Implementation Details

Architecture We use a ResNet-18 as our feature extractor, to be consistent with the previous work we compare against. The dataset classifier that is used in our Blender network is comprised of a permutation-invariant set encoder g (Zaheer et al., 2017) followed by a linear layer l, as explained in our main paper. We adopt a similar architecture for g to the one used in (Requeima et al., 2019) for their 'adaptation networks'. This consists of 5 convolutional blocks, each of which is comprised of a 3x3 convolution operation with 64 channels, followed by batch normalization, ReLU, and 2x2 max-pooling with stride 2. We then apply global average pooling to the output, followed by averaging over the first dimension (i.e. over the different examples of the batch), to obtain our set encoding of the given batch. This vector is then fed into l to classify the given batch into one of the M training datasets.

Training the shared and per-dataset parameters We train FLUTE via a joint phase that utilizes data from all M training datasets in order to learn a universal template Φ and M per-dataset sets of FiLM parameters $\Psi_1 \dots \Psi_M$. As detailed in the main paper, our training objective is a multi-task classification one, that requires M per-dataset classification readout heads. Following recent work (Chen et al., 2019; 2020; Dvornik et al., 2020), we treat each of those readout heads as a *cosine classifier*, i.e. a layer without a bias, parameterized only by a weight matrix, where the activations that are the inputs to the layer, as well as the rows of that matrix are L2-normalized before the matrix multiplication is performed. Following those previous works, we also utilize a learnable softmax temperature for these cosine classifiers.

We use stochastic gradient descent with a momentum of 0.9, with a cosine decay schedule with restarts for the learning rate. We also applied weight decay to the parameters of the convolutional layers and to the FiLM parameters. We tuned these parameters on the validation set, and used a starting learning rate of 0.01. The first round decays over 10000 steps from that starting learning rate to "alpha" (we use the default value of 0 for "alpha"). Then, a warm restart is performed, where the learning rate is now "m mul" times smaller than our original starting learning rate (we use the default value of 1 for "m mul"), and the decay is done over "t mul" times more steps than the previous decay round (we use the default value of 2 for "t mul"). We set the weight decay parameter for the convolutional layers to $7e-4$, and the weight decay for the FiLM parameters to 0.001, which regularizes the network's β offset parameters to 0 and the γ scaling parameters to 1 (i.e. for the γ, we apply the weight decay to $(\gamma - 1)$). Following previous work (Chen et al., 2020), during our joint training phase, we sample examples from ImageNet half the time, with the other half being devoted to examples from all training datasets uniformly.

Training the dataset classifier To train our dataset classifier, we use Adam with a cosine decay schedule for the learning rate, without restarts. The values that worked best for this (as per the validation set performance) were an initial learning rate of 0.001 that is decayed over 3000 steps. Note that this phase is significantly shorter compared to the previously-described phase that trains our feature extractor. We early-stopped the training of the dataset classifier based on the validation accuracy: specifically, this is the accuracy on the M-way dataset classification task computed on the validation set, which contains held-out classes of the M training datasets, as explained in the main paper.

Fine-tuning Ψ_{d^*} During evaluation, the fine-tuning phase within each test task also uses Adam as the optimizer, without any learning rate decay in this case. We tuned the learning rate and the number of fine-tuning steps based on episodes from the validation set. Our best variant used a learning rate of 0.005 and 6 steps. The values we considered for these were $0.0005, 0.001, 0.005$ for the learning rate and $1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 150, 20, 30$ for the number of steps.

Hypothesis testing We follow the same procedure as in (Triantafillou et al., 2020) to compute ranks for different methods that in turn determine which entries to bold in our tables. Specifically, we perform a 95% confidence interval statistical test on the difference between the mean accuracies of pairs of entries of each row. If for two entries we are not able to reject the null hypothesis that the difference between their means is 0, they will receive the same rank. For example, if model A and model B are tied for the first place according to that test, they will each receive the rank 1.5 (the average of the ranks 1 and 2). If we are able to reject that hypothesis, however, the entry with the larger mean accuracy will receive a higher rank than the other. In each row, we bold the entries that are tied for the highest rank.

The effect of the number of steps to train Ψ_{d^*}

In Figure 2 of the main paper we visualized the performance (on the support and query sets) for test episodes of held-out datasets throughout the fine-tuning of Ψ_{d^*}. For completeness, we also present in Figure 4 the same result, but for test episodes of seen datasets (weak generalization setting). We observe that the increase in accuracy is less pronounced for these seen datasets. This is expected, since we know that there already exists a set of FiLM parameters that performs well for each test task sampled from a training dataset m (namely the set Ψ_m of FiLM parameters), and assuming the dataset classifier is accurate, the Blender would almost exclusively pick that set of FiLM parameters.

Inpsecting the Blender's proposals

In Figure 3 in the main paper, we visualized the average combination co-efficients that the Blender produces for test tasks of different datasets. Since that figure only shows the average, we now take a closer look at the distribution of the Blender's proposed combination co-efficients differs *within* different test tasks of the same dataset. As a reminder, these co-efficients are computed based on the support set of each given test task, so they are not re-used across different tasks of the same dataset (in fact, the dataset identity is not known at test time).

We visualize the Blender's proposals for 600 test tasks of each dataset in Figure 6. We observe that, for each seen dataset (the first 8 sub-plots), the Blender almost exclusively picks the FiLM parameters dedicated to that specific dataset (although ImageNet and Birds sometimes pick each other to some small extent). This means that the dataset classifier is accurate across several held-out episodes of the seen datasets. For the unseen datasets, on the other hand, there is some more variability, as expected, consistent with Figure 3.

We also plot the variance of the distribution of the dataset classifier's predictions across several test tasks of each dataset, in Figure 5. Specifically, each column corresponds to a test dataset, and the different rows show the variance of the dataset classifier's predictions over the 8 dimensions of its output vector (one for each of the M training datasets). We observe that there is no variance for the first 8 columns (seen datasets) since, as expected, the dataset classifier is accurate on the seen datasets and successfully predicts the training dataset from which each support set originates from. Out of the held-out ones, we observe that MNIST also has very low variance (it always picks Quickdraw as we can see from Figure 6), but the remaining held-out datasets exhibit larger variance, especially Traffic Signs where the dataset classifier's estimate of whether support sets from test episodes of Traffic Signs belong to the Flower dataset really vary from task to task.

Additional confidence intervals

We omitted the confidence intervals of Table 3 from the main paper due to space constraints, so we report a copy of that table along with the 95% confidence intervals here in Table 4.

Additional runs of the dataset classifier network

Since we noticed in Table 3 of the main paper that the initialization of Ψ_{d^*} has a large effect on performance, here we evaluate different checkpoints of our dataset classifier. Specifically, we performed different runs when training the dataset classifier, each with different hyperparameters, as outlined in the previous section. In what follows, we present the results not only of the top-performing one (in terms of validation accuracy), but the five top-performing ones. This allows us to understand the sensitivity of our results to the choice of the specific checkpoint of the dataset classifier that we use. We show these results in Tables 6 and 7, with and without fine-tuning of Ψ_{d^*}, respectively. That is, the former sets the Blender's proposal as Ψ_{d^*} directly, instead of treating that as the initialization for fine-tuning via gradient descent, as we do in the latter. The last column of each table corresponds to the checkpoint of the dataset classifier that we used for our results in the main paper.

To generate the results in Table 7, for each of the 5 checkpoints of the dataset classifier, we performed a validation round where we used the performance on validation episodes to determine the learning rate and number of steps that will be used for fine-tuning. The hyperparameters that worked best for the 5 different checkpoints and their respective validation accuracies are shown in Table 5. These validation accuracies are averaged over a large number of validation episodes (600 per dataset in the validation set), where as a reminder the validation set contains held-out classes of the M training datasets.

From Table 6, we observe that the results are reasonably consistent across the 5 checkpoints of the dataset classifier that we consider. This is especially true of the performance on (held-out classes of) the seen datasets, in rows ImageNet-Flower (weak generalization setting). On the unseen datasets in rows Traffic Signs - CIFAR-100 (strong generalization setting), there is some more variance, as expected. This is because of the fact that during training (and validation), the dataset classifier is not exposed to any data from these held-out datasets, so its behavior is underspecified in that regard, and it is plausible that different solutions perform equally well on the training and validation sets, but behave differently on the held-out datasets of the test set. Nevertheless, we find the observed variance reasonable (the difference between the best and worst performing runs is at most 0.1% on average WG, at most 1.7% on Average SG, and at most 0.7% on the overall average).

Next, we look at Table 7, where there is an additional potential source of variance coming from the additional fine-tuning phase and the difference in the hyperparameters that were chosen for the different runs. However, we still find the observed variance reasonable (the difference between the best and worst performing runs is at most 0.2% on average WG, at most 2.7% on average SG, and at most 1% on the overall average). We note that even our worst variant outperforms the previous state-of-the-art on average, and in

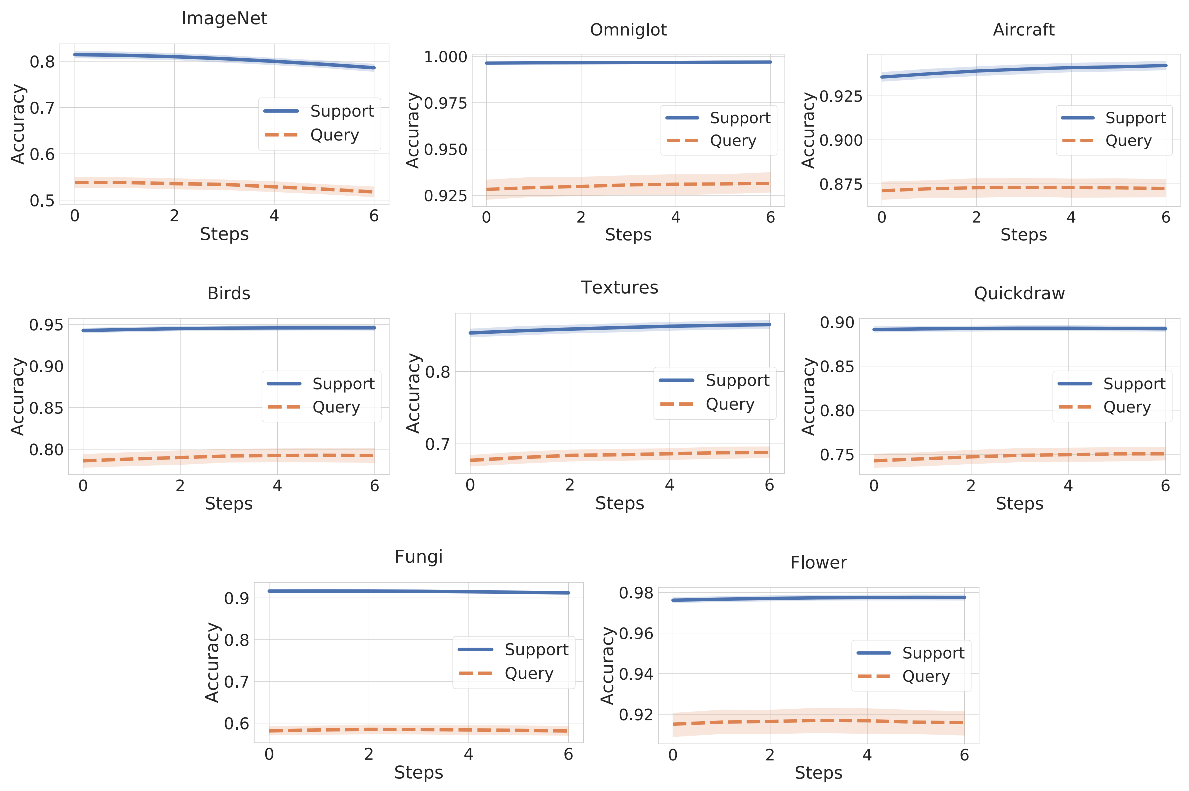

Figure 4. The support and query accuracy over 600 test episodes of seen datasets (weak generalization setting) as a function of the fine-tuning steps for Ψ_{d^*}.

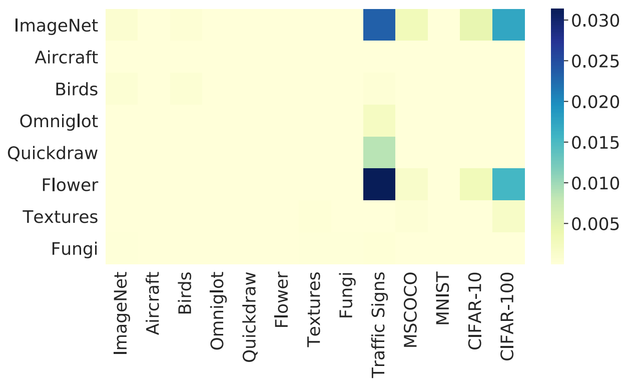

Figure 5. Visualization of the variance of the co-efficients that the Blender produces for each dataset over 600 test episodes of that dataset. This aids us to understand how much the dataset classifier's predictions change based on the specific support set that it ingests. As a reminder, the dataset classifier makes predictions based on the support set of each given test task, so the resulting combination co-efficients are not re-used across different tasks of the same dataset (in fact, the dataset identity is not known at test time).

fact with a large margin on the problem of few-shot dataset generalization that we study in this work ("Average SG"). These additional runs therefore further support FLUTE's effectiveness.

We encourage future work to also report the performance across several runs. We believe that there is an inherent underspecification in few-shot dataset generalization, due to the large gap between the training (and validation) data compared to the test data. (Gulrajani & Lopez-Paz, 2020) also offer an extensive discussion on the difficulty of model selection in the difficult regime of the domain generalization problem that they study, which is closely related to our setup. Given these difficulties, we believe it is important to report the variance of our approaches, instead of reporting only the accuracy of the top-performing run.

A closer look at the comparison between FLUTE and SUR-pf

As a reminder, SUR-pf makes the design choice of training the parametric family parameters only on ImageNet, and subsequently training a separate set of FiLM parameters for each other dataset, but without modifying the shared convolutional layer parameters. In this next experiment, we

Table 4. The effect of training on different data ('All', as in FLUTE, or 'ImageNet-only'), and alternative initialization schemes for Ψ_{d*}: from scratch ('scratch'), from ImageNet's FiLM parameters ('Ψ_{IN}') and from Blender. The column corresponding to the setting "All, Blender" is our proposed FLUTE model. This is the same Table as 3 from the main paper, but additionally annotated with confidence intervals.

Training data		All			ImageNet only	
Init scheme	Scratch	Ψ_{IN}	Blender	Scratch	Ψ_{IN}	Blender
ImageNet	47.7 ± 1.1	**53.9** ± 1.1	51.8 ± 1.1	34.2 ± 1.0	46.9 ± 1.1	46.9 ± 1.1
Omniglot	91.2 ± 0.6	75.6 ± 1.1	**93.2** ± 0.5	57.2 ± 1.4	61.6 ± 1.4	77.9 ± 1.1
Aircraft	80.6 ± 0.8	66.0 ± 0.9	**87.2 ± 0.5**	35.3 ± 0.8	48.5 ± 1.0	67.3 ± 0.8
Birds	72.0 ± 0.9	73.2 ± 0.9	**79.2 ± 0.8**	28.8 ± 0.8	47.9 ± 1.0	60.0 ± 0.9
Textures	69.8 ± 0.7	**71.9** ± 0.7	68.8 ± 0.8	55.5 ± 0.7	63.8 ± 0.8	61.2 ± 0.7
Quickdraw	78.1 ± 0.7	69.3 ± 0.8	**79.5** ± 0.7	50.6 ± 1.0	57.5 ± 1.0	60.9 ± 0.9
Fungi	51.6 ± 1.1	46.0 ± 1.1	**58.1** ± 1.1	23.0 ± 0.9	31.8 ± 1.0	34.3 ± 1.0
Flower	**91.0** ± 0.6	89.2 ± 0.6	**91.6 ± 0.6**	65.9 ± 1.0	80.1 ± 0.9	73.8 ± 0.8
Traffic Signs	**56.2** ± 1.1	54.8 ± 1.1	**58.4 ± 1.1**	36.4 ± 1.0	46.5 ± 1.1	43.0 ± 1.1
MSCOCO	42.5 ± 1.0	**50.6** ± 1.0	**50.0** ± 1.0	29.3 ± 0.9	41.4 ± 1.0	41.3 ± 1.0
MNIST	**95.5** ± 0.5	83.4 ± 0.7	**95.6 ± 0.4**	78.9 ± 0.7	80.8 ± 0.8	86.9 ± 0.6
CIFAR-10	69.6 ± 0.9	76.9 ± 0.7	**78.6 ± 0.7**	47.1 ± 0.8	65.4 ± 0.8	65.4 ± 0.8
CIFAR-100	58.0 ± 1.1	**67.5** ± 0.9	**67.1 ± 1.0**	34.5 ± 1.0	52.7 ± 1.1	52.2 ± 1.1
Average WG	72.8	68.1	**76.2**	43.8	54.8	60.3
Average SG	64.4	66.7	**69.9**	45.2	57.4	57.8
Average all	69.5	67.6	**73.8**	44.4	55.8	59.3

Table 5. The learning rate and number of steps that were deemed best (as per the validation set accuracy) for each of the 5 checkpoints of the dataset classifier, as well as their associated validation set accuracy. These are the hyperparameters of the fine-tuning phase that were used to generate the results of Table 7.

	Run 1	Run 2	Run 3	Run 4	Run 5
learn rate	1e-3	5e-4	5e-3	5e-4	5e-3
num steps	4	6	2	10	6
valid acc	77.6	77.5	77.5	77.5	78.1

run a variant of SUR-pf that trains on all datasets in the same way as FLUTE (we re-used the parameteric family we trained for FLUTE to achieve this). The only difference between this variant and FLUTE, then, is the algorithm for tackling each test task: FLUTE will learn a new set of batch normalization parameters for the task at hand, as described in the main paper. SUR-pf, on the other hand, creates a 'universal representation' by concatenating the activations of the different backbones (that share some but not all of their parameters) and applying SUR's selection mechanism to weigh the universal representation features appropriately for the task at hand (Dvornik et al., 2020). The results of this comparison are shown in Table 8. While our modified variant of SUR-pf outperforms the original SUR-pf, it still significantly falls short of FLUTE, especially on the strong generalization tasks. This suggests that FLUTE's superiority over SUR is not solely due to training on more data, but also due to its inductive bias that is particularly appropriate for the problem of few-shot dataset generalization.

Shuffled Traffic Signs

It was recently noticed [4] that in the introduction notebook that comes with the Meta-Dataset code-base[5], the usage examples given for the episode input pipeline did not set the parameter that dictates the size of the shuffle buffer, which defaults to not shuffling examples within each class. This led to many previous works on Meta-Dataset using unshuffled datasets, which evidently produced more optimistic results on the Traffic Signs dataset. Specifically, the examples of this dataset are organized as 30-image sequences of pictures from the same physical sign (successive frames from the same video), leading to support and query examples being more frequently really close when not shuffling the examples of each class.

The results we reported in this paper are computed as intended, using the shuffled datasets. For reference, there is also a leaderboard on the Meta-Dataset code-base repository that reflects the shuffled Traffic Signs numbers for different methods.

However, for completeness, we show here the results computed on the easier variant induced by not shuffling the images. These are in Table 9 (main results) and Table 10 (additional runs of the dataset classifier). The latter displays the results of the same 5 checkpoints as we used in the previous section. The last column represents the model that we used to report FLUTE's results in the main paper.

[4]https://github.com/google-research/meta-dataset/issues/54
[5]https://github.com/google-research/meta-dataset

Table 6. The performance of FLUTE when using each of 5 different checkpoints of the dataset classifier. In this case, we omit the fine-tuning phase, and treat the Blender's proposal directly as the FiLM parameters of the new task (instead of treating that as the initialization for further fine-tuning). This allows us to more closely inspect the difference in performance induced by different dataset classifiers. As usual, we report the performance on the test set of each of the seen datasets (ImageNet - Flower) for the weak generalization setting (WG), and the performance on the held-out datasets (Traffic Signs - CIFAR-100) for the strong generalization setting (SG), corresponding to the problem of few-shot dataset generalization that we focus on in this work.

Dataset	Run 1	Run 2	Run 3	Run 4	Run 5
ImageNet	53.4 ± 1.1	53.9 ± 1.1	53.9 ± 1.1	53.4 ± 1.1	53.8 ± 1.1
Omniglot	92.8 ± 0.5	92.8 ± 0.5	92.8 ± 0.5	92.8 ± 0.5	92.8 ± 0.5
Aircraft	87.1 ± 0.5	87.1 ± 0.5	87.1 ± 0.5	87.1 ± 0.5	87.1 ± 0.5
Birds	78.6 ± 0.8	78.6 ± 0.8	78.6 ± 0.8	78.5 ± 0.8	78.6 ± 0.8
Textures	67.7 ± 0.8	67.7 ± 0.8	67.7 ± 0.8	67.8 ± 0.8	67.7 ± 0.8
Quickdraw	79.6 ± 0.7	79.6 ± 0.7	79.6 ± 0.7	79.6 ± 0.7	79.6 ± 0.7
Fungi	58.1 ± 1.1	58.2 ± 1.1	58.2 ± 1.1	58.0 ± 1.1	58.1 ± 1.1
Flower	91.5 ± 0.6	91.5 ± 0.6	91.5 ± 0.6	91.5 ± 0.6	91.5 ± 0.6
Traffic Signs	51.8 ± 1.1	52.6 ± 1.1	51.8 ± 1.1	52.0 ± 1.1	53.8 ± 1.1
MSCOCO	48.7 ± 1.0	50.2 ± 1.0	50.3 ± 1.0	49.3 ± 1.0	50.1 ± 1.0
MNIST	96.0 ± 0.4	94.3 ± 0.5	94.3 ± 0.5	94.3 ± 0.5	94.3 ± 0.5
CIFAR-10	75.8 ± 0.7	75.1 ± 0.8	75.7 ± 0.8	73.8 ± 0.8	76.4 ± 0.7
CIFAR-100	66.3 ± 1.0	63.4 ± 1.0	65.1 ± 1.0	62.9 ± 1.0	66.4 ± 1.0
Average WG	76.1	76.2	76.2	76.1	76.2
Average SG	67.7	67.1	67.4	66.5	68.2
Average all	72.9	72.7	72.8	72.4	73.1

Table 7. The performance of FLUTE when using each of 5 different checkpoints of the dataset classifier. Contrary to Table 6, we now perform the fine-tuning phase too that learns the FiLM parameters for the new task, starting from the Blender's proposed initialization. Table 5 shows the hyperparameters used for fine-tuning for each of the 5 runs, and their respective validation accuracies. As usual, we report the performance on the test set of each of the seen datasets (ImageNet - Flower) for the weak generalization setting (WG), and the performance on the held-out datasets (Traffic Signs - CIFAR-100) for the strong generalization setting (SG), corresponding to the problem of few-shot dataset generalization that we focus on in this work. The results for the rightmost column (Run 5) are the results we reported for FLUTE in the main paper, since this run achieved the highest validation accuracy as shown in Table 5.

Dataset	Run 1	Run 2	Run 3	Run 4	Run 5
ImageNet	53.4 ± 1.1	53.8 ± 1.1	53.6 ± 1.1	53.4 ± 1.1	51.8 ± 1.1
Omniglot	92.9 ± 0.5	92.9 ± 0.5	93.0 ± 0.5	92.9 ± 0.5	93.2 ± 0.5
Aircraft	87.2 ± 0.5	87.2 ± 0.5	87.3 ± 0.5	87.2 ± 0.5	87.2 ± 0.5
Birds	78.8 ± 0.8	78.8 ± 0.8	79.0 ± 0.8	78.8 ± 0.8	79.2 ± 0.8
Textures	68.0 ± 0.8	68.0 ± 0.8	68.3 ± 0.8	68.1 ± 0.8	68.8 ± 0.8
Quickdraw	79.6 ± 0.7	79.6 ± 0.7	79.6 ± 0.7	79.6 ± 0.7	79.5 ± 0.7
Fungi	58.2 ± 1.1	58.3 ± 1.1	58.5 ± 1.1	58.3 ± 1.1	58.1 ± 1.1
Flower	91.6 ± 0.6	91.6 ± 0.6	91.6 ± 0.6	91.6 ± 0.6	91.6 ± 0.6
Traffic Signs	52.5 ± 1.1	53.2 ± 1.1	54.5 ± 1.1	53.4 ± 1.1	58.4 ± 1.1
MSCOCO	49.0 ± 1.0	50.4 ± 1.0	50.8 ± 1.0	49.7 ± 1.0	50.0 ± 1.0
MNIST	96.0 ± 0.4	94.5 ± 0.5	94.9 ± 0.5	94.7 ± 0.5	95.6 ± 0.5
CIFAR-10	76.6 ± 0.7	75.7 ± 0.8	77.4 ± 0.8	74.8 ± 0.8	78.6 ± 0.7
CIFAR-100	66.8 ± 1.0	63.8 ± 1.0	66.2 ± 1.0	63.5 ± 1.0	67.1 ± 0.9
Average WG	76.2	76.3	76.4	76.2	76.2
Average SG	68.2	67.5	68.8	67.2	69.9
Average all	73.1	72.9	73.4	72.8	73.8

Table 8. Comparison of FLUTE to SUR-pf and a different variant of SUR-pf that we ran ('All SUR-pf') whose parametric family is trained on all datasets, in the same way as FLUTE, instead of being trained on ImageNet only as SUR-pf is.

Dataset	FLUTE	SUR-pf	All SUR-pf
ImageNet	51.8 ± 1.1	**56.4** ± 1.2	54.4 ± 1.1
Omniglot	**93.2** ± 0.5	88.5 ± 0.8	92.0 ± 0.6
Aircraft	**87.2 ± 0.5**	79.5 ± 0.8	**87.0 ± 0.6**
Birds	**79.2 ± 0.8**	76.4 ± 0.9	**79.4 ± 0.8**
Textures	68.8 ± 0.8	**73.1 ± 0.7**	**72.3 ± 0.7**
Quickdraw	**79.5** ± 0.7	75.7 ± 0.7	**79.1 ± 0.7**
Fungi	**58.1** ± 1.1	48.2 ± 0.9	54.4 ± 1.1
Flower	**91.6 ± 0.6**	90.6 ± 0.5	**91.9 ± 0.6**
Traffic Signs	**58.4 ± 1.1**	52.2 ± 0.8	49.4 ± 1.1
MSCOCO	**50.0** ± 1.0	**52.1 ± 1.0**	47.6 ± 1.0
MNIST	**95.6 ± 0.4**	93.2 ± 0.4	**95.8 ± 0.4**
CIFAR-10	**78.6 ± 0.7**	66.4 ± 0.8	66.2 ± 0.8
CIFAR-100	**67.1 ± 1.0**	57.1 ± 1.0	56.4 ± 1.0
Average WG	**76.2**	73.6	76.3
Average SG	**69.9**	64.2	63.1
Average all	**73.8**	70.0	71.2

Table 9. Comparing FLUTE to recent state-of-the-art methods. This is the same table as Table 2 in the main paper, with the exception of the Traffic Signs row that now reflects the easier (unshuffled) variant of Traffic Signs.

Dataset	CNAPs	TaskNorm	SimpleCNAPs	SUR-pf	URT-pf	SUR (x8)	URT (x8)	FLUTE
ImageNet	52.3 ± 1.0%	50.6 ± 1.1%	**58.6 ± 1.1%**	56.4 ± 1.2%	55.5 ± 1.1%	56.3 ± 1.1%	55.7 ± 1.1%	51.8 ± 1.1%
Omniglot	88.4 ± 0.7%	90.7 ± 0.6%	91.7 ± 0.6%	88.5 ± 0.8%	90.2 ± 0.6%	93.1 ± 0.5%	**94.4 ± 0.4%**	93.2 ± 0.5%
Aircraft	80.5 ± 0.6%	83.8 ± 0.6%	82.4 ± 0.7%	79.5 ± 0.8%	79.8 ± 0.7%	85.4 ± 0.7%	85.8 ± 0.6%	**87.2 ± 0.5%**
Birds	72.2 ± 0.9%	74.6 ± 0.8%	74.9 ± 0.8%	76.4 ± 0.9%	77.5 ± 0.8%	71.4 ± 1.0%	76.3 ± 0.8%	**79.2 ± 0.8%**
Textures	58.3 ± 0.7%	62.1 ± 0.7%	67.8 ± 0.8%	**73.1 ± 0.7%**	**73.5 ± 0.7%**	71.5 ± 0.8%	71.8 ± 0.7%	68.8 ± 0.8%
Quickdraw	72.5 ± 0.8%	74.8 ± 0.7%	77.7 ± 0.7%	75.7 ± 0.7%	75.8 ± 0.7%	81.3 ± 0.6%	**82.5 ± 0.6%**	79.5 ± 0.7%
Fungi	47.4 ± 1.0%	48.7 ± 1.0%	46.9 ± 1.0%	48.2 ± 0.9%	48.1 ± 0.9%	**63.1 ± 1.0%**	**63.5 ± 1.0%**	58.1 ± 1.1%
Flower	86.0 ± 0.5%	89.6 ± 0.6%	90.7 ± 0.5%	90.6 ± 0.5%	**91.9 ± 0.5%**	82.8 ± 0.7%	88.2 ± 0.6%	**91.6 ± 0.6%**
Traffic Signs	60.2 ± 0.9%	67.0 ± 0.7%	73.5 ± 0.7%	65.1 ± 0.8%	67.5 ± 0.8%	70.4 ± 0.8%	69.4 ± 0.8%	**74.8 ± 0.7%**
MSCOCO	42.6 ± 1.1%	43.4 ± 1.0%	46.2 ± 1.1%	**52.1 ± 1.0%**	**52.1 ± 1.0%**	**52.4 ± 1.1%**	**52.2 ± 1.1%**	50.0 ± 1.0%
MNIST	92.7 ± 0.4%	92.3 ± 0.4%	93.9 ± 0.4%	93.2 ± 0.4%	93.9 ± 0.4%	94.3 ± 0.4%	94.8 ± 0.4%	**95.6 ± 0.5%**
CIFAR-10	61.5 ± 0.7%	69.3 ± 0.8%	74.3 ± 0.7%	66.4 ± 0.8%	66.1 ± 0.8%	66.8 ± 0.9%	67.3 ± 0.8%	**78.6 ± 0.7%**
CIFAR-100	50.1 ± 1.0%	54.6 ± 1.1%	60.5 ± 1.0%	57.1 ± 1.0%	57.3 ± 1.0%	56.6 ± 1.0%	56.9 ± 1.0%	**67.1 ± 1.0%**
Average WG	69.7 %	71.9 %	73.8 %	73.6 %	74.0 %	75.6 %	**77.3 %**	76.2 %
Average SG	61.4 %	65.3 %	69.7 %	66.8 %	67.4 %	68.1 %	68.1 %	**73.2 %**
Average all	66.5 %	69.3 %	72.2 %	70.9 %	71.5 %	72.7 %	73.8 %	**75.0 %**

Table 10. The performance of FLUTE when using each of 5 different checkpoints of the dataset classifier on the (easier) unshuffled version of the Traffic Signs dataset. The results for the rightmost column (Run 5) are the results produced by the FLUTE variant that we report in the main paper.

Dataset	Run 1	Run 2	Run 3	Run 4	Run 5
Unshuffled Traffic Signs	73.1 ± 0.7%	73.3 ± 0.7%	73.0 ± 0.7%	72.9 ± 0.7%	74.8 ± 0.7%

Hard Blender: using the dataset classifier but without taking a convex combination

An alternative design choice is to use a ‘Hard’ Blender, that instead of taking a convex combination of the training datasets’ FiLM parameters, selects only the FiLM parameters of the most likely training dataset (as assessed by the dataset classifier). The comparison with this variant is shown in Table 11. Perhaps unsurprisingly, the two initialization schemes perform similarly. This is expected, especially for the WG tasks, since the Blender, which is based on an accurate dataset classifier, puts most of its probability mass on a single dataset anyway. Taking the convex combination is a more general approach that doesn’t suffer, and in fact may slightly be beneficial for some SG tasks. We therefore adopt this initialization scheme as our default one for use with FLUTE.

Table 11. Comparing the Blender initialization scheme to the 'Hard Blender' variant. Specifically, instead of taking a convex combination of the training datasets' FiLM parameters as Blender does, 'Hard Blender' selects only the FiLM parameters of the most likely training dataset (as assessed by the dataset classifier). The columns marked as 'fine-tune' train the FiLM parameters using gradient descent from the Blender or Hard Blender initialization, whereas the others use the initialization directly, allowing to more closely inspect the difference between these two initialization schemes.

Dataset	Blender	Blender (fine-tune)	Hard Blender	Hard Blender (fine-tune)
ImageNet	53.8 ± 1.1	51.8 ± 1.1	53.9 ± 1.1	51.8 ± 1.1
Omniglot	92.8 ± 0.5	93.2 ± 0.5	92.8 ± 0.5	93.2 ± 0.5
Aircraft	87.1 ± 0.5	87.2 ± 0.5	87.1 ± 0.5	87.3 ± 0.5
Birds	78.6 ± 0.8	79.2 ± 0.8	78.6 ± 0.8	79.2 ± 0.8
Textures	67.7 ± 0.8	68.8 ± 0.8	67.7 ± 0.8	68.8 ± 0.8
Quickdraw	79.6 ± 0.7	79.5 ± 0.7	79.6 ± 0.7	79.5 ± 0.7
Fungi	58.1 ± 1.1	58.1 ± 1.1	58.2 ± 1.1	58.1 ± 1.1
Flower	91.5 ± 0.6	91.6 ± 0.6	91.5 ± 0.6	91.6 ± 0.6
Traffic Signs	53.8 ± 1.1	58.4 ± 1.1	52.4 ± 1.1	57.2 ± 1.1
MSCOCO	50.1 ± 1.0	50.0 ± 0.7	50.4 ± 1.0	50.1 ± 1.0
MNIST	94.3 ± 0.5	95.6 ± 1.0	94.3 ± 0.5	95.6 ± 0.5
CIFAR-10	76.4 ± 0.7	78.6 ± 0.5	76.5 ± 0.7	78.4 ± 0.7
CIFAR-100	66.4 ± 1.0	67.1 ± 0.7	67.0 ± 0.9	66.7 ± 1.0
Average WG	76.2	76.2	76.2	76.2
Average SG	68.2	69.9	68.1	69.6
Average all	73.1	73.8	73.1	73.7

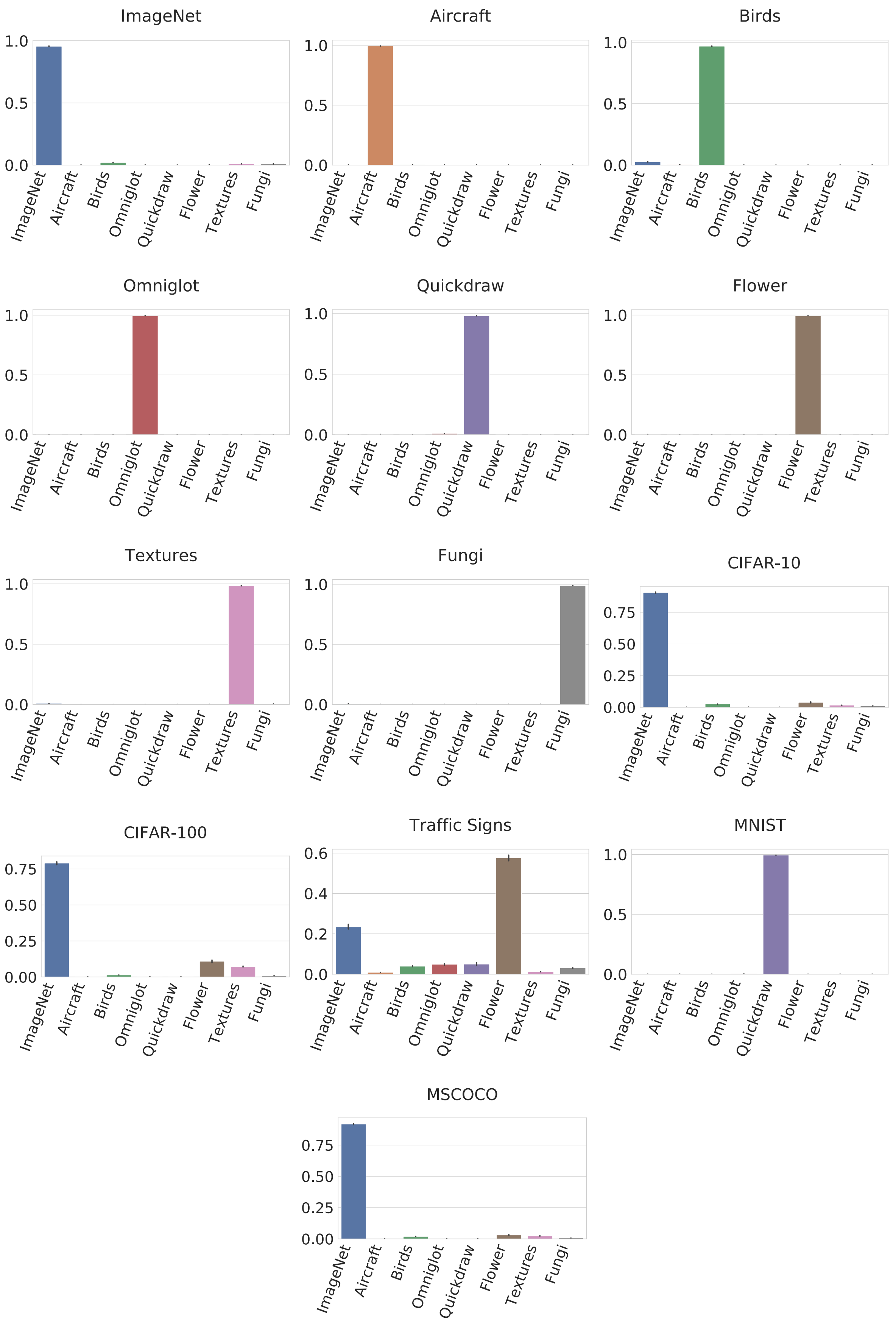

Figure 6. The combination co-efficients that the Blender outputs for test episodes of each dataset. Each plot is creating using 600 test episodes of its corresponding dataset.