A. Implementation Details

This section describes the implementation details for each dataset.

A.1. CIFAR-10

CIFAR-10 consists of consists of 60,000 images of size $32 \times 32 \times 3$ from ten classes. The linear models were trained using a max learning rate of 0.06 for all parameters on 5K, 1K, 500, and 500 epochs for 100, 500, 1K, 50K samples, respectively. The hybrid WRN models were trained using a max learning rate of 0.1 on 3K, 2K, 1K, and 200 epochs for 100, 500, 1K, and 50K samples respectively. We use batch gradient descent except when the models are trained with 50K samples where we use mini-batch gradient descent of size 1024. On the entire training set, we also train the models on 10 seeds and, in all cases, the standard errors are always inferior to 0.3. All scattering networks use a spatial scale $J = 2$.

A.2. COVIDx CRX-2

COVIDx CRX-2 is a two-class (positive and negative) dataset of $1024 \times 1024 \times 1$ chest X-Ray images of COVID-19 patients [42]. In our experiments, we always train on a class-balanced subset of the training set. We resize the images to 260×260 and train our network with random crops of 224×224 pixels. The only data augmentation we use is random horizontal flipping. All models were trained on 400 epochs using a max learning rate of 0.01. All hybrid models are trained with a mini-batch size of 128. All scattering networks use a spatial scale $J = 4$.

A.3. KTH-TIPS2

We resize the images to 200×200 and train our network with random crops of 128×128 pixels. The training data is augmented with random horizontal flips and random rotations. All scattering networks use a spatial scale of 4. We set the maximum learning rate of the scattering parameters to 0.1 while it is set to 0.001 for all other parameters. All hybrid models are trained with a mini-batch size of 128. The hybrid linear models are trained for 250 epochs, while the hybrid WRN models are trained for 150 epochs. We evaluate each model, training it with four different seeds on each *sample* of material, amounting to 16 total runs. All scattering networks use a spatial scale $J = 4$.

B. Wide Residual Network Hybrid Architecture

In the experiments of Sec. 4, the scattering networks are combined with a WRN hybrid described in [31]. We follow a similar architecture to the WRN hybrid used in [33]. The description of the architecture used for the experiments is given in Table 6. We use the same architecture for all the datasets. The architecture consists of a scattering network that greatly reduces the spatial resolution of the input followed by a WRN. The CIFAR-10 scattering stage yields output with 8×8 spatial resolution (scattering with J=2). Similarly, the KTH-TIPS2 data and COVIDx-CRX2 data give outputs with 16×16 and 8×8 spatial resolutions respectively.

Stage	Description
scattering	Learned or Not Learned
conv1	3×3, Conv Layer $128 \rightarrow 256$
conv2	$\begin{bmatrix} 3 \times 3, \text{Conv Layer } 256 \\ 3 \times 3, \text{Conv Layer } 256 \end{bmatrix} \times 4$
conv3	$\begin{bmatrix} 3 \times 3, \text{Conv Layer } 256 \\ 3 \times 3, \text{Conv Layer } 256 \end{bmatrix} \times 4$
avg-pool	Avg pooling to a size 1x1

Table 6. Description of the WRN hybrid architecture used for the experiments. Each convolutional layer represents a 2-dimensional convolution followed by a batch normalization and a ReLU non-linearity function.

C. Backpropagation through the Parametric Scattering Network

We show here that it is possible to backpropagate through this construction. Namely, we verify the differentiability of this construction by explicitly computing the partial derivatives with respect to these parameters. First, the $\mathbb{R}$-linear derivative of the complex modulus $f(z) = |z|$ is $f'(z) = \frac{z}{|z|}$. Next, we show the differentiation of convolution with wavelets with respect to their parameters. For simplicity, we focus here on differentiation of the Gabor portion[1] of the filter construction from Eq. 1, written as:

$$\begin{aligned}\varphi(u) =& \exp(-\frac{1}{2\sigma^2}(u_1^2(\cos^2(\theta)+\sin^2(\theta)\gamma^2) \\ &+ u_2^2(\cos^2(\theta)\gamma^2+\sin^2(\theta)) \\ &+ 2\cos(\theta)\sin(\theta)u_1u_2(1-\gamma^2)) \\ &+ i\xi(\cos(\theta)u_1+\sin(\theta)u_2)).\end{aligned} \tag{4}$$

Its derivatives with respect to the parameters are

$$\begin{aligned}\frac{\partial\varphi}{\partial\theta}(u) =& \frac{1}{\sigma^2}(u_2\cos\theta - u_1\sin\theta)(i\xi\sigma^2 + u_1(\gamma^2-1)\cos\theta \\ &+ u_2(\gamma^2-1)\sin\theta)\varphi(u);\end{aligned} \tag{5}$$

$$\begin{aligned}\frac{\partial\varphi}{\partial\sigma}(u) =& \frac{1}{\sigma^3}(u_1^2(\cos^2\theta+\gamma^2\sin^2\theta) + u_2^2(\gamma^2\cos^2\theta+\sin^2\theta) \\ &+ 2u_1u_2\cos\theta\sin\theta(1-\gamma^2))\varphi(u);\end{aligned} \tag{6}$$

$$\frac{\partial\varphi}{\partial\xi}(u) = i(u_1\cos\theta + u_2\sin\theta)\varphi(u); \text{ and} \tag{7}$$

$$\begin{aligned}\frac{\partial\varphi}{\partial\gamma}(u) =& -\frac{1}{\sigma^2}(u_1^2\gamma\sin^2\theta + u_2^2\gamma\cos^2\theta \\ &- 2u_1u_2\gamma\cos\theta\sin\theta)\varphi(u).\end{aligned} \tag{8}$$

Finally, the derivative of the convolution with such filters is given by $\frac{\partial}{\partial\zeta}(f*\varphi)(t) = \int f(t-u)\frac{\partial\varphi}{\partial\zeta}(u)du$ where ζ is any of the filter parameter from Table 1. It is easy to verify that these derivations can be chained together to propagate through the scattering cascades defined in Sec. 3.1. We can now learn these jointly with other parameters in an end-to-end differentiable architecture.

D. Equivariant Filters

We observe that, in some cases, using equivariant filters yields better accuracy, as shown in Table 2. Figure 4 illustrates equivariant filters initialized using tight frame construction before and after optimization. The scattering network is combined with a linear layer and trained on 500 training samples of CIFAR-10. The spatial scale is set to $J = 2$. Similarly, Figure 5 illustrates equivariant filters initialized randomly before and after optimization. In the two figures, each row corresponds to a different spatial scale (J). Since J is set to 2, we have two rows. We observe that the filters in each row are the exactly the same, except for the global orientation of the wavelet.

E. No Autoaugment Ablation

The training set of CIFAR-10 is augmented with pre-specified autoaugment in Table 2 to demonstrate the best possible results. To understand the effect of autoaugment, we replicate the same experiments except for not augmenting the training set with autoaugment. Table 7 reports the performance of the different architectures on CIFAR-10. We observe that the scattering networks followed by WRN underperform when no autoaugment is used. The difference in performance between using autoaugment and not using it is smaller when the scattering network is followed with a linear layer. Surprisingly, the performance of the scattering networks followed with a linear layer trained on all data increased without autoaugment. It seems that in the case of a scattering network followed by a linear model, autoaugment is not as useful as with a deep model on top and can also be harmful in some cases.

[1]It is not difficult to extend this derivation to Morlet wavelets, but the resulting expressions are rather cumbersome and left out for brevity.

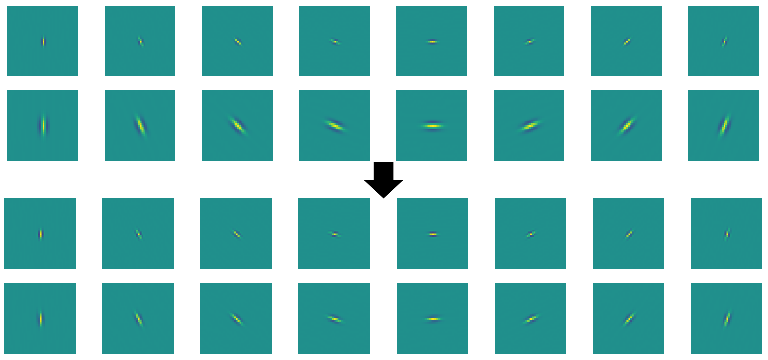

Figure 4. **Example of equivariant filters initialized using tight frame construction**. (Top) Real part of wavelet filters before optimization. (Bottom) Real part of wavelet filters after optimization.

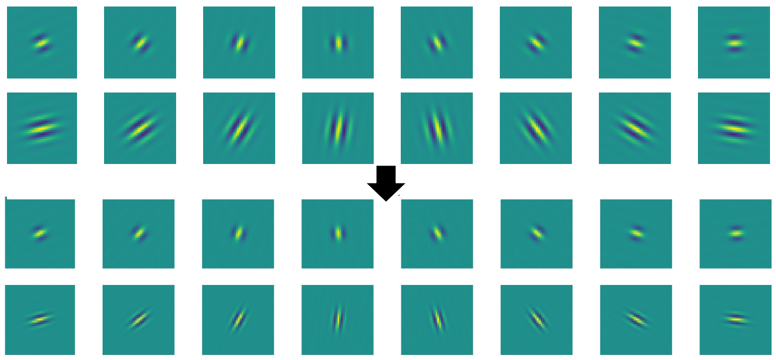

Figure 5. **Example of equivariant filters initialized randomly**. (Top) Real part of wavelet filters before optimization. (Bottom) Real part of wavelet filters after optimization.

Table 7. CIFAR-10 mean accuracy and std. error over 10 seeds with $J = 2$ and multiple training sample sizes. The table compares the effect of augmenting the training set with pre-specified autoaugment. When the scattering network is followed by a WRN, using autoaugment is necessary to obtain better performance.

Init.	Arch.	AA	100 samples	500 samples	1000 samples	All
TF	LS+LL†	Yes	37.84 ± 0.57	$\mathbf{52.68} \pm 0.31$	$\mathbf{57.43} \pm 0.17$	69.57 ± 0.1
TF	LS+LL†	No	$\mathbf{39.70} \pm 0.62$	50.74 ± 0.30	54.76 ± 0.22	$\mathbf{74.94} \pm 0.06$
TF	S +LL	Yes	36.01 ± 0.55	48.12 ± 0.25	53.25 ± 0.24	65.58 ± 0.04
TF	S +LL	No	$\mathbf{37.55} \pm 0.62$	$\mathbf{49.67} \pm 0.33$	$\mathbf{53.96} \pm 0.48$	$\mathbf{70.71} \pm 0.03$
Rand	LS+LL†	Yes	$\mathbf{34.81} \pm 0.60$	$\mathbf{49.6} \pm 0.39$	$\mathbf{55.72} \pm 0.39$	69.39 ± 0.41
Rand	LS+LL†	No	32.64 ± 0.38	42.88 ± 0.23	47.40 ± 0.32	$\mathbf{74.71} \pm 0.08$
Rand	S +LL	Yes	29.77 ± 0.47	$\mathbf{41.85} \pm 0.41$	$\mathbf{46.3} \pm 0.37$	57.72 ± 0.1
Rand	S +LL	No	$\mathbf{31.71} \pm 0.34$	40.57 ± 0.32	44.42 ± 0.51	$\mathbf{61.79} \pm 0.31$
TF	LS+WRN†	Yes	$\mathbf{43.60} \pm 0.87$	$\mathbf{63.13} \pm 0.29$	$\mathbf{70.14} \pm 0.26$	$\mathbf{93.61} \pm 0.12$
TF	LS+WRN†	No	34.95 ± 0.96	54.21 ± 0.39	62.17 ± 0.28	90.17 ± 0.34
TF	S +WRN	Yes	$\mathbf{43.16} \pm 0.78$	$\mathbf{61.66} \pm 0.32$	$\mathbf{68.16} \pm 0.27$	$\mathbf{92.27} \pm 0.05$
TF	S +WRN	No	35.15 ± 0.43	52.77 ± 0.35	60.72 ± 0.21	89.05 ± 0.38
Rand	LS+WRN†	Yes	$\mathbf{41.42} \pm 0.65$	$\mathbf{59.84} \pm 0.40$	$\mathbf{67.4} \pm 0.28$	$\mathbf{93.36} \pm 0.19$
Rand	LS+WRN†	No	31.08 ± 1.00	48.37 ± 0.76	55.41 ± 0.49	88.80 ± 0.47
Rand	S +WRN	Yes	$\mathbf{32.08} \pm 0.46$	$\mathbf{46.84} \pm 0.21$	$\mathbf{52.76} \pm 0.33$	$\mathbf{85.35} \pm 1.06$
Rand	S +WRN	No	27.73 ± 0.43	41.05 ± 0.32	47.19 ± 0.37	79.67 ± 0.59

†: ours TF: tight-frame LS: Learnable Scattering AA: Autoaugment
# params : 156k for S+LL; 155k for LS+LL; 11M for S+WRN; 22.6M LS+WRN; and 22.3M for WRN only

F. Cosine Loss Ablation

We replicate the experiments of Sec. 4 with learnable scattering networks followed by a WRN on CIFAR-10, COVIDx-CRX2, and KTH-TIPS2. We use the same parameters except for using the cosine loss function [8] instead of cross-entropy. The cosine loss is described in Sec. 2. Wavelet filters are initialized using the tight frame construction. Table 8 demonstrates the average accuracy on the three datasets. For CIFAR-10 and COVIDx-CRX2, the performance is lower when models are trained using the cosine loss. The same behavior is not observed when the models are trained on KTH-TIPS2. In fact, the performance increases slightly by using the cosine loss function. Thus, cosine loss can improve performance over small data regimes for some datasets.

Table 8. CIFAR-10, COVIDx-CRX2 and KTH-TIPS2 mean accuracy and std. error using cosine loss function.

Init.	Arch.	Dataset	Loss	100 samples	500 samples	1000 samples	1188 samples
TF	LS+WRN	CIFAR-10	CE	$\mathbf{43.6} \pm 0.87$	$\mathbf{63.13} \pm 0.29$	$\mathbf{70.14} \pm 0.26$	-
TF	LS+WRN	CIFAR-10	Cosine	42.94 ± 0.77	61.42 ± 0.26	68.29 ± 0.18	-
TF	LS+WRN	COVIDx	CE	$\mathbf{81.20} \pm 1.73$	$\mathbf{90.50} \pm 0.70$	$\mathbf{93.68} \pm 0.35$	-
TF	LS+WRN	COVIDx	Cosine	80.03 ± 2.16	89.53 ± 0.89	92.75 ± 0.65	-
TF	LS+WRN	KTH-TIPS2	CE	-	-	-	69.23 ± 0.67
TF	LS+WRN	KTH-TIPS2	Cosine	-	-	-	$\mathbf{70.86} \pm 0.67$

TF: tight-frame LS: Learnable Scattering S: Scattering CE: Cross-Entropy Loss

G. Robustness to Deformations

In Section 4.2, we investigated if the parametric scattering transform is stable to small deformations (i.e., rotation, scale and shear). Here, we explore additional deformations. Models used were pre-trained and evaluated using the COVIDxCRX-2 setup mentioned in Section 4.2 with a linear classifer head. The additional deformations explored are translation and several diffeomorphisms (denoted Custom 1 and Custom 2). The strengths for the all the deformations used ranges from 0 to the maximum value given in Table 9, except for scale which goes from 1 to its maximum value. Additional results are depicted in Figure 6.

Custom 1, $\tau_\epsilon^1(u)$, and Custom 2, $\tau_\epsilon^2(u)$, are defined as such:

$$\tau_\epsilon^1(u) = \epsilon \begin{bmatrix} 0.3u_1^2 + 0.2u_2^2 \\ 0.2(0.2u_1) \end{bmatrix}, \tau_\epsilon^2(u) = \epsilon \begin{bmatrix} 0.3(u_1^2 + u_2^2) \\ -0.3(2u_1u_2) \end{bmatrix}.$$

Table 9. Deformations and their maximum value

Deformation	Maximum Value
Custom1	1
Custom2	1
Rotation	10
Scale	1.4
Shear	5
Translation	22

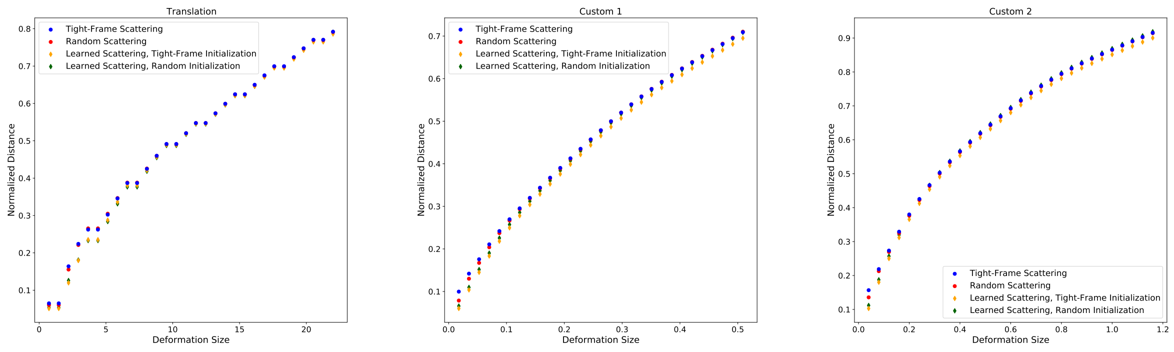

Figure 6. **Normalized distances between scattering representations of an image and its deformation.** (Left) Translation. (Middle) Custom 1 Transformation. (Right) Custom 2 Transformation.

H. Details of Training Unsupervised Scattering with SimCLR Objective

The learnable scattering networks with tight-frame and random initializations were pretrained on CIFAR-10 via the SimCLR method using a temperature of 0.5 and batch size of 128 for 500 epochs. The following basic augmentations were used as part of the SimCLR augmentation pipeline: random crop and resize, random flip, and color distortion. The optimizer used was Adam, with a learning rate of 1e-3, similar to settings from [13]. Scattering weights were then frozen and used as the backbone for a linear evaluation. For linear evaluation, we used SGD with the same optimization settings as our experiments on CIFAR-10.

I. Dataset Specific parameterizations

The following figures (7,8,9,10,11) show individual filter assignments obtained by hungarian matching as described in in 4.1. They also show the individual parameter values of each filter by adopting a common naming scheme for each title. The first letter of the title is either F (fixed filters) or O (optimized filters). The next character is always a number and corresponds to the ID of the match. For all Oixxxxxxx titles, there will be a corresponding Fixxxxxxxx title; these filters correspond to the ith match. The next character is always D (distance). It is superseded by a numerical value, the Morlet wavelet distance (see 4.1) between the filter and its match. The next character is the Gaussian window scale σ, followed by a number corresponding to the magnitude of the distances between the σ parameters of both filters. The next character is the aspect ratio γ, followed by a number corresponding to the magnitude of the distances between the γ parameters of both filters. The next character is the frequency scale ξ, followed by a number corresponding to the magnitude of the distances between the ξ parameters of both filters. We omit the filter orientation θ as it is easy enough to perceive visually.

I.1. COVIDX-CRX2

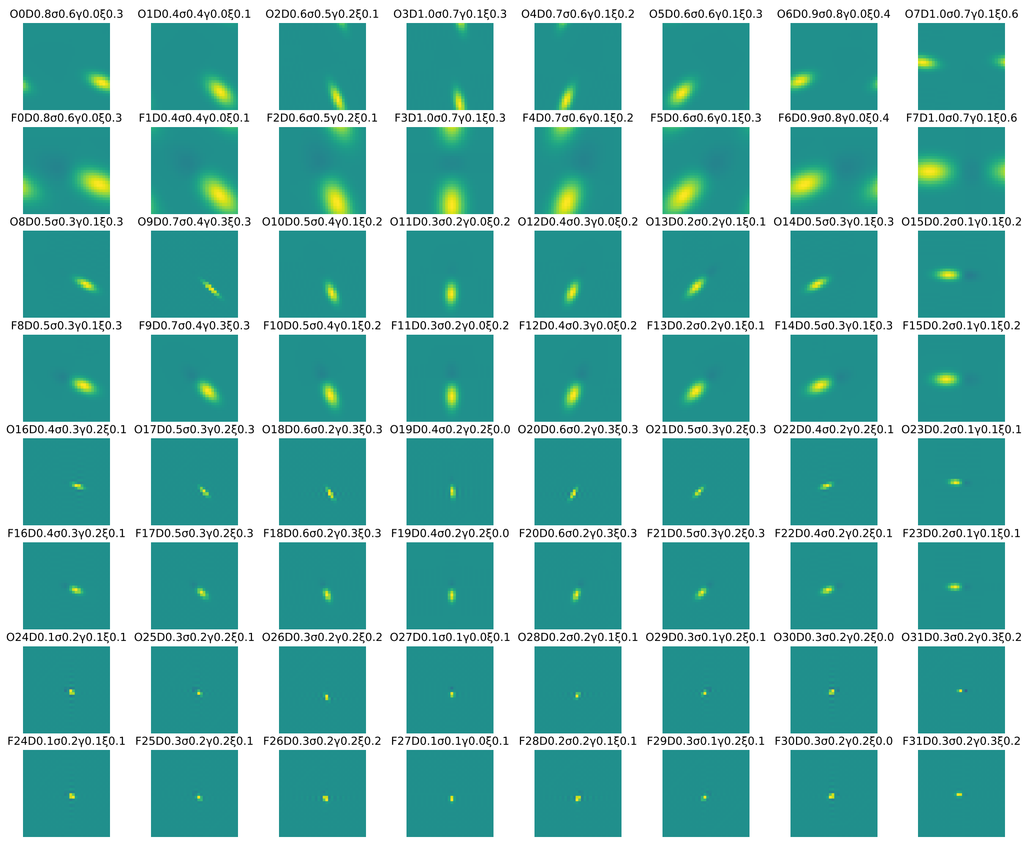

Figure 7. Filters trained on 1188 samples of COVIDx-CRX2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the 'closest' (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2.

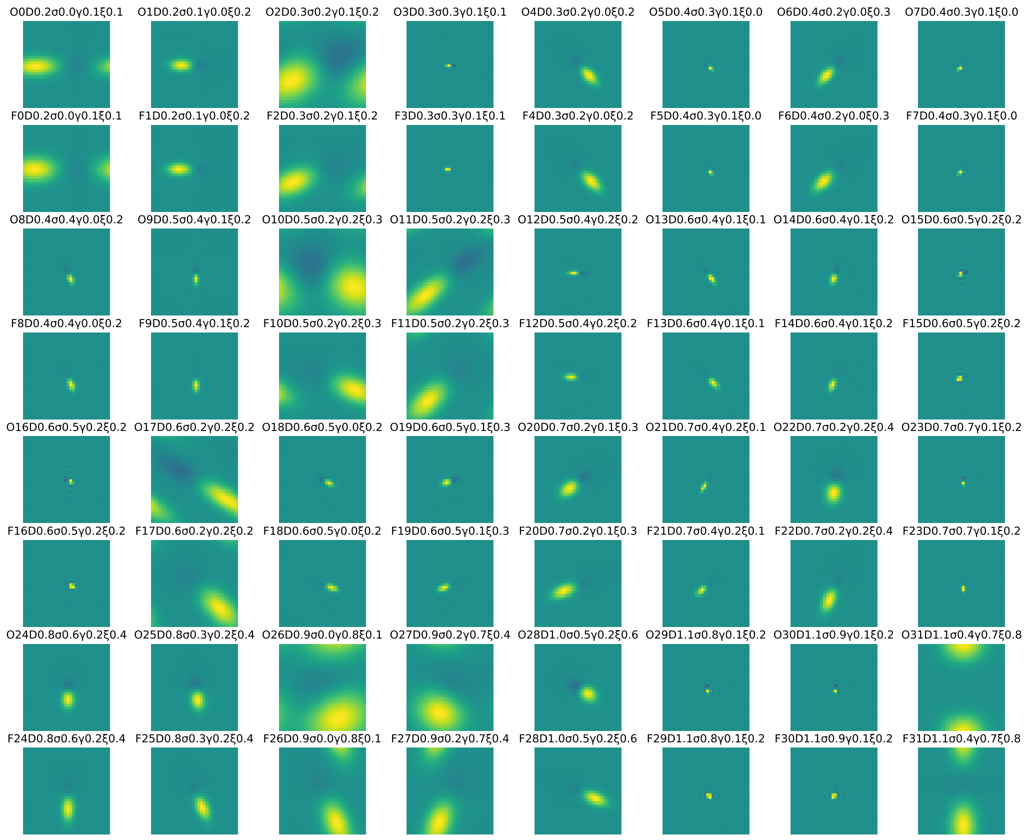

Figure 8. Filters trained on 1188 samples of COVIDx-CRX2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the 'closest' (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row 2. The filters are displayed in increasing order of their distances. The top left corner corresponds to the filters that changed the least from their initialization, while the filters in the bottom right corner changed the most.

I.2. KTH-TIPS2

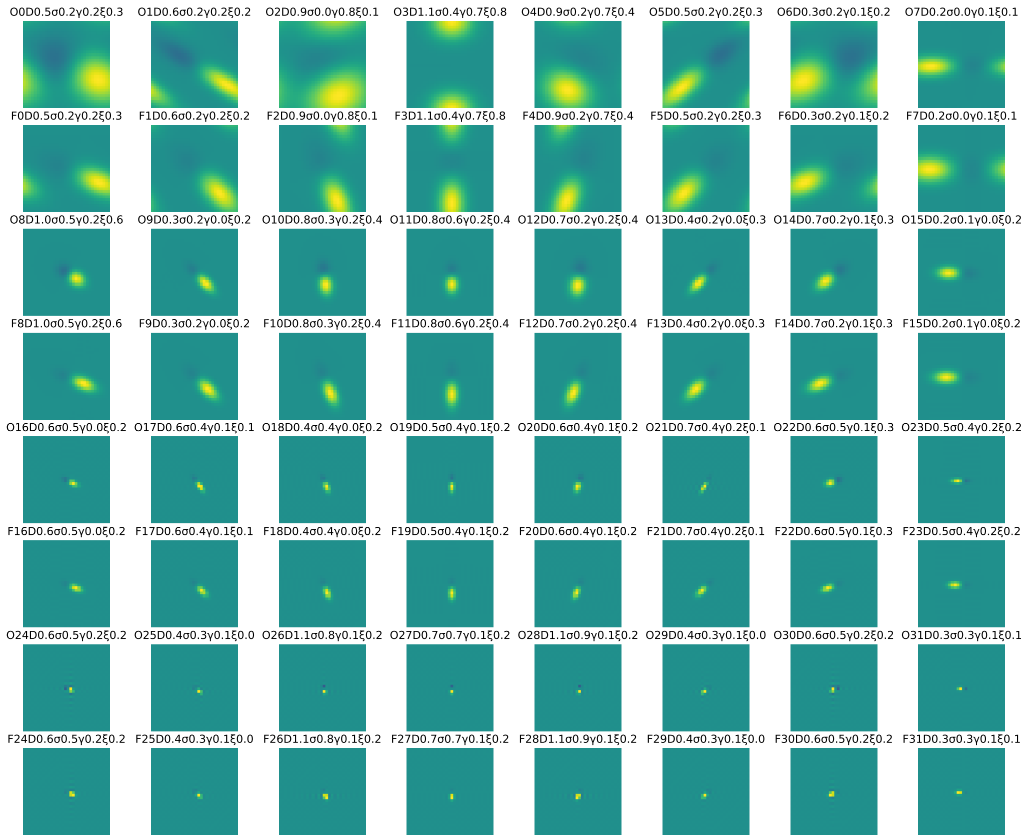

Figure 9. Filters trained on 1188 samples of KTH-TIPS2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the 'closest' (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row two.

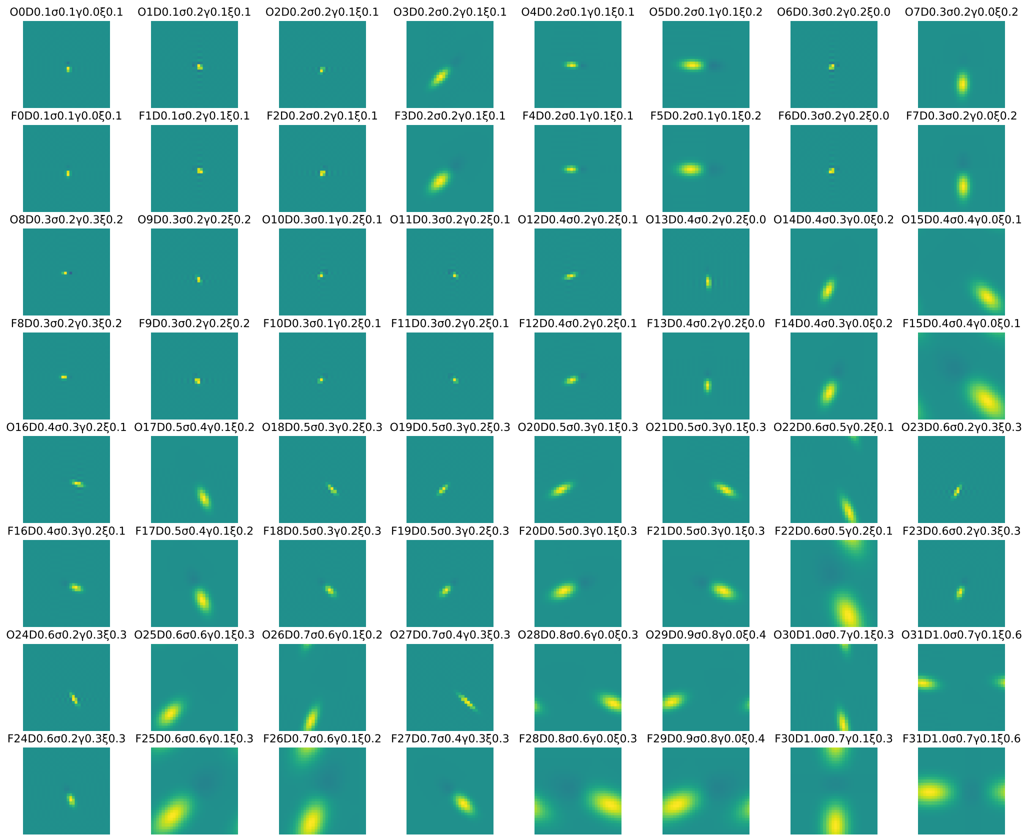

Figure 10. Filters trained on 1188 samples of KTH-TIPS2 for 500 epochs, the first, third, fifth, and seventh rows correspond to filters optimized from a tight-frame, while the second, fourth, sixth, and eighth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the 'closest' (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row two. The filters are displayed in increasing order of their distances. The top left corner corresponds to the filters that changed the least from their initialization, while the filters in the bottom right corner changed the most.

I.3. CIFAR-10

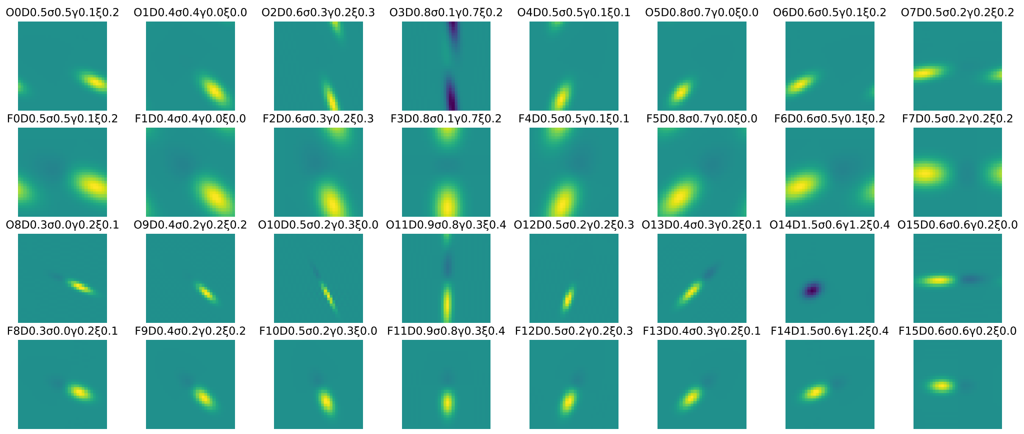

Figure 11. Filters trained on 1190 samples of CIFAR-10 for 500 epochs, the first and third, rows correspond to filters optimized from a tight-frame, while the second and fourth rows correspond to tight-frame initialized filters. The filters are displayed in pairs correspond to the 'closest' (by our distance metric defined above) filters of both types. For instance, the first filter of row one matches the first filter of row two.

I.4. Dataset Specific initializations with a Random Initialization

In Figure 12, we show how the filters adapt when initialization begins from a random setting. We note the deviation to tight frame is much greater than in the case where we initialize in a tight frame. However, as per our filterbank distance, we observe the filters do move closer to the tight frame than their initialization.

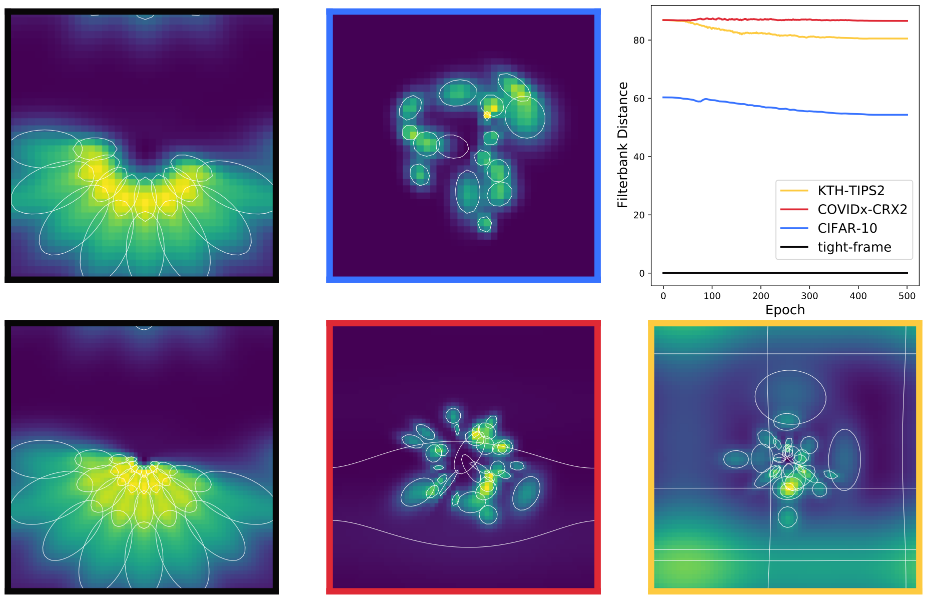

Figure 12. The graph shows the *filterbank distance* over epochs as the filters, initialized from random initialization, are trained on different datasets. To the left, we visualize dataset specific parameterizations of scattering filterbanks in Fourier space. The graph on the right shows that the randomly initialized filterbanks become more similar to a tight frame during training.

J. Number of Filters Ablation

In this section, we investigate the effect of modifying the number of filters (L) per spatial scale on CIFAR-10. We use the same settings and hyperparameters as described in Appendix A. So far, in all experiments, we have set the number of filters per scale at 8 for fair comparison with parameters.

For this ablation, we train a parametric scattering network followed by a linear layer where the wavelets are initialized using the tight frame construction and the canonical parameterization. The spatial scale is set to 2. We do not use autoaugment on the training set since, as shown in Appendix E, autoaugment can be harmful when the scattering network is followed by a linear layer. Table 10 shows the accuracy of the full training set for different values of L. We observe that the performance increases when the number of filters per scale also increases. Around 14-16 filters per spatial scale, the performance seems to have stopped increasing.

Table 11 demonstrates the mean accuracy over different sizes of training samples where the number of filters per scale is set to 8 or 16. We also consider the fixed version of the scattering transform. Over all training sample sizes, we observe that the highest performance is obtained with learnable scattering using 16 filters per spatial scale. When the scattering network is fixed, we observe that performances are lower with 16 filters instead of 8. It seems that increasing the number of filters per scale is only beneficial in the learned version of the scattering network.

Table 10. CIFAR-10 accuracy of learnable scattering followed by a linear layer (LS + LL) and multiple numbers of filters per scale (L) trained on all the training set. The wavelet filters are initialized using the tight frame construction and the canonical parameterization. The spatial scale is set to 2. No autoaugment is used for this experiment. We observe that the performance increases when the number of filters per scale (L) also increases. Around 14 filters per spatial scale, the performance seems to have stopped increasing.

L	All Data
2	63.59
4	70.94
6	74.03
8	74.94
10	76.40
12	77.01
14	**77.36**
16	77.33

Table 11. CIFAR-10 mean accuracy and std. error over 10 seeds with multiple training sample sizes and different values of L. The wavelet filters are initialized using the tight frame construction. The spatial scale is set to 2. No autoaugment is used for this experiment. Over all training sample sizes, we observe that the highest performance is obtained with learnable scattering using 16 filters per spatial scale.

Arch.	Parameterization	L	100 samples	500 samples	1000 samples	All
LS+LL†	Canonical	8	39.70 ± 0.62	50.74 ± 0.30	54.76 ± 0.22	74.94
LS+LL†	Canonical	16	39.73 ± 0.39	$\mathbf{54.17} \pm 0.36$	$\mathbf{58.36} \pm 0.29$	**77.33**
S +LL	-	8	37.55 ± 0.62	49.67 ± 0.33	53.96 ± 0.48	70.71
S +LL	-	16	35.85 ± 0.48	48.2 ± 0.27	52.74 ± 0.25	70.64

K. Perturbing a Converged Parameterization Ablation

The training accuracies and losses over epochs are shown in Figure 13 for LS+L and S+L. We observe that both networks are following similar patterns. Local minima are expected here (see results with different init Figure 1), as in any non-convex problem. To showcase learnable scattering's stability to stochastic optimization, we perform an ablation below (Figures 14 & 15) for trained LS+L, where each parameter is perturbed and then individually optimized. We find that they all return rapidly to the (local) optimum.

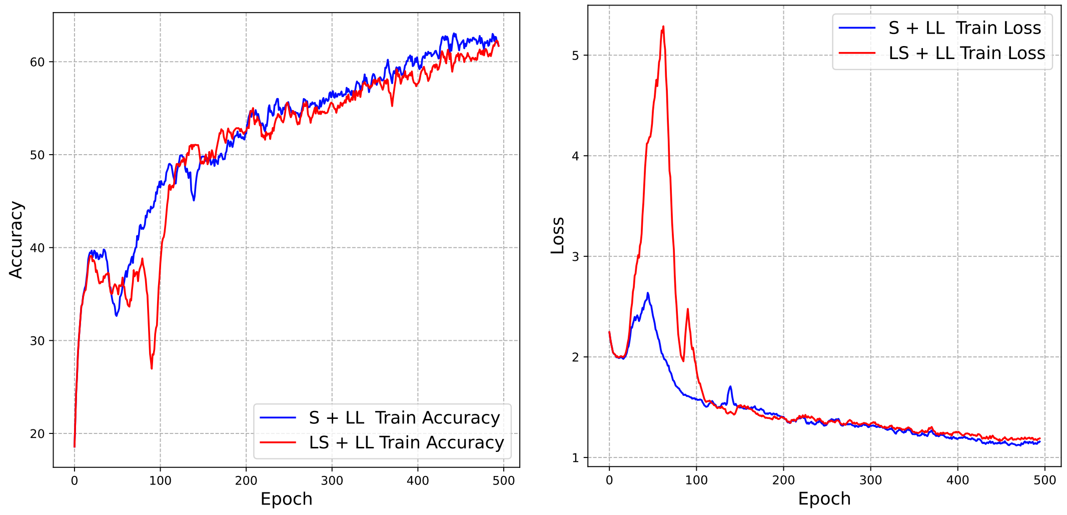

Figure 13. Plots comparing the training accuracies and losses over epochs of LS+LL and S+LL. Note that scattering parameters are only optimized for LS+LL. The networks were trained for 500 epochs on 1000 samples of CIFAR-10. (Left) Accuracy: LS+LL v.s. S+LL. (Right) Loss: LS+LL v.s. S+LL.

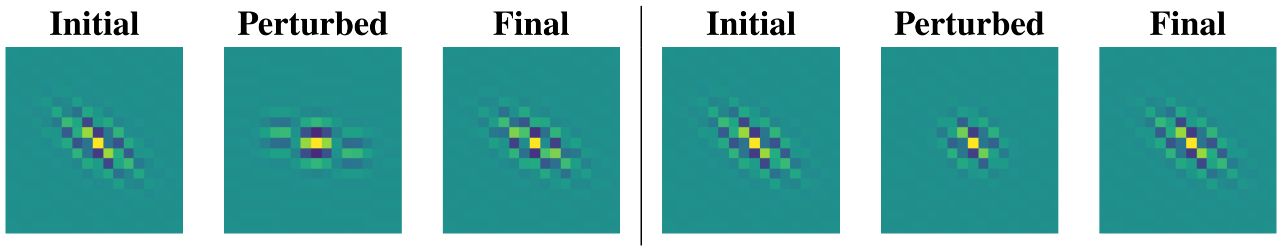

Figure 14. (left) θ parameter of a learned parameterization is perturbed and returns to an almost identical orientation after gradient based optimization. (right) ξ parameter of a learned parameterization is perturbed and returns to an almost identical frequency scale after gradient based optimization.

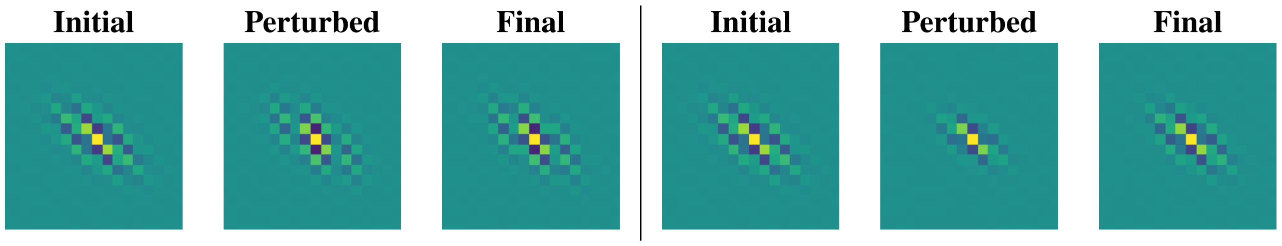

Figure 15. (left) σ parameter of a learned parameterization is perturbed and returns to an almost identical gaussian window scale after gradient based optimization. (right) γ parameter of a learned parameterization is perturbed and returns to an almost identical aspect ratio after gradient based optimization.

L. Visualizing Parameter Values over Time

Figure 17 shows, on the left, a Morlet wavelet filter before training and, in the middle, the same wavelet filter after training. The parametric scattering network was trained for 1K epochs on 1000 samples of CIFAR-10. We use the Morlet canonical parameterization. We observe that the global orientation has changed during training. Figure 17 (right) shows the values of the parameters from Table 1 over the training steps. We observe that the value of all parameters changed during training. However, the aspect ratio seems to have returned to its initial value. In Figure 17 (right), we also show the value of $\xi \times \sigma$, which is a proxy for the wavelet scale factor, over the steps. The value of $\xi \times \sigma$ is smaller after training. This can be observed in Figure 17 (left-middle), where the wavelet after training seems to be more compressed than before training.

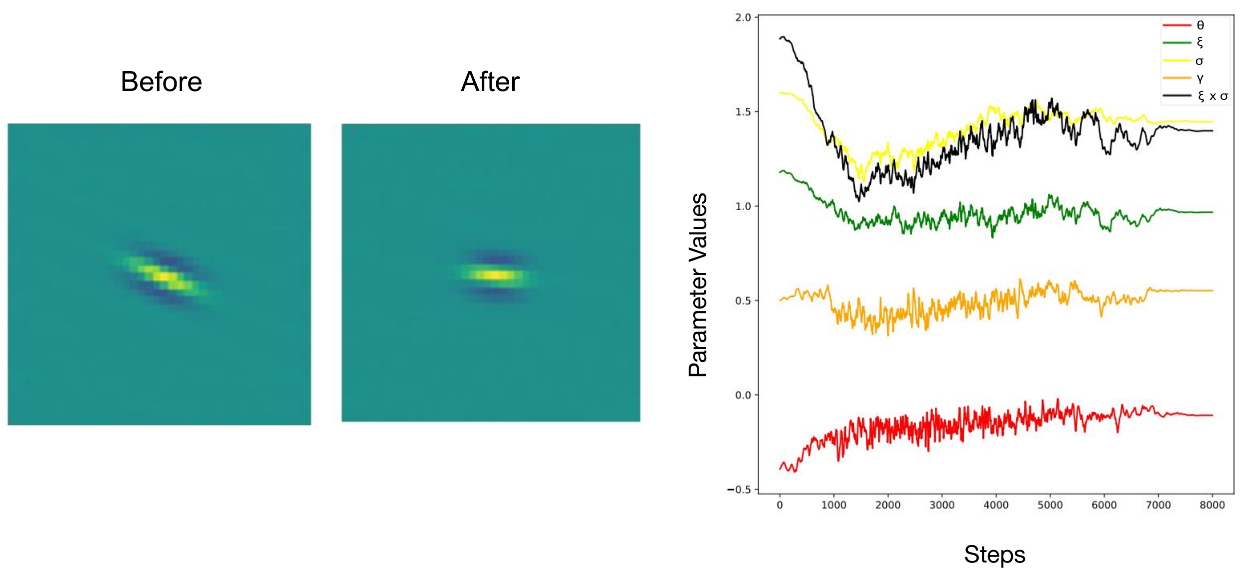

Figure 17. Parameter values over time. Real part of Morlet wavelet filters initialized with *tight-frame* schemes before (left) and after (middle) training. The network was trained for 1K epochs on 1000 samples of CIFAR-10. We use the Morlet canonical wavelet parameterization. The plot (right) shows the parameter values over epochs.