Supplement Contents

A Additional Details and Results for FabricFlowNet

A.1 FabricFlowNet Implementation Details

Data Collection. We collect data in SoftGym by taking random pick and place actions on the cloth. The random actions are biased to pick corners of the cloth mask (detected using Harris corner detection [1]) 45% of the time, and "true" corners of the square cloth 45% of the time. If the true corners are occluded then Harris corners are used instead. For the remaining 10%, the pick actions are uniformly sampled over the visible cloth mask. After the pickers grasp the cloth, they lift to a fixed height of 7.5 cm.

We constrain the place points of the action so that both place points are offset in the same direction and distance from their respective pick points. The direction is orthogonal to the segment connecting the two pick points, and points towards the center of the image, so the cloth does not move out of the frame (similar to Lee *et al.* [5]). The distance between the pick point and the place point along this direction is uniformly sampled between $[25, 150]$ px. The distance is truncated if it exceeds a margin of 20 px from the image edge, again to prevent moving the cloth out of the frame. While these heuristics may seem to overly constrain the data we collect, we observe that our data still contains highly diverse cloth configurations, as shown in Fig. S1.

For each sample, we save the initial depth observation image, the dual-arm pick and place pixel locations of the action, the next depth observation resulting from the executed action, and the cloth

particle positions of both observations (See Fig. S1). The camera for capturing depth observations is fixed at 65 cm above the support surface. We mask the depth observations to only include the cloth by setting all background pixels to zero. The dataset for training both the flow and pick networks consists of 20k samples from 4k episodes, where each episode consists of five dual-arm pick and place actions.

Flow Network Training. We use FlowNet [3] as our flow network architecture. The input to FlowNet is the initial and next depth image from a sample in our dataset, stacked channel-wise. The ground truth flow for supervising FlowNet comes from the cloth particles used by the simulator to model the cloth's dynamics: we collect the cloth particle positions for each observation in our dataset and correspond them across observations to get flow vectors (See Fig. S1). The ground truth flow is sparse because the cloth particles are sparse, so we train FlowNet using a masked loss that only includes pixels with corresponding ground truth flow. Similar to Lee *et al.* [5], we apply spatial augmentation of uniform random translation (up to 5 px) and rotation (up to 5 degrees) to augment the training data. We train the network using the dataset of 20k random actions described above. We use the Adam [4] optimizer, learning rate 1e-4, weight decay 1e-4, and batch size 8.

PickNet Network Training. PickNet1 and PickNet2 are fully-convolutional network architectures based on Lee *et al.* [5], with 4 convolutional layers in the encoder, each with 32 filters of size 5. The first three layers of the encoder have stride 2 and the last one has stride 1. The decoder consists of 2 interleaved convolutional layers and bilinear upsampling layers.

The input to the PickNet1 is a 200 $\times$ 200 flow image. PickNet2 receives the first pick point location (the argmax of the Picknet1 output, as described in the main text) as an additional input, represented as a 2D Gaussian $\mathcal{N}(p_1, \sigma)$ (where $\sigma = 5$). Similar to Nair *et al.* [6], the output of both networks is a 20 x 20 spatial grid. If the pick points predicted by PickNet are not on the cloth mask, we project them to the closest pixel on the mask using an inverse distance transform. In practice, we find that the predictions are usually either on the cloth mask or very close to the mask. To train PickNet1 and PickNet2, we use the same dataset of 20k random actions described above. We use the Adam [4] optimizer, learning rate 1e-4, and batch size 10.

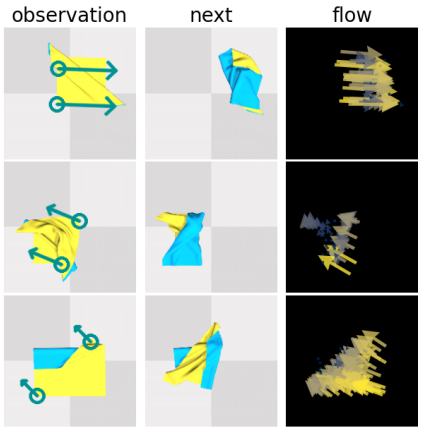


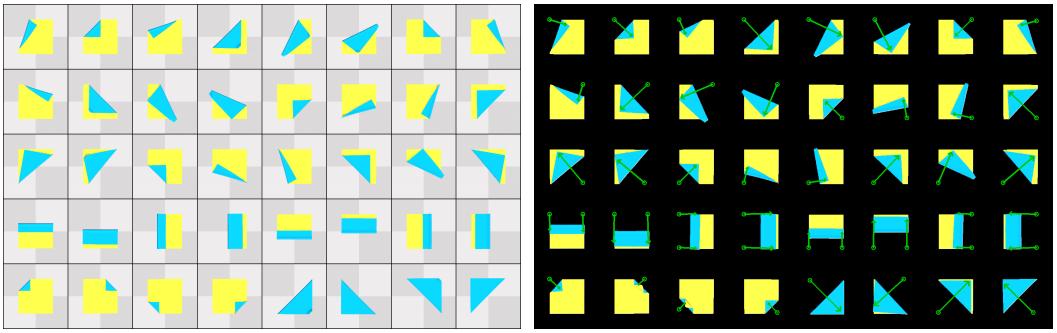
Figure S1: Training data for FFN.

A.2 Additional Simulation Results for FabricFlowNet

Fig. S2b and Fig. S2f show the cloth configurations achieved by FabricFlowNet for each of the one-step goals (Fig. S2a) and multi-step goals (Fig. S2e). Fig. S1 provides examples of the data used to train FFN. Our policy is deterministic and the simulation is near-deterministic, so we only need 1 trial for our simulation experiments (unlike our real world experiments which use 3 trials).

A.3 Additional Real World Details and Results for FabricFlowNet

Cloth Masking. In simulation, we can obtain a perfect cloth mask. In the real world, we first obtain a background mask of the table using color-based HSV thresholding, which we can determine before the cloth is placed on the table. We then use the inverse of this background mask to obtain a mask of



(a) One-step goals

(b) One-step FFN performance

(c) One-step Fabric-VSF performance

(d) One-step Lee et al. performance

(e) Multi-step goals

(f) Multi-step FFN performance

(g) Multi-step Fabric-VSF performance

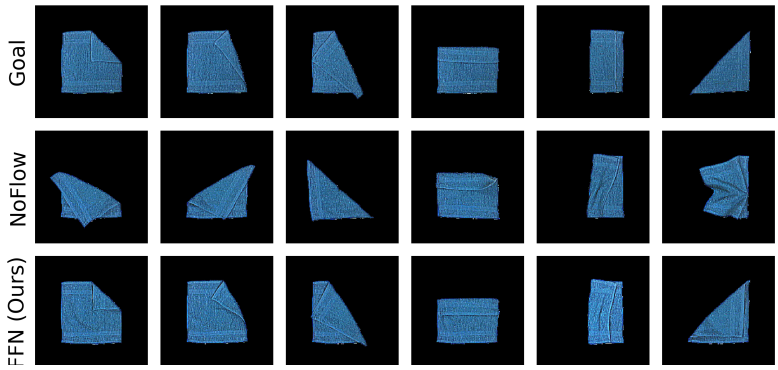
(h) Multi-step Lee et al. performance

Figure S2: Goal configurations, achieved configurations, and training data in simulation. Arrows indicate the executed action. Fabric-VSF uses a lower camera height than FFN (45 cm vs. 65 cm), thus the cloth looks slightly larger.

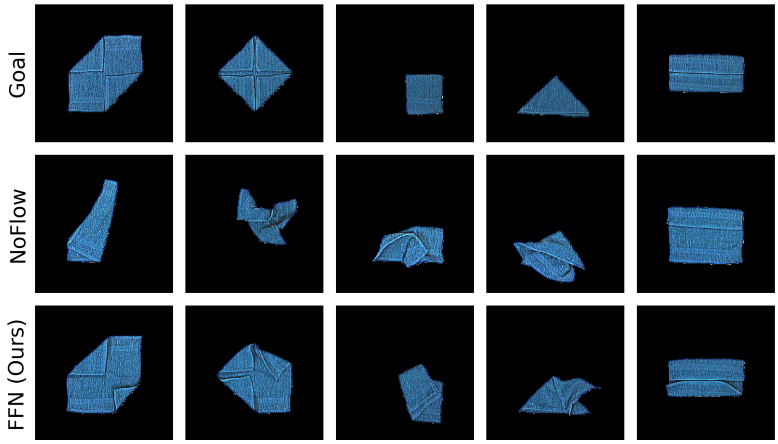
the cloth. Note that while we use background color of the table for cloth masking, the network itself only takes depth input, allowing the network to be robust to colors and patterns on the cloth itself.

Results on Real Cloth Folding. Table S1 provides mean IOU (mIOU) performance for NoFlow and FFN on real cloth goals. The NoFlow ablation performs considerably worse compared to FFN on real cloth folding. Qualitative results and the complete set of real square cloth goals are in Fig. S3; the complete set of real rectangle and T-shirt goals are in the main text. We found that for FFN, using FlowNet weights from epochs at the start of convergence transferred better to the real world than using weights from epochs long after convergence.

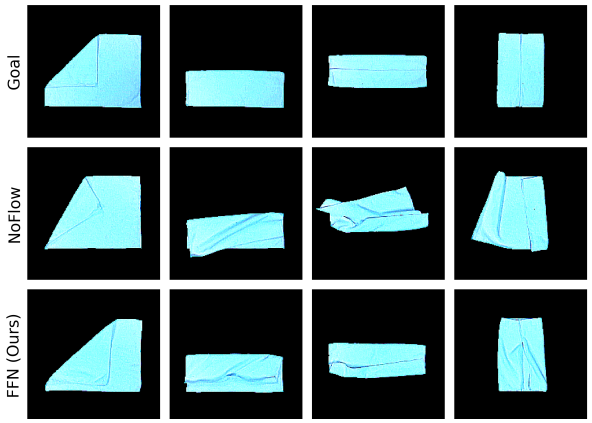
Failure Cases. This work focused on high level actions with fixed primitives for picking and placing that may not be ideal for all cloth types, sizes, or folds. Causes of failures include the grasped portion of the cloth "flopping back" against the folding direction, undoing small folding actions



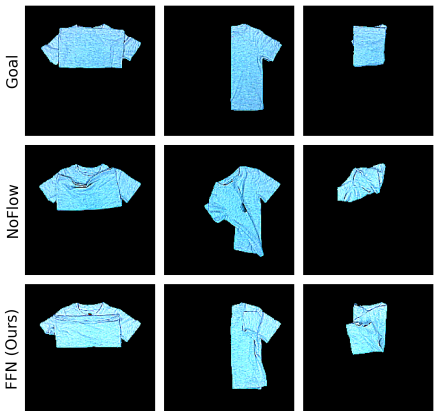
Single-step Square Towel



Multi-step Square Towel



Rectangular Cloth



T-shirt

Figure S3: Qualitative performance of FFN and NoFlow on real cloth. The trial corresponding to the best achieved IOU is shown for each example. For multi-step goals, only the final goal is shown. FFN only takes depth images as input, allowing it to easily transfer to cloths of different colors. Contrast and brightness have been adjusted to enhance visibility.

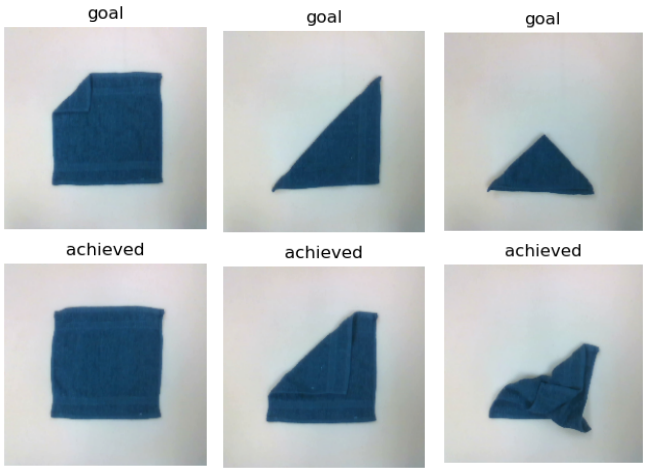
or causing unwanted secondary folds (Fig. S4a). Potential future work is to learn better pick and place primitives. Another source of failure was over- or under-estimating the fold distance due to slight inaccuracies in the flow prediction (Fig. S4b). We also see some failures during multi-step folding; since we provide sub-goals in sequence and allow only one action per sub-goal, the

Table S1: mIOU for Folding Square Towel, Rectangular Cloth, and T-shirt

Method	1-Step Sq. ↑ ($n = 6$)	Multi-Step Sq. ↑ ($n = 5$)	All Sq. ↑ ($n = 11$)	Rect. ↑ ($n = 3$)	T-shirt ↑ ($n = 3$)
NoFlow	0.59 ± 0.04	0.45 ± 0.01	0.53 ± 0.02	0.65 ± 0.07	0.61 ± 0.06
FFN (Ours)	$\mathbf{0.89 \pm 0.01}$	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.80 \pm 0.03}$	$\mathbf{0.81 \pm 0.04}$	$\mathbf{0.82 \pm 0.02}$

Average of 3 rollouts. Higher mIOU scores are better; the max achievable score is 1.0.

discrepancy between the starting image of the demonstration and the observed image can result in poor predictions (Fig. S4c). Allowing the policy to take multiple actions to achieve a sub-goal before proceeding may improve performance. For example, the flow can be recalculated after each action to determine if the observation is sufficiently close to the desired sub-goal configuration before proceeding to the next sub-goal.



(a) Flopping back (b) Undershooting (c) Poor prediction

Figure S4: Examples of failure cases

B Additional Details and Results for Fabric-VSF [2]

B.1 Fabric-VSF [2] Implementation Details

The original Fabric-VSF [2] paper uses single arm actions and a top-down close camera view such that the cloth covers the whole image. To match the camera view, we set the camera height to be 45 cm above the table in our case. The training dataset consists of 7115 trajectories, each with 15 random pick-and-place actions, totaling 106725 data points. Note that this dataset is 5x larger than the 20k samples we train FFN on. During training, Fabric-VSF takes as input 3 context frames and predicts the next 7 target frames.

We trained 8 variants of Fabric-VSF. Each variant differs in the following aspects: 1) whether it uses single arm or dual arms; 2) during data collection, whether the pick-and-place actions are randomly sampled, or use the corner biasing sampling strategy as described in Sec. A.1, and 3) whether it uses the original small action size ("Small Action", bounded to half of the cloth width) or a larger action size ("Large Action", bounded to the diagonal length of the cloth). Other than these three changes, we set all other parameters to be the same as in the original paper. Therefore, the variant with single arm actions, no corner biasing during data collection, and small action size is exactly how Fabric-VSF is trained in the original paper.

After the training, we plan with cross-entropy method (CEM) to find actions for achieving a given goal image. We use the exact same CEM parameters as in the original paper, *i.e.,* we run CEM for 10 iterations, each with a population size of 2000 and elite size of 400.

B.2 Additional Fabric-VSF [2] Results

The results for the Fabric-VSF variants are summarized in Table S2. We note that the variant using single arm actions, corner biasing for data collection, and large action size performs the best out of all variants. This variant outperforms FFN on overall error and one-step error, but performs slightly worse than FFN on multi-step error (See Fig. S2c and Fig. S2g for qualitative results). However, we note that Fabric-VSF was trained on 5x more data than FFN. Additionally, Fabric-VSF takes much longer to run at inference time, requiring $\sim$7 minutes of CEM iterations to compute a single action compared to $\sim$0.007 seconds for a forward pass through FFN. 7 minutes of CEM planning time is impractical for real-world folding. We also demonstrate in the following section that FFN generalizes to other cloth shapes better than Fabric-VSF.

Analyzing the performance between different Fabric-VSF variants, for single-arm actions, using large actions instead of small actions always leads to better performance. However, this is not true for the dual arm variants. Interestingly, we find that using dual arms tends to result in worse performance compared with using a single arm. The reason for this could be that during CEM planning, dual-arm variants double the action dimension, which increases complexity for CEM and makes it difficult to find optimal actions.

Table S2: Mean Particle Distance Error (mm) and Inference Time (sec) for Fabric-VSF Variants

Baseline	1-Step (n=40)	Multi-Step (n=6)	All (n=46)	Inf. Time
1-Arm, No CB, Sm. Action	12.92 ± 13.00	46.05 ± 48.07	17.24 ± 23.93	$\sim$420s
1-Arm, No CB, Lg. Action	10.13 ± 07.33	33.06 ± 12.46	13.12 ± 11.25	$\sim$420s
1-Arm, CB, Sm. Action	14.09 ± 11.36	38.68 ± 27.72	17.30 ± 16.76	$\sim$420s
1-Arm, CB, Lg. Action	6.30 ± 06.55	$\mathbf{21.33 \pm 11.20}$	8.27 ± 08.90	$\sim$420s
2-Arm, No CB, Sm. Action	24.60 ± 14.69	50.26 ± 27.54	27.94 ± 19.00	$\sim$420s
2-Arm, No CB, Lg. Action	10.98 ± 05.80	40.92 ± 18.06	14.89 ± 13.17	$\sim$420s
2-Arm, CB, Sm. Action	16.21 ± 13.81	36.42 ± 26.51	18.84 ± 17.43	$\sim$420s
2-Arm, CB, Lg. Action	15.58 ± 10.88	54.06 ± 26.68	20.60 ± 19.07	$\sim$420s
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$	$\sim$**0.007s**

CB: Corner Bias Sm. Action: Small Action Lg. Action: Large Action

C Additional Details and Results for Lee *et al.* [5]

C.1 Lee *et al.* [5] Implementation Details

Lee *et al.* [5] learns a fabric folding policy for a discrete action space using a fully convolutional state-action value function, or Q-network. Observation and goal images are stacked channel-wise, then duplicated and transformed to form a batch of m image rotations and n scales to represent different pick and place directions and action lengths. The whole batch is input to the Q-network to compute the Q-value of executing an action for each rotation and scale at every point on the image. The action corresponding to the max Q-value from the outputs is executed. The discrete action space of m rotations and n action lengths for Lee *et al.* [5] enables efficient policy learning, but greatly limits the actions of the learned policy compared to FFN.

We extend Lee *et al.* [5] from a single-arm approach to a dual-arm one. To represent two pickers instead of one, we input two pairs of observation and goal images to the Q-network. When rotating and scaling the images to represent different actions, the images are constrained to have the same rotation, but are allowed to be scaled differently. In other words, the dual-arm actions are constrained to execute pick and place actions in the same direction, but can have different pick and place lengths. The Q-network outputs a pair (one for each arm) of Q-value heatmaps for every action in the discrete action space (*i.e.,* every rotation and scale). The max Q-value in each of the two heatmaps is averaged, and the heatmap pair with the highest averaged Q-value is selected from the set of all discrete rotations and scales. The picker action corresponding to the argmax of each heatmap is executed.

We train each Lee *et al.* variant below using hyperparameters similar to the original paper [5], training for 25k steps with learning rate 1e-4, batch size 10, and evaluating performance on test goals every 500 steps to find the best performing step.

C.2 Additional Lee *et al.* [5] Results

We trained variants of Lee *et al.* to compare single-arm vs. dual-arm performance, depth input vs. RGB input, collecting data with corner bias similar to FFN vs. without bias, and using the original close-up image of the cloth ("Low Cam") vs. images from further away ("High Cam"). All variants were trained with 20k training examples. We also provide results for two variants of FFN trained on the same amount of data, one where actions are sampled from the discrete action space (*i.e.,* discretized action angles and lengths) in Lee *et al.* [5] ("Discrete Actions"), and the other where actions are sampled using our continuous action space described in Sec. A.1 ("Cont. Actions"). Lee *et al.* [5] is an inherently discrete approach and cannot be trained to output continuous actions, nor can it be trained on data with actions outside of its discrete action space.

Table S3 shows that the performance of all Lee *et al.* variants is poor compared to FFN, particularly on 1-step goals (see Appendix Fig. S2d and Appendix Fig. S2h for qualitative results). FFN outperforms Lee *et al.* when trained on either the discrete action dataset or the continuous one. Training FFN on continuous actions results in better performance for 1-step goals, but the discrete action dataset also performs fairly well. These results indicate that the improved performance of FFN vs. Lee *et al.* cannot be solely explained by training on continuous vs. discrete action data, though other factors like outputting continuous actions instead of discrete ones may still play significant role in FFN's improved performance.

Table S3: Mean Particle Distance Error for Lee *et al.* on 20k Training Examples

Baseline	1-Step (40)	Multi Step (6)	All (46)
Lee *et al.*, 1-Arm, D, No CB, LC	18.94 ± 16.43	24.18 ± 17.75	19.62 ± 16.49
Lee *et al.*, 1-Arm, D, No CB, HC	16.18 ± 08.38	26.20 ± 16.31	17.49 ± 10.10
Lee *et al.*, 1-Arm, D, CB, LC	20.99 ± 18.88	34.61 ± 31.35	22.77 ± 20.97
Lee *et al.*, 1-Arm, D, CB, HC	19.70 ± 09.37	38.91 ± 24.05	22.20 ± 13.53
Lee *et al.*, 1-Arm, RGB, No CB, LC	49.29 ± 18.10	52.03 ± 33.62	49.65 ± 20.26
Lee *et al.*, 1-Arm, RGB, No CB, HC	47.12 ± 21.04	64.48 ± 29.85	49.38 ± 22.75
Lee *et al.*, 1-Arm, RGB, CB, LC	33.89 ± 19.01	58.90 ± 43.34	37.15 ± 24.38
Lee *et al.*, 1-Arm, RGB, CB, HC	39.01 ± 25.36	55.46 ± 38.38	41.15 ± 27.43
Lee *et al.*, 2-Arm, D, No CB, LC	36.62 ± 14.51	47.72 ± 21.95	38.07 ± 15.82
Lee *et al.*, 2-Arm, D, No CB, HC	40.75 ± 13.22	52.88 ± 19.03	42.33 ± 14.45
Lee *et al.*, 2-Arm, D, CB, LC	47.18 ± 18.60	57.29 ± 28.65	48.50 ± 20.07
Lee *et al.*, 2-Arm, D, CB, HC	35.98 ± 24.60	64.75 ± 51.76	39.73 ± 30.30
FFN, 2-Arm, D, CB, HC, Discrete Actions	9.57 ± 06.07	$\mathbf{10.15 \pm 07.20}$	10.17 ± 07.34
FFN, 2-Arm, D, CB, HC, Cont. (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

D: Depth CB: Corner Bias LC: Low Camera HC: High Camera Cont: Continuous Actions

Lee *et al.* with and without Subgoals. FFN uses subgoals at inference time in order to fully specify the task; many cloth folding goals have final goal configurations in which large portions of the cloth are self-occluded. Subgoals are required to ensure the task is completed correctly and that the cloth is correctly folded. Lee *et al.* [5] demonstrated cloth folding without subgoals at inference time by relying on a learned Q-value heatmap to select actions toward a final end goal. We compare the performance of the best Lee *et al.* variant with and without subgoals at test-time. The results of this experiment are in Table S4. While the performance on 1-step goals are similar because those tasks do not have subgoals, performance on multi-step goals is worse without subgoals.

Table S4: Mean Particle Distance Error for Lee *et al.* With and Without Subgoals

Method	1-Step (40)	Multi Step (6)	All (46)
Lee *et al.*	16.92 ± 9.28	37.74 ± 38.99	19.71 ± 20.27
Lee *et al.*, With Subgoals	$\mathbf{16.18 \pm 8.38}$	$\mathbf{26.20 \pm 16.31}$	$\mathbf{17.49 \pm 10.10}$

D Additional Details and Results for Ablations

D.1 Ablation Implementation Details

NoFlowIn The architecture for this ablation is identical to our main method, except that it takes depth images instead of flow images as input. We use a conditioned architecture with two PickNets; PickNet1 receives the observation and goal depth images as input both of size 200×200. The place point is computed by querying the flow image similar to our main method.

NoFlowPlace We predict the place points similarly to the pick points by using an additional place network. The place network architecture is identical to PickNet. The input is a flow image and the output is the place point predictions.

NoFlow This ablation is a combination of NoFlowIn and NoFlowPlace, where PickNet and PlaceNet both take observation and goal depth images as input.

NoCornerBias This ablation is the same as our main method except for the training dataset. We use a dataset that does not bias the data to pick corners (See Sec. A.1). Instead, the pick actions are always uniformly sampled over the visible cloth mask. We still constrain the folding actions for both arms to be in the same direction and distance from their respective pick points and point towards the center of the frame.

NoSplitPickNet The architecture of PickNet is modified so that we only have one PickNet for both arms instead of the conditioned architecture used in our main method. The PickNet takes as input the flow image and outputs two heatmaps corresponding to the two pick points.

NoMinLoss The loss in Eq. 1 is replaced with the following:

$$\mathcal{L}_{\text{NoMin}} = \text{BCE}(H_1, H_1^*) + \text{BCE}(H_2, H_2^*) \tag{2}$$

D.2 Additional Ablation Results

We provide ablation results in Table S5 grouped by single-step, multi-step, and all goals.

Table S5: Mean Particle Distance Error for Ablations

Ablation	One Step (n=40)	Multi Step (n=6)	All (n=46)
NoFlowIn	5.14 ± 3.62	24.63 ± 21.30	9.37 ± 12.20
NoFlowPlace	7.61 ± 5.44	30.25 ± 17.62	10.56 ± 11.15
NoFlow	8.97 ± 7.45	28.79 ± 19.33	18.02 ± 20.34
NoCornerBias	9.79 ± 5.57	$\mathbf{19.61 \pm 17.52}$	11.07 ± 8.83
NoSplitPickNet	4.87 ± 2.61	23.41 ± 18.87	7.29 ± 9.56
NoMinLoss	5.10 ± 4.04	20.81 ± 17.57	7.15 ± 9.08
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

E Additional Results on Unseen Cloth Shapes

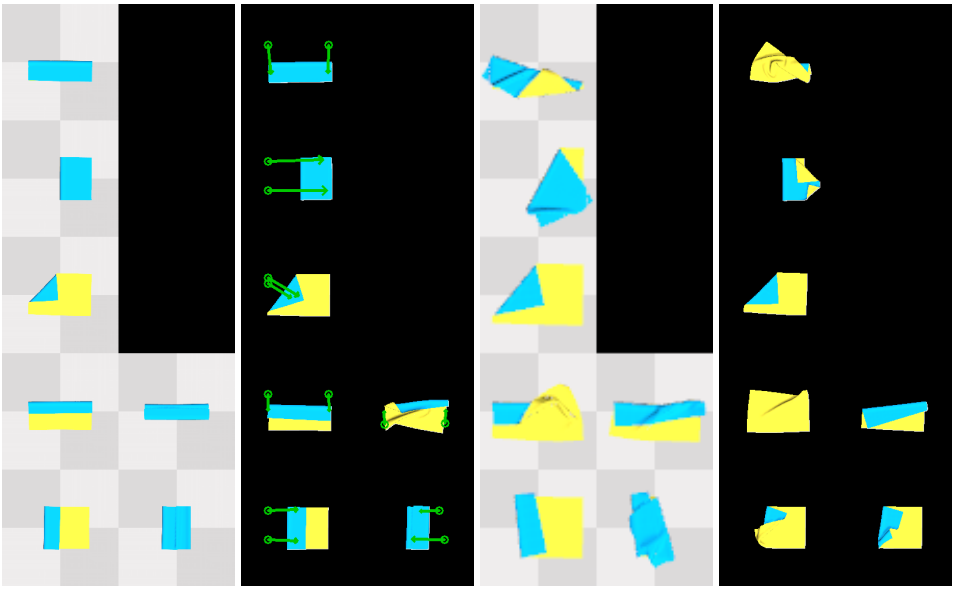
We also evaluate Fabric-VSF and Lee *et al.* on generalization to unseen cloth shapes. FFN generalizes well to new shapes, as shown in the main text (see Fig. 5 and Sec. 4.2.1). Table S6 provides quantitative results on the rectangle cloth and T-shirt for the best Fabric-VSF method and best Lee *et al.* method compared to FFN. FFN outperforms both methods by a large margin. Fabric-VSF generalizes poorly, likely because it relies on planning with a learned visual dynamics model. Lee *et al.* also does not generalize well compared to FFN. Fig. S5 provides a qualitative comparison.

F End-to-End Variants of FFN

We investigate the effect of training our FFN architecture end-to-end. First, we train the FFN architecture with pick losses as well as the flow loss; all losses are allowed to backpropagate through the

Table S6: Mean Particle Distance for Folding Unseen Cloth Shapes in Simulation

Method	Rectangle (n=6)	T-Shirt (n=3)
Lee *et al.* , 1-Arm, No Corner Bias, High Cam, 20k Actions	31.63 ± 18.04	86.65 ± 34.67
Fabric-VSF, 1-Arm, Corner Bias, Large Action	25.68 ± 11.21	45.25 ± 13.83
FFN (Ours)	$\mathbf{10.70 \pm 08.54}$	$\mathbf{20.91 \pm 11.28}$



(a) Rect. cloth goals (b) FFN achieved (c) Fabric-VSF achieved (d) Lee *et al.* achieved

Figure S5: Qualitative performance of FFN, Fabric-VSF, and Lee *et al.* on rectangular cloth.

entire combined network, including through the FlowNet layers. The results on the square towel are presented in Table S7 ("JointFFN"). This variant performs significantly worse than FFN (9.28 vs. 7.14 on all goals).

Table S7: Mean Particle Distance Error (mm) for End-to-End Variants of FFN

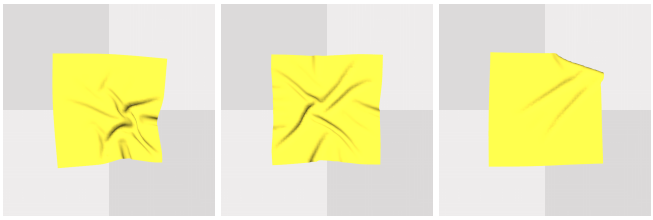
Method	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
JointFFN	07.60 ± 05.62	$\mathbf{17.53 \pm 15.56}$	09.28 ± 09.39
JointPredictPlace	12.90 ± 11.67	35.25 ± 19.22	22.88 ± 23.24
JointFFN, No Flow Loss	32.41 ± 22.61	68.17 ± 50.35	37.07 ± 30.34
JointPredictPlace, No Flow Loss	16.31 ± 22.73	50.27 ± 31.44	24.39 ± 29.77
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

We also trained another variant which consists of a FlowNet, a PickNet, and a PlaceNet, trained end-to-end ("JointPredictPlace" in Table S7). This is similar to our ablation "PredictPlace" in Table 2, which uses the same architecture but is not trained end-to-end. JointPredictPlace performs significantly worse than FFN (22.88 vs. 7.14 on all goals) and also underperforms compared to PredictPlace (10.56 on all goals). Overall, this result, as well as the one in the paragraph above, indicate that end-to-end training leads to significantly worse performance for this task. Our intuition for this is that the flow network should be trained only with the flow loss, and that backpropagating the gradients from the pick loss into the flow network adds noise and reduces its performance.

Lastly, we evaluated variants of the above two architectures with the flow loss removed, to see if we could train these architectures end-to-end with just a single loss at the end, instead of using an intermediate flow loss. The results, shown in Table S7, are worse for both variants, showing the importance of the intermediate flow loss.

G FFN Performance with Crumpled Starting Configurations

Our experiments focused on folding tasks, and we assume that a previous method was used to flatten the cloth before our method is executed. To evaluate the robustness of our method to imperfect smoothing, we evaluate the performance of FFN in simulation on slightly crumpled initial cloth configurations. We generated crumpled configurations by taking the flat cloth and executing a random pick and place action with a maximum translation of 10 pixels. The three configurations used in our experiments are shown in Fig. S6.



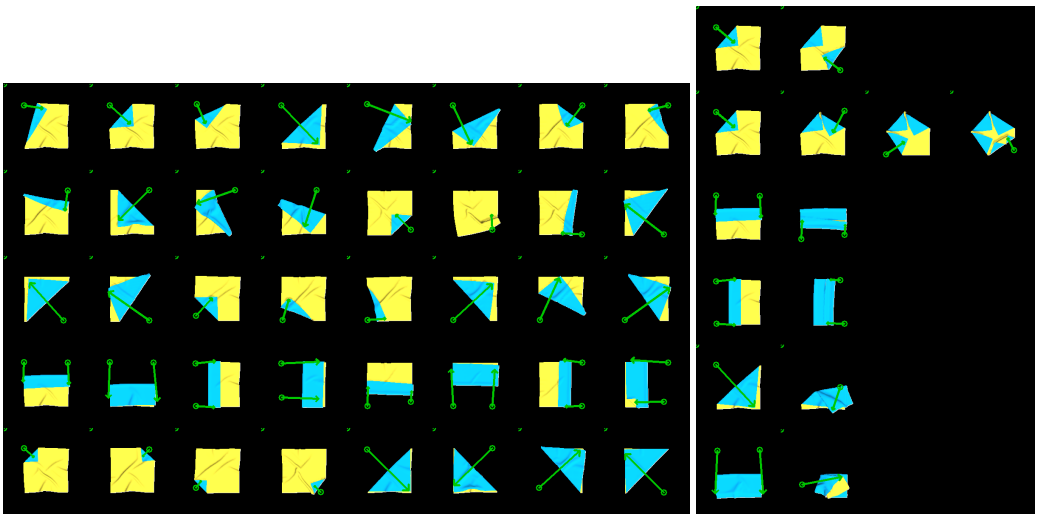
(a) Crumpled 0 (b) Crumpled 1 (c) Crumpled 2

Figure S6: Crumpled initial cloth configurations

For each crumpled configuration, we evaluated FFN on the full set of 46 evaluation goals, where the starting configuration of the cloth was set to the given crumpled configuration. The results of these evaluations are in Table S8. The particle distance error is slightly higher with the crumpled starting configurations, but the qualitative results in Fig. S7 show that FFN still produces actions that are very close to the intended goals.

Table S8: Mean Particle Distance Error (mm) for FFN with Different Start Configurations

Starting Config	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
FFN, Crumpled 0	12.40 ± 4.82	24.82 ± 24.81	14.01 ± 10.86
FFN, Crumpled 1	10.68 ± 2.89	23.54 ± 22.56	12.36 ± 9.61
FFN, Crumpled 2	10.68 ± 4.29	$\mathbf{21.05 \pm 14.70}$	12.03 ± 7.51
FFN, Flat	$\mathbf{4.46 \pm 2.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$



(a) Crumpled one-step FFN performance (b) Crumpled Multi-step FFN performance

Figure S7: Configurations achieved by FFN when starting from the "Crumpled 1" configuration for each attempt (compare with Fig. S2)

H FFN Performance with Iterative Refinement

Table S9: Mean Particle Distance Error (mm) for FFN with Iterative Refinement

Starting Config	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
FFN, No Refinement	$\mathbf{4.46 \pm 2.62}$	25.04 ± 22.88	7.14 ± 11.06
FFN, Iterative Refinement	4.54 ± 2.58	$\mathbf{20.47 \pm 19.49}$	$\mathbf{6.62 \pm 9.17}$

In our normal evaluations, each goal or subgoal is attempted only once by each method. With a single attempted action for each subgoal, FFN is able to achieve a diverse set of goals, as demonstrated in this work. However, we find that FFN can achieve even better performance when attempting goals multiple times, using the flow to compare the current observation with the goal and taking actions that move the observation closer to the goal if it has not yet been reached. We evaluate the benefit of using this "iterative refinement" procedure in simulation. FFN moves to the next subgoal when a minimum threshold for the average flow is achieved, so the flow acts as a goal recognizer. The policy is allowed a maximum of 3 iterative actions per subgoal to limit potential divergence. The results in Table S9 show that iterative refinement can improve performance, particularly on multi-step goals, where reaching the current subgoal accurately is important for achieving subsequent goals.

I FlowNet Performance

FlowNet achieves an average endpoint error (EPE) of 1.0268 on the set of simulated test goals. The test goals are not seen during training. Fig. S8 provides qualitative examples of FlowNet performance on simulated test goals.

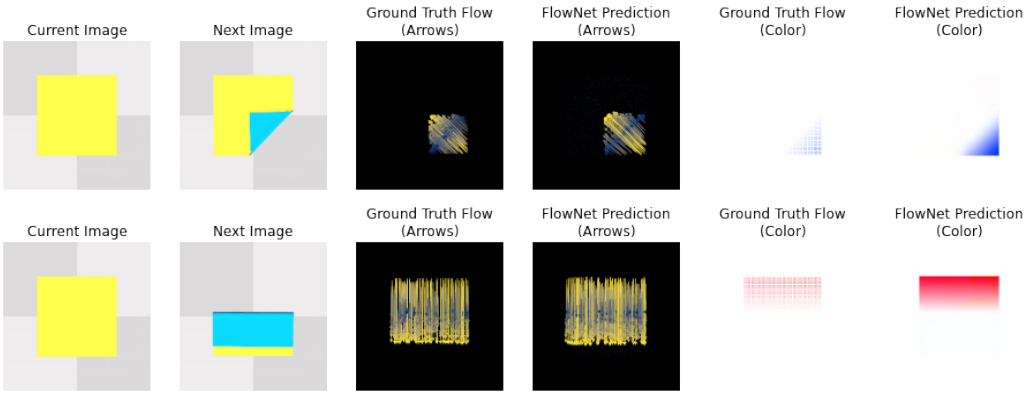


Figure S8: FlowNet Qualitative Performance. Two types of visualizations are provided: representing the flow vector as arrows, and representing the flow vector using RGB channels. FlowNet outputs a dense flow image but is trained on sparse ground truth flow. FlowNet takes only depth images as input; RGB images are shown as a visual aid only.

References

[1] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[2] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[3] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Robert Lee, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. Learning arbitrary-goal fabric folding with one hour of real robot experience. *Conference on Robot Learning*, 2020.

[6] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017.