal{O}^{\mathcal{T}} \subseteq \mathbb{R}^{H_o \times P_d} \times \mathbb{R}^{H_o \times s_d},$$

where H_o denotes the observation horizon, P_d the dimensionality of the pixel space which is usually $\mathbb{R}^{C \times H \times W}$ and s_d the dimensionality of robot state i.e end effector pose. At every timestep t discretized over the interval $[0, 1]$, the flow model takes in an isotropic gaussian $x_0 \sim \mathcal{N}(0, I_d)$, the latest observation features o_t (up to the observation horizon) and timestep t to produce a velocity field. This velocity field is then integrated using Euler sampling to generate an action sequence $x_1 \sim p_{data}$ over the prediction horizon H_p. To enable long horizon task execution, only action prediction H_a number of actions are executed in the environment without re-planning. This is repeated again for the next timestep and observation sequence. The proposed methodology is systematically evaluated in two environments, namely the PushT environment, and Bimanual Insertion.

A. PushT Environment

The PushT environment [2] is a widely used benchmark for evaluating learning-based manipulation algorithms due to its relative simplicity and low-dimensional state and action spaces. The task consists of pushing a T-shaped block (gray)

Fig. 1: PushT environment.

into a fixed target region (green) using a circular robot end-effector (blue). The start positions of the block and end-effector are randomized to induce variations. Despite the apparent simplicity, successful completion requires precise control under contact-rich and nonlinear object dynamics, as the object must be manipulated through point contacts. An illustration of the PushT environment is shown in Fig. 1. The dataset contains 200 paired samples of observations and actions. Each observation comprises an RGB image of the scene in $\mathbb{R}^{3\times96\times96}$ along with the 2D Cartesian position (x, y) of the robot end-effector. Actions correspond to target 2D positions (x, y) for the end-effector. The prediction horizon is set to $H_p = 16$, the observation horizon to $H_o = 2$, and the action horizon to $H_a = 8$. Formally, an action sequence is defined as

$$x \in \mathcal{A}^T \subseteq \mathbb{R}^{16\times2},$$

and an observation sequence as

$$o \in \mathcal{O}^T \subseteq \mathbb{R}^{2\times3\times96\times96} \times \mathbb{R}^{2\times2}.$$

B. Bimanual Insertion Environment

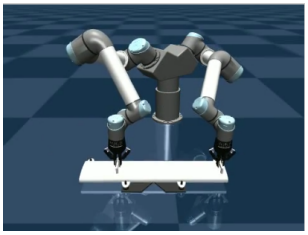

Fig. 2: Bimanual insertion environment.

The bimanual insertion environment [31] represents a significantly more challenging manipulation task due to its high-dimensional action and observation spaces and the use of a physics-based simulator (MuJoCo). The objective is to transfer a dynamic adapter with four holes and insert it into a stationary adapter equipped with four pegs. The task difficulty arises from tight mechanical tolerances: the holes have a diameter of 11 mm, while the rounded pegs have a base diameter of 10 mm, leaving approximately 1 mm of clearance. Successful execution therefore requires high precision. An example of the environment is shown in Fig. 2.

The dataset consists of 200 expert human demonstrations. During evaluation, the stationary adapter remains fixed, while the pose of the dynamic adapter is randomized. The action space comprises differential pose commands, namely translational and rotational increments $(\delta\rho,\ \delta\phi)$ applied to the end-effector. Rotational commands are represented using the 6D rotation representation (6DRR) [32], which provides a continuous mapping between $\mathbb{R}^6$ and $\mathcal{SO}(3)$. This results in 9-dimensional actions per arm, or 18 dimensions in total.

The observation space includes both proprioceptive and visual information. For each arm, the state consists of: (i) the difference between the expert hover pose over the stationary adapter and the current end-effector pose, (ii) the cube root of the distance between the end-effector and the nearest peg, (iii) forces and torques measured at the gripper sensors, and (iv) an RGB image of the scene. This results in an 18-dimensional state per arm, or 36 dimensions in total. The observation horizon is set to 4, the prediction horizon to 8, and the action horizon to 4.

Formally, an action sequence is defined as

$$x \in \mathcal{A}^\mathcal{T} \subseteq \mathbb{R}^{8\times18},$$

and an observation sequence as

$$o \in \mathcal{O}^\mathcal{T} \subseteq \mathbb{R}^{4\times3\times96\times96} \times \mathbb{R}^{4\times36}.$$

The environment reward used to evaluate algorithmic performance is defined as in [31], for consistent benchmarking:

$$R = \sum_{j=1}^{J} \left[e^{\gamma\left(\rho^j(t)-\rho^j_*(t)\right)^2} \ \oplus\ e^{\lambda\left(d\left(\phi^j(t),\phi^j_*(t)\right)\right)^2} \right] - \eta + \omega \tag{1}$$

Here, for each degree of freedom j, $x^j(t) - x^j_*(t)$ denotes the positional deviation between the end-effector and the expert demonstration at time t, and $d\Big(\phi^j(t), \phi^j_*(t)\Big)$ denotes the axis component of the quaternion difference, computed using the multiplicative inverse. The term η represents a time penalty, ω provides a positive reward upon successful insertion. $\oplus$ denotes concatenation of the six terms in the summand with the previous iteration, followed by averaging after iteration J.

III. Text-Conditioned Flow Matching

In this section, we describe the theoretical formulation for text-conditioned flow matching, as well as the architecture of the neural network model used to implement the policy. A flow model is given by

$$X_0 \sim p_{\text{init}}, \qquad \mathrm{d}X_t = u^\theta_t(X_t)\,\mathrm{d}t. \tag{2}$$

This ODE defines the velocity field (flow field) to drive the system from an initial distribution p_{init} to the data distribution p_{data}. p_{data} is the data distribution which cannot be explicitly modeled, but the dataset of $(observation,\ action)$ pairs acts as a sufficient surrogate since it allows for sampling. The goal is to have $X_1 \sim p_{\text{data}}$ such that the solution always converges to a sample from the data distribution. If only unconditional generation is considered, the neural network is defined as $u^\theta_t : \mathcal{A}^\mathcal{T} \times [0, 1] \to \mathcal{A}^\mathcal{T}$. In order to train the neural network $u^\theta_t(x)$,

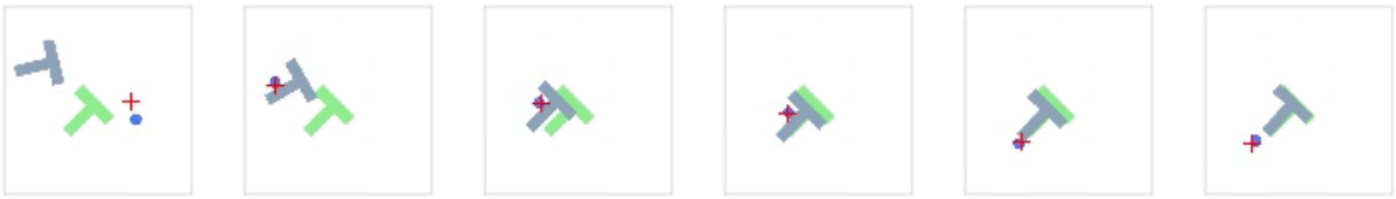

Fig. 3: Illustrative trajectory for successful completion of the PushT task.

Fig. 4: Illustrative trajectory for successful completion of the bimanual peg-in-hole insertion task.

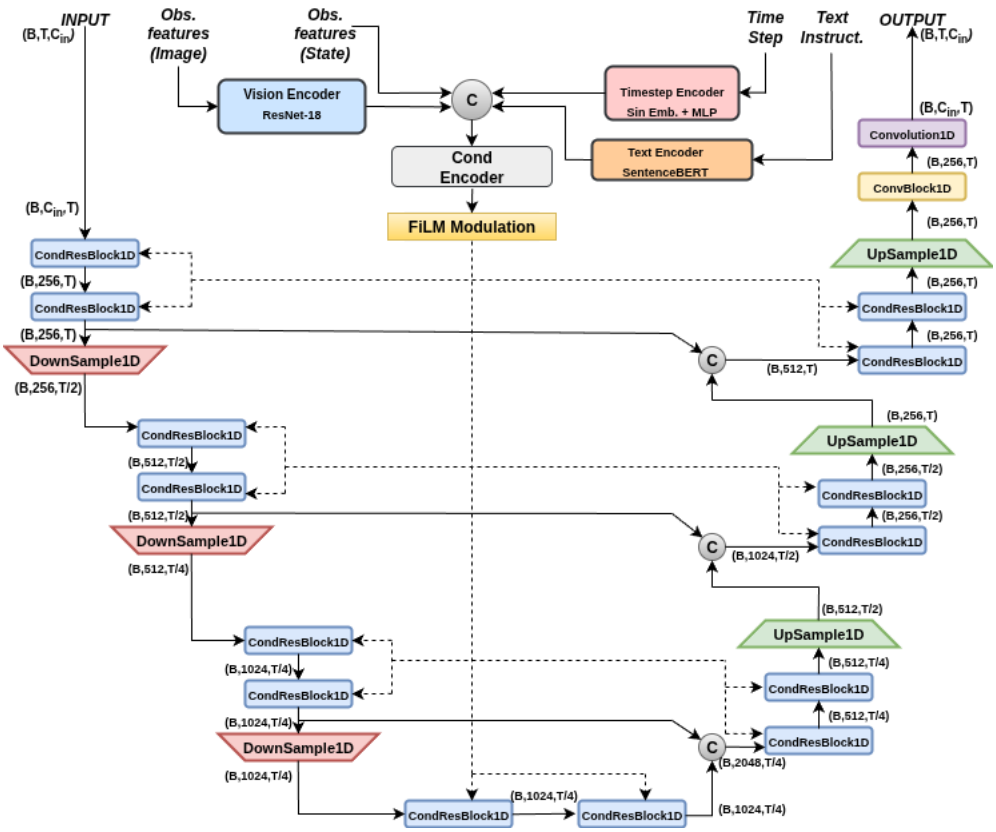

Fig. 5: Architecture of the Text-Conditioned Flow Matching Policy (TC-FMP).

it must be regressed against a known target $u_t^{\text{target}}(x)$. The flow matching loss is defined as

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\text{Unif}_{[0,1]},\, z\sim p_{\text{data}},\, x\sim p_t(\cdot|z)}\left[\|u_t^\theta(x) - u_t^{\text{target}}(x)\|^2\right] \tag{3}$$

Intuitively, the loss function follows these steps: (i) sample a random time uniformly $t \in [0, 1]$, (ii) draw a random point $z \in \mathcal{A}^{\mathcal{T}}$ from the dataset, followed by sampling $x \sim p_t(\cdot \mid z)$ and compute $u_t^\theta(x)$, (iii) finally, compute the mean-squared error between the output of the neural network and the target vector field $u_t^{\text{target}}(x)$. The initial step to construct a target involves defining probability paths which interpolate between p_{init} and p_{data}.

$p_t(x \mid z)$ is a conditional probability path, which is a set of distributions $p_t(x \mid z)$ over $\mathcal{A}^{\mathcal{T}}$ such that

$$p_0(\cdot \mid z) = p_{\text{init}}, \qquad p_1(\cdot \mid z) = \delta_z \quad \text{for all } z \in \mathcal{A}^{\mathcal{T}}. \tag{4}$$

Since the denoising diffusion models leverage Gaussian probability paths, it serves as an ideal candidate for this purpose as well. The Gaussian conditional probability path is defined as

$$p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d), \tag{5}$$

where α_t, β_t are noise schedulers that are monotonic in nature and continuously differentiable with the conditions $\alpha_0 = \beta_1 = 0$ and $\alpha_1 = \beta_0 = 1$. In the special case of Gaussian conditional optimal transport, the probability path can be constructed by taking $\alpha_t = t$ and $\beta_t = 1 - t$. It can be be seen that it satisfies the conditions from (4)

$$p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I_d) = \mathcal{N}(tz, (1-t) I_d) \tag{6}$$
$$p_0(\cdot \mid z) = \mathcal{N}(0 * z, 1 * I_d) = \mathcal{N}(0, I_d) \tag{7}$$
$$p_1(\cdot \mid z) = \mathcal{N}(1 * z, 0 * I_d) = \delta_z \tag{8}$$

Every conditional probability path induces a marginal probability path $p_t(x)$ which is determined by sampling a data point $z \sim p_{data}$ followed by sampling from $p_t(\cdot \mid z)$.

$$p_t(x) = \int p_t(x \mid z)\, p_{\text{data}}(z)\, \mathrm{d}z. \tag{9}$$

The above integral is intractable. However sampling can be done from the marginal probability path. After $z \sim p_{data}$ and using the reparameterization trick:

$$x \sim p_t(\cdot \mid z) = \alpha_t z + \beta_t \epsilon, \qquad \epsilon \sim (0, I_d) \tag{10}$$

It can be seen that the marginal probability path interpolates between p_{init} and p_{data} from $t = 0$ to $t = 1$.

For every data sample $z \in \mathcal{A}^{\mathcal{T}}$, a conditional vector field $u_t^{\text{target}}(\cdot \mid z)$ is defined such that the solution to the ODE yields the conditional probability path $p_t(\cdot \mid z)$ with $X_0 \sim p_{\text{init}}$

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^{\text{target}}(X_t \mid z) \implies X_t \sim p_t(\cdot \mid z) \quad \forall t \in [0, 1] \tag{11}$$

The conditional Gaussian vector field is given by [33]

$$u_t^{\text{target}}(x \mid z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z + \frac{\dot{\beta}_t}{\beta_t} x, \tag{12}$$

Using the marginalization trick, the marginal vector field $u_t^{\text{target}}(x)$ can be defined as

$$u_t^{\text{target}}(x) = \int u_t^{\text{target}}(x \mid z)\, \frac{p_t(x \mid z)\, p_{\text{data}}(z)}{p_t(x)}\, \mathrm{d}z \tag{13}$$

The marginal vector vector field is defined such that the solution to the ODE yields the marginal probability path $p_t(x)$.

$$X_0 \sim p_{\text{init}}, \quad \frac{\mathrm{d}}{\mathrm{d}t} X_t = u_t^{\text{target}}(X_t) \implies X_t \sim p_t \quad \forall t \in [0, 1] \tag{14}$$

In conclusion, the marginal vector field drives the system from p_{init} to p_{data} since $X_1 \sim p_{\text{data}}$. Hence it is the ideal candidate for u_t^{target} as defined in (3). Unfortunately, it is not directly usable as the integral in (13) is intractable.

Hence, it is imperative to find an alternative. Leveraging the fact that the conditional vector field $u_t^{\text{target}}(x \mid z)$ is tractable, a conditional flow matching loss is defined replacing (3):

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\square}\left[\| u_t^{\theta}(x) - u_t^{\text{target}}(x \mid z) \|^2 \right] \tag{15}$$

where $\square \hookrightarrow t \sim \text{Unif}[0, 1]$, $z \sim p_{\text{data}}$, $x \sim p_t(\cdot \mid z)$.

While the goal is to learn the marginal vector field, regressing against its conditional counterpart results in a closed form expression. This choice implicitly induces regression toward the marginal vector field. This is made possible because $\mathcal{L}_{\text{FM}}(\theta) = \mathcal{L}_{\text{CFM}}(\theta) + C$ and C is independent of θ. Hence their gradients coincide [34].

$$\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta). \tag{16}$$

Using (12) in (15)

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\square}\left[\left\| u_t^{\theta}(x) - \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t} \alpha_t \right) z - \frac{\dot{\beta}_t}{\beta_t} x \right\|^2 \right] \tag{17}$$
$$\overset{(i)}{=} \mathbb{E}_{\square}\left[\left\| u_t^{\theta}(\alpha_t z + \beta_t \epsilon) - (\dot{\alpha}_t z + \dot{\beta}_t \epsilon) \right\|^2 \right] \tag{18}$$

For the special case of Gaussian conditional optimal transport paths where $\alpha = t$, $\beta = 1 - t$ resulting in $\dot{\alpha}_t = 1$, $\dot{\beta}_t = -1$, such that

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\square}\left[\| u_t^{\theta}(tz + (1-t)\epsilon) - (z - \epsilon) \|^2 \right] \tag{19}$$

The loss can be made more concrete for a conditional generation scenario. Since the action trajectories are conditioned on observation sequence $o \in \mathcal{O}^{\mathcal{T}} \in \mathbb{R}^{H_o \times O_d}$, the neural network is defined as $u_t^{\theta} : \mathcal{A}^{\mathcal{T}} \times \mathcal{O}^{\mathcal{T}} \times [0, 1] \to \mathcal{A}^{\mathcal{T}}$. Since time is also an input to the neural network, (19) can be written as

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\square}\left[\| u_t^{\theta}(x \mid o, t) - (z - \epsilon) \|^2 \right] \tag{20}$$

In order to inject the model with additional guidance i.e. text prompts in this case, the neural network must mimic a guided vector field $u_t^{\theta}(\cdot \mid y)$. Here, $y \in \mathcal{Y}$ is a guidance vector that consists of text embeddings drawn from a pretrained model which drives the action trajectories to the original distribution. The neural network can now be defined as

$$u_t^{\theta} : \mathcal{A}^{\mathcal{T}} \times \mathcal{O}^{\mathcal{T}} \times \mathcal{Y} \times [0, 1] \to \mathcal{A}^{\mathcal{T}} \;\; (x, o, t, y) \hookrightarrow (x \mid o, t, y) \tag{21}$$

The guided conditional flow matching objective can be written as

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\square}\left[\| u_t^{\theta}(x \mid o, t, y) - (z - \epsilon) \|^2 \right] \tag{22}$$

In order to implement the classifier-free guidance (CFG) technique, the model has to be trained in the absence as well as presence of guidance. This can be achieved while randomly dropping the guidance condition y to ϕ (null or zero) with a certain probability (p) during training. The entire training procedure is summarized in Algorithm 1. The scaled vector field can be written as the linear combination of the guided and unguided vector fields [34]. During inference, this

neural network must be leveraged for the generation of action sequences, as summarized in Algorithm 2.

$$\hat{u}_{scaled} = (1 - w)\, u_t^\theta(x \mid o, t, y = \phi) + w\, u_t^\theta(x \mid o, t, y) \tag{23}$$

where $w > 1$ is the guidance scale.

A. Neural Network Architecture

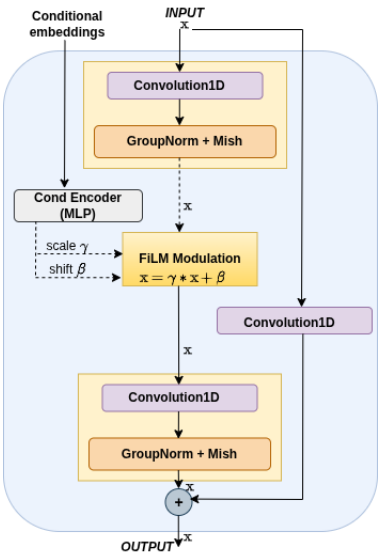

Fig. 6: Structure of the Conditional Residual Block (CondResBlock1D).

The choice of neural network used to parameterize the velocity field $u_t^\theta(x \mid o, t, y)$ is the U-Net architecture [22]. The neural network has the following inputs: the action sequence $x \in \mathcal{A}^\mathcal{T}$, the sampling time $t \in [0, 1]$, the observation sequence $o \in \mathcal{O}^\mathcal{T}$, and the guiding vector $y \in \mathcal{Y}$. The output is a flow field that has the same dimensions as the input action sequence. Although the classic U-Net architecture is commonly used for images, its effectiveness can be leveraged to model action trajectories as well through minor modifications. In contrast to images in which 2-D spatial convolutions are used, it uses 1-D temporal convolutions for action trajectories. It is also imperative to model the input images and text instructions via frozen pre-trained models in order to get the vector embeddings.

The entire architecture for TC-FMP is illustrated in Fig. 5. The architecture consists of a series of encoders and corresponding decoders with a latent processing block (bottleneck or mid-coders) in between. The encoders are connected to the decoders via residual connections. Each encoder block consists of two Conditional Residual Blocks (CondResBlock1D) followed by a downsampling layer. The output of the first block is used as a skip connection which is fed to the decoders. The skip connections preserve fine temporal detail across scales. The downsampling layer is a 1-D convolution layer to reduce the temporal resolution with the exception of the final encoder block.

Algorithm 1: Training procedure for TC-FMP

Input: Paired (action,observation) dataset $(a, o) \sim p_{\text{data}}$, Pretrained text embeddings y, Neural Network u_t^θ

foreach *mini-batch of data* **do**
- Sample a data example $z \sim p_{data}$ from the dataset;
- Sample random time $t \sim \text{Unif}_{[0,1]}$;
- Sample noise $\epsilon \sim \mathcal{N}(0, I_d)$;
- Set $x = tz + (1 - t)\epsilon$;
- With probability p, semantic conditioning: $y \leftarrow \varnothing$;
- Compute loss:

$$\mathcal{L}(\theta) = \left\| u_t^\theta(x \mid o, t, y) - (z - \epsilon) \right\|^2$$

- Update parameters θ via gradient descent on $\mathcal{L}(\theta)$;

Each Conditional Residual block consists of two 1-D convolution layers followed by Group Normalisation and Mish activation functions. Feature wise Linear Modulation (FiLM) [19] is applied to the output of the first activation. A conditional encoder layer, which is a standard multilayer perceptron, takes the concatenated conditional embeddings and produces two parameters γ and β. The output of the first activation is multiplied by γ and shifted by β. FiLM conditioning injects diffusion timestep and global context uniformly at every level. This is followed by another 1-D convolution layer followed by group normalization. After activation, the output is added to the original input to the block passed via another 1-D convolution block (identity if the input and output channels are the same). The residual connections help to stabilize every block. The structure of the Conditional Residual Block is shown in Fig. 6.

The bottleneck layer consists of two consecutive Conditional Residual Blocks. The output is then concatenated with the skip connections of the last encoder block and fed to the decoder blocks. Each decoder block consists of two successive Conditional Residual Blocks followed by an upsampling layer which is a 1-D convolution transpose operation to increase the temporal resolution with the exception of the final decoder block. The output is then passed through final convolution blocks to restore the original input shape.

All the conditioning embeddings are concatenated prior to FiLM modulation. The sampling timestep embeddings are generated by first generating sinusoidal embeddings followed by a standard MLP. The image observation features are converted to embeddings using a vision encoder, which is ResNet-18 [29] in this case. The state observation features are concatenated directly. The text instructions are converted into embeddings using the pre-trained SentenceBERT model [28].

IV. Results

The training procedure is summarized in Algorithm 1. The neural network has approximately 8M parameters and is trained using the AdamW optimizer with stochastic dropout $(p = 0.1)$. During training, 10 evenly spaced evaluations are conducted, each consisting of 10 rollouts. The guidance scale (w) is empirically selected as 3.0. The evaluation procedure is summarized in Algorithm 2. The performance is measured as

Algorithm 2: Evaluation procedure for all policies

Input: Environment $\mathcal{E}$, Trained Neural Network u^θ, normalization statistics, observation horizon H_o, action horizon H_a, guidance scale w

Output: Episode reward and rollout visualization

Initialize environment $\mathcal{E}$ with fixed seed;
Reset environment and obtain initial observation o_0;
Initialize observation deque $\mathcal{D} \leftarrow [o_0]^{H_o}$;
Initialize reward list $\mathcal{R} \leftarrow \emptyset$;
Set step counter $n \leftarrow 0$;
while *episode not terminated and* $n < N_{\max}$ **do**
  Stack observations $O \leftarrow \text{stack}(\mathcal{D})$;
  Normalize observations $\tilde{O} \leftarrow \text{Normalize}(O)$;
  Flatten observation conditioning vector $\tilde{o} \leftarrow \text{vec}(\tilde{O})$;
  Initialize a linearly spaced timesteps vector $\mathcal{T} \in [0, 1]$;
  Initialize noisy action sequence $A^{(0)} \sim \mathcal{N}(0, I_d)$;
  foreach *timestep* $t \in \mathcal{T}$ **do**
    Predict guided velocity field $\hat{u}_{guided} \leftarrow u^\theta(A^{(t)} \mid \tilde{o}, t, y)$;
    Predict unguided velocity field $\hat{u}_{unguided} \leftarrow u^\theta(A^{(t)} \mid \tilde{o}, t, y = \phi)$;
    Compute scaled velocity field $\hat{u}_{scaled} = (1 - w) * \hat{u}_{unguided} + w * \hat{u}_{guided}$;
    Perform 2^{nd} order Heun Integration
    $$A^{(t+1)} \leftarrow A_t + \frac{h}{2}\left(\hat{u}_{\text{net}}(A_t) + \hat{u}_{\text{net}}(A_t + h\,\hat{u}_{\text{net}}(A_t))\right)$$
  Unnormalize predicted action sequence $A \leftarrow \text{Unnormalize}(A^{(1)})$;
  Extract executable actions upto action horizon $a_{1:H_a} \leftarrow A[H_o : H_o + H_a]$;
  for $i = 1$ **to** H_a **do**
    Execute action a_i in environment $\mathcal{E}$;
    Observe next state o, reward r, and termination flag;
    Append o to deque $\mathcal{D}$;
    Append reward r to $\mathcal{R}$;
    Increment step counter n;
    if *episode terminated* **then**
      break
return $\max(\mathcal{R}), \text{mean}(\mathcal{R})$ and rollout visualization;

the average return across the five best checkpoints, the best return across all checkpoints, and the average rollout time across all evaluation episodes rounded off to next integer. The proposed Text-Conditioned Flow Matching Policy (TC-FMP) and its performance is measured against baselines which are Diffusion policy (DP) [2], Flow Matching Policy (FMP) [7] and Streaming Flow Policy (SFP) [10]. Each environment is prompted with four different task instructions. This ablation is designed to quantify the effects of natural language since the instructions range from being more meaningful to more abstract. Note that not providing any prompts would result in an unguided vector field, which would provide the same results as the Flow Matching Policy from [7].

The results for the planar PushT environment are reported in Table I. It can be seen that the proposed policy (TC-FMP) yields a higher mean as well maximum return compared to the others. However the policy demands a higher rollout time. The prompts to the policy in decreasing order of abstraction are as follows: $(P1)$ *Push the block to goal zone.* $(P2)$ *Push the gray block to goal zone.* $(P3)$ *Push the gray block to goal zone in green.* $(P4)$ *Push the T-shaped gray block to goal zone in green.*

#	Method	Action imitation Avg/Max Return ↑	Rollout Time ↓
1	Diffusion Policy [1]	91.8 / 96.0	7s
3	Flow matching policy [2]	84.5 / 86.6	4s
4	Streaming Flow Policy [3]	89.5 / 91.9	**1s**
5	TC-FMP $(P1)$	89.2 / 91.3	10s
5	TC-FMP $(P2)$	97.2 / 89.2	10s
5	TC-FMP $(P3)$	98.7 / 93.4	10s
6	**TC-FMP** $(P4)$	**95.9 / 100.0**	10s

TABLE I: Imitation learning accuracy on the PushT environment. Green indicates proposed method, red indicates baselines, and blue indicates ablations.

Table II reports the performance of the challenging bimanual insertion task under low and high noise conditions. Noise is injected into observations and actions via random scaling and shifting, as in [31]. Across noise levels, TC-FMP achieves higher returns as language instructions become more specific, demonstrating the benefit of informative text conditioning. However, this comes at a substantial computational cost: the Streaming Flow variant incurs approximately 10x lower rollout latency, reflecting a trade-off between performance and inference speed. The increased latency arises from second-order numerical integration and classifier-free guidance, which together require inference over four neural networks. The prompts to the policy in decreasing order of abstraction are as follows. $(P1)$ *Insert the dynamic adapter into the stationary adapter.* $(P2)$ *Insert the white dynamic adapter into the black stationary adapter.* $(P3)$ *Insert the white dynamic adapter with holes in the black stationary adapter with pegs.* $(P4)$ *Insert the white rectangular dynamic adapter with four holes at corners into the X-shaped black stationary adapter with pegs at corners.*

V. Conclusion

This paper proposed the Text-Conditioned Flow Matching Policy (TC-FMP), a continuous-time generative imitation learning framework. This was applied for single tasks in robotic manipulation and used natural language as an additional guidance signal. Using a combination of flow matching and classifier-free guidance (CFG), TC-FMP steers action trajectories towards higher quality solutions without compromising on simplicity and efficiency due to FiLM based conditioning. Across both the planar PushT benchmark and the contact-rich, high dimensional bimanual insertion task, the results show that more informative language prompts improve returns compared to prior diffusion and flow-based baselines. This highlights the fact that even single task settings can benefit from language input. The increased returns come at a cost of higher rollout latency which might induce certain limitations. The empirical results provide a valuable insight into the performance-time trade-off, informing practical design and deployment decisions.

	Method	*Low Noise*		*High Noise*	
		Action imitation Avg/Max Return ↑	**Rollout Time ↓**	**Action imitation Avg/Max Return ↑**	**Rollout Time ↓**
1	Diffusion Policy [2]	79.6 / 82.5	80s	71.8 / 73.2	95s
3	Flow matching policy [7]	72.1 / 75.3	55s	67.8 / 68.3	67s
4	Streaming Flow Policy [10]	81.0 / 83.6	**11s**	74.8 / 77.7	**15s**
5	TC-FMP $(P1)$	80.6 / 82.4	130s	75.6 / 77.2	157s
5	TC-FMP $(P2)$	81.7 / 83.3	122s	76.5 / 77.8	147s
5	TC-FMP $(P3)$	84.5 / 85.6	115s	77.4 / 80.8	137s
6	**TC-FMP** $(P4)$	**86.6 / 88.1**	110s	**78.0 / 81.3**	129s

TABLE II: Imitation learning accuracy on the bimanual insertion environment. Green indicates proposed method, red indicates baselines, and blue indicates ablations.

The scope for future work includes reducing inference time using distillation methods, time-decayed CFG, or fewer step solvers without sacrificing guidance methods, improving robustness to prompt ambiguity and distribution shift, and extending the approach to multi task and cross-embodiment settings while preserving the simplicity of text conditioning.

REFERENCES

[1] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

[2] Chi, Cheng, et al. "Diffusion policy: Visuomotor policy learning via action diffusion." The International Journal of Robotics Research 44.10-11 (2025): 1684-1704.

[3] Team, Octo Model, et al. "Octo: An open-source generalist robot policy." arXiv preprint arXiv:2405.12213 (2024).

[4] Nichol, Alexander Quinn, and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models." International conference on machine learning. PMLR, 2021.

[5] Lipman, Yaron, et al. "Flow matching for generative modeling." arXiv preprint arXiv:2210.02747 (2022).

[6] Ye, Sean, and Matthew C. Gombolay. "Efficient trajectory forecasting and generation with conditional flow matching." 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2024.

[7] Zhang, Fan, and Michael Gienger. "Affordance-based robot manipulation with flow matching." arXiv preprint arXiv:2409.01083 (2024).

[8] Black, Kevin, et al. "π_0: A Vision-Language-Action Flow Model for General Robot Control." arXiv preprint arXiv:2410.24164 (2024).

[9] Frans, Kevin, et al. "One Step Diffusion via Shortcut Models." The Thirteenth International Conference on Learning Representations.

[10] Jiang, Sunshine, et al. "Streaming Flow Policy: Simplifying diffusion / flow-matching policies by treating action trajectories as flow trajectories." arXiv preprint arXiv:2505.21851 (2025).

[11] Ha, Huy, Pete Florence, and Shuran Song. "Scaling up and distilling down: Language-guided robot skill acquisition." Conference on Robot Learning. PMLR, 2023.

[12] Xu, Mengda, et al. "Flow as the Cross-domain Manipulation Interface." Conference on Robot Learning. PMLR, 2025.

[13] Li, Hang, et al. "Language-guided object-centric diffusion policy for generalizable and collision-aware manipulation." 2025 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2025.

[14] Liang, Zhixuan, et al. "Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

[15] Reuss, Moritz, et al. "Goal-Conditioned Imitation Learning using Score-based Diffusion Policies."

[16] Xian, Zhou, et al. "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation." 7th Annual Conference on Robot Learning. 2023.

[17] Chen, Lawrence Yunliang, et al. "RoVi-Aug: Robot and Viewpoint Augmentation for Cross-Embodiment Robot Learning." Conference on Robot Learning. PMLR, 2025.

[18] Ke, Tsung-Wei, Nikolaos Gkanatsios, and Katerina Fragkiadaki. "3D Diffuser Actor: Policy Diffusion with 3D Scene Representations." Conference on Robot Learning. PMLR, 2025.

[19] Perez, Ethan, et al. "FiLM: Visual reasoning with a general conditioning layer." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[20] Pearce, Tim, et al. "Imitating Human Behaviour with Diffusion Models." The Eleventh International Conference on Learning Representations.

[21] Wang, Zidan, et al. "Cold diffusion on the replay buffer: Learning to plan from known good states." Conference on Robot Learning. PMLR, 2023.

[22] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Cham: Springer international publishing, 2015.

[23] Ho, Jonathan, and Tim Salimans. "Classifier-Free Diffusion Guidance." NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications.

[24] Power, Thomas, et al. "Sampling constrained trajectories using composable diffusion models." IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning. 2023.

[25] Reuss, Moritz, et al. "Goal-conditioned imitation learning using score-based diffusion policies." arXiv preprint arXiv:2304.02532 (2023).

[26] Rouxel, Quentin, et al. "Flow matching imitation learning for multi-support manipulation. In 2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)." (2024): 528-535.

[27] Schaal, Stefan, et al. "Learning movement primitives." Robotics Research. The Eleventh International Symposium: With 303 Figures. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[28] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

[29] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[30] Florence, Pete, et al. "Implicit behavioral cloning." Conference on robot learning. PMLR, 2022.

[31] Drolet, Michael, et al. "A comparison of imitation learning algorithms for bimanual manipulation." IEEE Robotics and Automation Letters (2024).

[32] Zhou, Yi, et al. "On the continuity of rotation representations in neural networks." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[33] Holderrieth, Peter, and Ezra Erives. "An Introduction to Flow Matching and Diffusion Models." arXiv preprint arXiv:2506.02070 (2025).

[34] Lipman, Yaron, et al. "Flow matching guide and code." arXiv preprint arXiv:2412.06264 (2024).

[35] Sridhar, Ajay, et al. "Nomad: Goal masked diffusion policies for navigation and exploration." 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024.