As large as it gets – Studying Infinitely Large Convolutions via Neural Implicit Frequency Filter

Supplementary Material

In the following, we provide additional information and details that accompany the main paper:

A Kernel Mass Evaluation

In this section, we evaluate the kernel mass ratio for more ResNet models trained on ImageNet-1k (Figure A1) and different network architectures trained on ImageNet-100 (Figure A2). The networks show similar behavior already observed in the main paper, all models predominately learn small, well-localized kernels regardless of the potential to learn much larger kernels. However, the smaller ResNet-18 model learns larger kernels than the ResNet-50 or ResNet-101 in the second layer. For ImageNet-100, MobileNet-v2 does not learn as large kernels as observed for ImageNet-1k. Further, ResNet-50 trained on ImageNet-100 seems to learn larger kernels in the second layers compared to the ResNet-50 trained on ImageNet-1k (Figure 4).

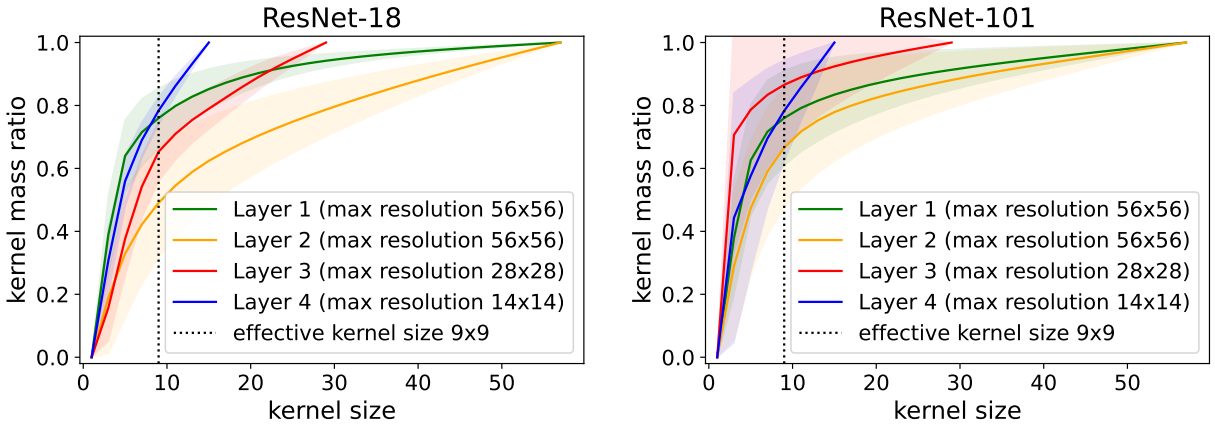

Figure A1: Effective kernel size evaluation on ImageNet-1k for further ResNet models. We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. The layers are summarised as follows: Layer 1 encoding 56×56, Layer 2 28×28, Layer 3 14×14 and Layer 4 7×7.

Non-square kernels Following Romero et al. (2022a), we analyse not only square shape kernels, which are typically applied in CNNs but also rectangular shape kernels. Thus, we fit a 2D Gaussian to kernels learned using our NIFF and compare the variance given by σ_x and σ_y in the x- and y- direction. In detail, we build the ratio between σ_x and σ_y of the Gaussian. To aggregate over all kernels within one layer, we plot the mean and standard deviation of these ratios in Figure A3 and Figure A4. The mean over all kernels within a layer is near to a ratio of one indicating that most kernels exhibit square-shapes. For some layers (the last layer for ResNet-50 and ResNet-101 on ImageNet-1k and ResNet-18 and ResNet-50 on ImageNet-100) the variance is quite high, indicating that σ_x and σ_y differ and non-square, rectangle kernels are learned. Not that the learned kernels by Romero et al. (2022a) are parameterized by a Siren Sitzmann et al. (2020) leading to more wave-like, smooth kernels. In contrast, we learn the kernels in the frequency domain which could be wave-like, but are mostly not wave-like as shown in Figure A22 and Figure A23. Therefore, the measured standard deviations in x and y direction should not be understood as a kernel mask as argued in Romero et al. (2022a). They merely indicate the rough spatial distribution of filter weights.

B Filter Visualization

B.1 Principle Component Analysis

The Principle Component Analysis, short PCA, is typically a dimensionality reduction method. The goal of PCA is to maintain most information of a dataset while transforming it into a smaller one. The first principle component explains the most variance of the data, thus representing the majority. The second principle component explains the next highest variance while being orthogonal to the first. For more details on PCA, we refer to Dunteman (1989). For our analysis, we use the PCA of the learned kernels to visualize the predominate structure. Hence, we use the dimensionality reduction property of the PCA to simplify the visualization of the kernels. We also provide images of the original kernels in Figure A22 and A23.

B.2 Spatial Kernels

In this section, we show the PCA evaluation of the learned spatial kernels by the NIFFs for additional architectures and datasets (ImageNet-100). The results are similar to the ones in the paper (Figure 6, Figure 1 and Figure 7. The learned filters in the spatial domain are well-localized and relatively small compared to the size they could actually learn. This holds true for different architectures on ImageNet-1k (Figure A5, A6 and A7) as well as for ImageNet-100 (Figure A9, A10 and A12).

Further, we show a grid of the actually learned filter in the spatial domain in Figure A22 and A23.

The learned spatial filters on CIFAR-10 are shown in Figure A13. Similarly, to the results shown in Figure 7. The network learns well-localized, small filters n the spatial domain. Yet, for the feature map size of 8×8 in Layer 2 the network uses significantly more than the standard kernel size of 3×3.

B.3 NIFF multiplication weights

Moreover, we analyze the learned element-wise multiplication weights for the real and imaginary parts of different models trained on ImageNet-1k in the frequency domain. Figures A14, A15, A16 and A17 show the PCA per layer for the learned element-wise multiplication weights for ResNet-50, DenseNet-121, ConvNeXt-tiny and MobileNet-V2 respectively. For ResNet-50 and ConvNext-tiny, it seems as if the networks focus in the first layer on the middle-frequency spectra and in the later layers more on the high-frequency spectra. The multiplication weights learned for MobileNet-V2 (Figure A16) focus in the first layer on low-frequency information in the second layer on high-frequency information and in the third layer again on low-frequency information. The DenseNet-121 (Figure A17) learns high-frequency information prior in the first two layers and low-frequency information predominately in the later, third layer. Hence, a general claim for different models and their learned multiplication weights in the frequency domain can not be derived from our empirical analysis. Still for all networks, the imaginary part seems to be less important for these networks and thus the learned structures are less complex. This might be owed to the fact that with increased sparsity through the activation function in the network, the network favours cosine structures (structures with a peak in the center) over sine structures.

C Performance Evaluation

ImageNet-100 We report the accuracy our NIFF CNNs could achieve on ImageNet-100 and the number of learnable parameters in Table A1. The trend is similar to ImageNet-1k, the larger models benefit from NIFF while the lightweight models do not so much. In addition to the models considered in the main paper, we additionally evaluated MobileNet-v3 Howard et al. (2019). We observe that both the baseline model and the NIFF version have comparably low accuracy. For MobileNets, the training pipeline is usually highly optimized for best performance. The data augmentation scheme from Liu et al. (2022b), that we employ for all trainings to achieve comparable results, does not seem to have a beneficial effect here, neither on ImageNet-1k (for Sandler et al. (2018), Table 2) nor on ImageNet-100 (for Sandler et al. (2018); Howard et al. (2019), Table A1).

Table A1: Performance evaluation of top 1 and 5 test accuracy on ImageNet-100 for different network architectures. We used the standard training parameter for each network architecture and stayed consistent with these for each architecture respectively. For the bigger networks like ResNet and DenseNet, which include 2D convolutions, we split into depth-wise and 1×1 convolution as described in Section 3.2 to reduce the number of parameters, for faster training. For all models, our NIFF CNNs perform slightly better, even with reduced number of parameters.

Name	# Parameters	Acc@1	Acc@5
ConvNeXt-tiny Liu et al. (2022b)	27.897.028	91.70	98.32
NIFF (ours)	28.231.684	**92.00**	**98.42**
ResNet-18 He et al. (2016)	11.227.812	**87.52**	**97.50**
NIFF 2D conv (ours)	20.665.620	86.42	97.08
ResNet-18 separated conv	2.865.700	86.52	97.20
NIFF (ours)	3.198.660	86.52	97.14
ResNet-50 He et al. (2016)	23.712.932	89.88	98.22
NIFF 2D conv (ours)	31.934.228	**90.08**	98.06
ResNet-50 separated conv	16.431.588	89.78	98.18
NIFF (ours)	16.760.900	89.98	**98.44**
ResNet-101 He et al. (2016)	42.705.060	**90.54**	98.14
NIFF 2D conv (ours)	60.954.180	89.94	98.14
ResNet-101 separated conv	26.549.988	90.20	98.36
NIFF (ours)	27.343.332	**90.54**	**98.38**
DenseNet-121 Huang et al. (2017)	7.056.356	90.06	98.20
NIFF 2D conv (ours)	9.197.252	89.66	**98.32**
DesNet-121 separated conv	5.222.628	89.94	98.08
NIFF (ours)	5.237.012	**90.24**	98.18
MobileNet-v2 Sandler et al. (2018)	2.351.972	84.06	96.52
NIFF (ours)	2.359.660	**85.46**	**96.70**
MobileNet-v3 Howard et al. (2019)	1.620.356	79.84	94.76
NIFF (ours)	1.617.748	78.90	95.40

CIFAR-10 Although NIFF CNNs can perform on par with the respective baseline on high-resolution datasets, their performance is limited on low-resolution dataset. Table A2 shows the results on CIFAR-10 with different architectures. Unfortunately, our NIFF CNNs lose around 1 to 3 % points compared to the baseline models. This can be addressed to our previous observation: The networks trained on CIFAR-10 do only use a small amount of the potential kernel size NIFF provides being as big as the kernels of the baseline model (3×3).

D Circular vs Linear Convolution

Our NIFF as proposed above performs a circular convolution, which allows us to directly apply the convolution theorem and execute it as multiplication in the frequency domain. However, standard convolutions in CNNs are finite linear convolutions. A circular convolution can mimic a linear convolution when zero-padding a signal with length M and a kernel with length K to length $L \leq M + K - 1$ Winograd (1978). Thus, to ablate on circular versus finite linear convolutions, the input featuremaps with size $N \times N$ are

Table A2: Performance evaluation of different networks trained on CIFAR-10 and the number of learnable hyperparameters for each network. To be comparable between all models and architecture changes we used the same training schedule for all of them. One can see that NIFF CNNs perform slightly better with a ConvNeXt Liu et al. (2022b) backbone. However, for other architectures, it performs slightly worse.

Method	# Parameters	Top 1 Acc
ConvNeXt-tiny Liu et al. (2022b)	6.376.466	90.37
NIFF (ours)	6.305.746	**91.48**
ResNet-18 He et al. (2016)	11.173.962	**92.74**
NIFF 2D conv (ours)	20.613.546	92.66
ResNet-18 separated conv	2.810.341	90.18
NIFF (ours)	1.932.432	90.63
ResNet-50 He et al. (2016)	23.520.842	**93.75**
NIFF 2D conv (ours)	31.743.914	93.39
ResNet-50 separated conv	16.237.989	92.13
NIFF (ours)	15.491.600	93.11
DenseNet-121 Huang et al. (2017)	6.956.426	**93.93**
NIFF 2D conv (ours)	9.099.098	92.47
DenseNet-121 separated conv	5.121.189	92.00
NIFF (ours)	5.555.856	92.49
MobileNet-v2 Sandler et al. (2018)	2.236.682	**94.51**
NIFF (ours)	2.593.760	94.03
MobileNet-v3 Howard et al. (2019)	1.528.106	86.28
NIFF (ours)	1.526.466	**86.60**

zero-padded to $2N \times 2N$. For both, the linear and the circular case, NIFF learns filters with the original size $N \times N$ of the featuremap. To mimic linear filters, the learned filters by our NIFF are transformed into the spatial domain and zero-padded similarly to the input featuremaps to $2N \times 2N$. Afterwards, they are transformed back into the frequency domain and the point-wise multiplication is executed. Note that this is not efficient and just serves the academic purpose of verifying whether any accuracy is lost when replacing linear convolutions by circular ones in our approach. However, the resulting networks experience a performance drop compared to the baseline and our NIFF as shown in Table A3 and Table A4. We hypothesize that this drop in performance results from the enforcement of really large kernels. The additional padding mimics linear finite convolutions that are as big as the featuremaps. Related work has shown that larger context can improve model performance Ding et al. (2022); Liu et al. (2022a). Still, there is a limit to which extent this holds as with large kernels artifacts may arise Tomen & van Gemert (2021). Thus, enforcing kernels as large as the featuremaps seems to be not beneficial as shown by our quantitative results. Another explanation for the drop in accuracy could be the introduction of sinc interpolation artifacts into the padded and transformed featuremaps and kernels. The padding is formally a point-wise multiplication with a box-function in the spatial domain. Thus, sinc-interpolation artifacts in the frequency domain can arise. Figure A20 and Figure A21 show that the learned spatial kernels are larger than the learned kernel when we apply a circular convolution with our NIFF. While the first and second layers still learn relatively small filters compared to the actual size they could learn, the third and fourth layers make use of the larger kernels. Since these results come with a significant drop in accuracy, we should however be careful when interpreting these results.

Table A3: Evaluation of top 1 and top 5 accuracies on ImageNet-100 for networks learning filters with our NIFF but afterwards those are padded in the spatial domain and transformed back into the frequency domain to mimic linear convolutions. All ResNet architectures and DenseNet-121 were trained with separated depth-wise and 1x1 convolutions due to efficiency reasons. All models using finite linear convolutions perform significantly worse than the baseline and our NIFF which applies circular convolutions. This observation is consistent with low-resolution data like CIFAR-10.

Name	Acc@1	Acc@5
ConvNeXt-tiny Liu et al. (2022b)	91.70	98.32
NIFF (ours)	**92.00**	**98.42**
NIFF linear	83.00	95.36
ResNet-18 separated conv	86.52	97.20
NIFF (ours)	86.52	97.14
NIFF linear	81.90	95.42
ResNet-50 separated conv	89.78	98.18
NIFF (ours)	89.98	98.44
NIFF linear	86.76	97.16
ResNet-101 separated conv	90.20	98.36
NIFF (ours)	90.54	**98.38**
NIFF linear	86.70	97.06
DesNet-121 separated conv	89.94	98.08
NIFF (ours)	**90.24**	98.18
NIFF linear	81.40	95.52
MobileNet-v2 Sandler et al. (2018)	84.06	96.52
NIFF (ours)	**85.46**	**96.70**
NIFF linear	73.90	93.16

E Runtime

As discussed in the computing costs section of the main paper, our approach is slower than the current implementation with spatial convolutions due to the repetitive use of FFT and IFFT. However, when comparing the number of FLOPs needed to compute convolutions with kernel sizes as big as the featuremaps to our NIFF approach, NIFF requires significantly fewer FLOPs, especially with increased featuremap size. Figure 8 shows that most of the FLOPs for our NIFF result from the additional FFT and IFFT operation. Still, we require much fewer FLOPs than large spatial convolutions.

Moreover, we evaluate the runtime per epoch for each model on CIFAR-10 (Table A5) and ImageNet-100 (Table A6) and compare it to the standard spatial 3×3 convolution, which has a much smaller spatial context than our NIFF as well as spatial convolutions which are as large as the featuremaps. This would be comparable to our NIFF. Obviously, small spatial kernels (3×3) are much faster than larger kernels like NIFF or large spatial kernels. However, NIFF is much faster than the large spatial kernels during training. Especially on high-resolution datasets like ImageNet-100 our NIFF is over four times faster on ResNet-50 and over three times faster on ConvNeXt-tiny compared to the large convolution in the spatial domain.

In general, we want to emphasize that our NIFF models still learn infinite large kernels while all kernels in the spatial domain are limited to the set kernel size. If one would like to learn a 2D convolution in the spatial domain with an image g of size $N \times N$ and filters with the same size $N \times N$ this would be in $O(N^4)$ whereas using FFT and pointwise multiplication (Equation 1) would result in $O(N^2 \log(N))$.

Table A4: Evaluation of top 1 on CIFAR-10 for networks learning filters with our NIFF but afterwards those are padded in the spatial domain and transformed back into the frequency domain to mimic linear, non-circular convolutions. All models using finite linear convolutions perform significantly worse than the baseline and our NIFF. This observation is consistent with high-resolution data like ImageNet-100.

Method	Top 1 Acc
ConvNeXt-tiny Liu et al. (2022b)	90.37
NIFF (ours)	**91.48**
NIFF linear	84.30
ResNet-18 He et al. (2016)	**92.74**
NIFF full (ours)	92.66
NIFF full linear	85.10
ResNet-18 separated conv	90.18
NIFF (ours)	90.63
NIFF linear	83.11
ResNet-50 He et al. (2016)	**93.75**
NIFF full (ours)	93.39
NIFF full linear	88.22
ResNet-50 separated conv	92.13
NIFF (ours)	93.11
NIFF linear	87.54
DenseNet-121 Huang et al. (2017)	**93.93**
NIFF full (ours)	92.47
NIFF full linear	85.73
DenseNet-121 separated conv	92.00
NIFF (ours)	92.49
NIFF linear	79.95
MobileNet-v2 Sandler et al. (2018)	**94.51**
NIFF (ours)	94.03
NIFF linear	93.21

Table A5: Average training time per epoch in seconds and standard deviation on one NVIDIA Titan V of NIFF compared to standard spatial convolutions 3x3 or 7x7 and maximal larger spatial convolutions on CIFAR-10.

Name	Baseline 3x3/7x7	Spatial Conv featuremap sized	NIFF (ours)
ConvNeXt-tiny Liu et al. (2022b)	68.31 ± 0.62	108.97 ± 0.25	96.48 ± 1.97
ResNet-18 He et al. (2016)	8.57 ± 0.20	22.75 ± 0.28	17.87 ± 0.54
ResNet-50 He et al. (2016)	7.98 ± 0.10	37.05 ± 0.48	27.36 ±0.16
DenseNet-121 Huang et al. (2017)	26.36 ± 1.32	67.11 ± 0.25	84.51 ± 3.71
MobileNet-v2 Sandler et al. (2018)	22.50 ± 0.26	143.47 ± 0.20	83.41 ± 4.78

F Ablation on more modules

We show that our NIFF can be combined with other frequency modules to achieve networks that operate mostly in the frequency domain. Hence, the number of transformations can be reduced. Table A9 demon-

Table A6: Average training time per epoch in seconds and standard deviation on four NVIDIA A100 of NIFF compared to standard spatial convolutions (3x3 or 7x7) and maximal larger spatial convolutions on ImageNet-100.

Name	Baseline 3x3/7x7	Spatial Conv featuremap sized	NIFF (ours)
ConvNeXt-tiny Liu et al. (2022b)	92.19 ± 4.21	487.89 ± 3.54	149.70 ± 1.32
ResNet-18 He et al. (2016)	31.99 ± 0.71	608.13 ± 22.53	89.00 ± 0.60
ResNet-50 He et al. (2016)	85.51 ± 0.22	951.43 ± 6.09	204.82 ± 0.33
ResNet-101 He et al. (2016)	152.39 ± 5.07	1392.21 ± 47.91	349.83 ± 2.44
DenseNet-121 Huang et al. (2017)	128.95 ± 1.64	10188.15 ± 28.17	408.08 ± 2.20
MobileNet-v2 Sandler et al. (2018)	32.64 ± 0.25	856.84 ± 4.35	100.75 ± 0.15

Table A7: Average inference time in seconds and standard deviation on one NVIDIA Titan V of NIFF compared to standard spatial convolutions (3x3 or 7x7) and maximal larger spatial convolutions on the full CIFAR-10 validation set.

Name	Baseline 3x3/7x7	Spatial Conv featuremap sized	NIFF (ours)
ConvNeXt-tiny Liu et al. (2022b)	2.35 ± 0.05	4.06 ± 0.02	6.32 ± 0.14
ResNet-18 He et al. (2016)	2.69 ± 0.10	3.29 ± 0.08	3.14 ± 0.54
ResNet-50 He et al. (2016)	3.09 ± 0.13	5.05 ± 0.45	5.26 ± 0.06
DenseNet-121 Huang et al. (2017)	3.40 ± 0.05	5.27 ± 0.02	6.50 ± 0.14
MobileNet-v2 Sandler et al. (2018)	2.93 ± 0.06	8.68 ± 0.43	7.02 ± 0.04

strates that adding the downsampling layer Grabinski et al. (2022; 2023) or the last average pooling and the fully connected layer also yields good results. Also combining all of them, NIFF, FLC Pooling Grabinski et al. (2022) and the Average Pooling plus Fully connected layer performs quite well. Also, incorporating the ComplexBatchNorm Trabelsi et al. (2018) leads to a drop in accuracy by roughly 15%. We also tried to incorporate the non-linearity into the frequency domain, but we were not able to achieve much better results than by removing it fully.

G Ablation on Padding

We evaluate our NIFF when the featuremaps are padded with different kinds of padding methods. Due to the padding of the featuremaps and the cropping after the application of our NIFF, possible artifacts can be mitigated. The padding is applied around the featuremaps before transforming them into the frequency domain. The padding size is as large as the original featuremap. After the application of our NIFF, the featuremaps are transformed back into the spatial domain and cropped to their original size. The resulting networks experience similar performance as our baseline NIFF as shown in Table A10. Figure A18 and Figure A19 show the learned spatial kernels when only the featuremaps are padded. The learned spatial kernels are still relatively small and well-localized.

H NIFF's architecture

In the following, we describe the architecture used for our NIFFs for each backbone network architecture. Note that the size of the NIFF is adjusted to the size of the baseline network as well as the complexity of the classification task.

Table A8: Average inference time in seconds and standard deviation on four NVIDIA A100 of NIFF compared to standard 3x3 full or 7x7 depth-wise spatial convolutions and maximal larger spatial convolutions on the full ImageNet-100 validation set.

Name	Baseline 3x3/7x7	Spatial Conv featuremap sized	NIFF (ours)
ConvNeXt-tiny Liu et al. (2022b)	4.86 ± 0.13	17.07 ± 0.19	6.06 ± 0.08
ResNet-18 He et al. (2016)	4.53 ± 0.52	24.57 ± 0.21	4.52 ± 0.18
ResNet-50 He et al. (2016)	5.43 ± 0.30	42.33 ± 0.21	10.35 ± 0.14
ResNet-101 He et al. (2016)	7.35 ± 0.17	54.85 ± 0.25	16.82 ± 0.10
DenseNet-121 Huang et al. (2017)	5.12 ± 0.07	104.36 ± 0.40	12.04 ± 0.18
MobileNet-v2 Sandler et al. (2018)	4.27 ± 0.13	28.97 ± 0.04	6.61 0.16

Table A9: Comparison on CIFAR-10 of NIFF MobileNet-v2 Sandler et al. (2018) incorporating more modules besides our NIFF into the frequency domain.

NIFF	FLC Pooling [2022]	Complex BatchNorm [2018]	AveragePooling + FC	Accuracy
✓	✗	✗	✗	**94.03**
✓	✓	✗	✗	93.61
✓	✗	✓	✗	78.60
✓	✗	✗	✓	93.82
✓	✓	✓	✗	79.83
✓	✓	✗	✓	93.27
✓	✗	✓	✓	73.90
✓	✓	✓	✓	55.81

Low-resolution task All networks trained on CIFAR-10 incorporate the same NIFF architecture. The NIFF consists of two stacked 1×1 convolutions with a ReLU activation function in between. The 1×1 convolution receives as input two channels, which encode the x and y coordinate as described in Figure 2. The 1×1 convolution expands these two channels to 32 channels. From these 32 channels, the next 1×1 convolution maps the 32 channels to the desired number of point-wise multiplication weights.

High-resolution task For the networks trained on ImageNet-100 and ImageNet-1k the size of the neural implicit function to predict the NIFF is kept the same for each architecture respectively, while the size of the neural implicit function is adjusted to the network architecture to achieve approximately the same number of trainable parameters. Hence, the lightweight MobileNet-v2 model Sandler et al. (2018) and the small DensNet-121 Huang et al. (2017) incorporate a smaller light-weight neural implicit function to predict the NIFF, while larger models like ResNet He et al. (2016) or ConvNeXt-tiny Liu et al. (2022b) incorporate a larger neural implicit function. For simplicity, we define two NIFF architectures. One for the large models and one for the smaller, lightweight models.

For the smaller, lightweight models, the neural implicit function consists of three stacked 1×1 convolutions with one SiLU activation after the first one and one after the second one. The dimensions for the three 1×1 convolutions are as follows. We start with two channels and expand to eight channels. From these eight channels, the second 1×1 convolution suppresses the channels down to four. Afterwards, the last 1×1 convolution maps these four channels to the desired number of point-wise multiplication weights.

For the larger models, we used four layers within the neural implicit function for NIFF. The structure is similar to all NIFFs between each 1×1 convolution a SiLU activation function is applied. The dimensions for the four layers are as follows. First from two to 16 channels, secondly from 16 to 128 channels and

Table A10: Evaluation of top 1 and top 5 accuracies on ImageNet-100 for ResNet-18 with different kinds of padding.

Name	Acc@1	Acc@5
ResNet-18 baseline	87.52	97.50
NIFF (ours)	86.52	97.14
NIFF zero padding	87.00	97.54
NIFF reflect padding	86.64	97.24
NIFF circular padding	87.06	97.34

afterwards suppressed down from 128 to 32 channels. The last 1×1 convolution maps these 32 channels to the desired number of point-wise multiplication weights.

We show that the smaller NIFF size for the lightweight models does not influence the resulting performance. Thus, we train a lightweight MobileNet-v2 with larger NIFFs (similar size as the larger models). The results are presented in Table A11. We can see that the network does not benefit from the larger NIFF size. Hence, we assume that keeping the smaller NIFFs for the smaller, lightweight models can achieve a good trade-off between the number of learnable parameters and performance.

Table A11: Evaluation of top 1 and top 5 accuracies on ImageNet-100 for different NIFF sizes for the lightweight MobileNet-v2 Sandler et al. (2018).

Name	Acc@1	Acc@5
MobileNet-v2 baseline	**84.94**	96.28
small NIFF	83.72	**96.40**
big NIFF	83.82	96.32

Low-resolution task For all models trained on CIFAR-10 the NIFF architectures is kept the same. The neural implicit function consists of two stacked 1×1 convolutions with one ReLU activation in between. The dimensions for the two 1×1 convolutions are as follows. We start with two channels and expand to 32 channels. The second 1×1 convolution maps these 32 channels to the desired number of point-wise multiplication weights.

Ablation on Separated Convolution Further, we ablate our design choices to use separated depth-wise and 1×1 convolutions instead of full convolutions for efficiency. Hence, we train all ResNet and DenseNet networks with additionally separated convolutions (separated in depth-wise and 1×1 convolution) as well as our NIFF as full convolution. Tables 2, A1 and A2 show that using separated convolutions in the spatial domain performs slightly worse than the baseline but also reduces the amount of learnable parameters similarly to our NIFF. Using full convolutions in our NIFF leads to an increase in accuracy but also an increased amount of learnable parameters. Hence, we can see a clear trade-off between number of learnable parameters and accuracy.

I Training Details

ImageNet. The training parameters and data preprocessing are kept the same for ImageNet-1k and ImageNet-100. For the training of each network architecture, we used the data preprocessing as well as the general training pipeline provided by Liu et al. (2022b). The training parameters for each individual

network are taken from the original papers provided by the authors ResNet He et al. (2016), DenseNet-121 Huang et al. (2017) ConvNeXt-tiny Liu et al. (2022b) and MobileNet-v2 Sandler et al. (2018).

CIFAR-10. For CIFAR-10 we used the same training parameter for all networks. We trained each network for 150 epochs with a batch size of 256 and a cosine learning rate schedule with a learning rate of 0.02. we set the momentum to 0.9 and weight decay to 0.002. The loss is calculated via LabelSmoothingLoss with label smoothing of 0.1 and as an optimizer, we use Stochastic Gradient Descent (SGD).

For data preprocessing, we used zero padding by four and cropping back to 32×32 and horizontal flip, as well as normalizing with mean and standard deviation.

Computing Infrastructure For training our models and the baseline we use NVIDIA Titan V and NVIDIA A100 GPUs. For the training on low-resolution data (CIFAR-10). We used one NVIDIA Titan V, depending on the model architecture and the convolution used (baseline, NIFF or large convolution) the training took between 15 minutes and 90 minutes. For the training on high-resolution data (ImageNet-100 and 1k) we used four NVIDIA A100 in parallel. The training time depends on the used model architecture and varies if we used the full ImageNet-1k dataset or only ImageNet-100. The training time for ImageNet-1k varies between one day and one hour and ten days and nine hours for ImageNet-100 between 93 minutes and one day eight hours dependent on the model architecture and the number of epochs for training.

J Convolution Theorem

Following, we demonstrate the proof of the convolution theorem. For more details, please refer to (for example) Bracewell & Kahn (1966); Forsyth & Ponce (2003).

As stated in Equation 1 in the main paper we make use of the convolution theorem Bracewell & Kahn (1966); Forsyth & Ponce (2003) which states that a circular convolution, denoted by $\circledast$, between a signal $g(x)$ and filter $k(x)$ in the spatial domain can be equivalently represented by a point-wise multiplication, denoted by $\odot$, of these two signals in the frequency domain, by computing their Fourier Transform, denoted by the function $\mathcal{F}(.)$:

$$\mathcal{F}(g \circledast k) = \mathcal{F}(g) \odot \mathcal{F}(k) \tag{7}$$

with

$$\mathcal{F}(g(x)) = G(u) = \int_{-\infty}^{\infty} g(x) e^{-j2\pi ux} dx \tag{8}$$

To show that this holds, we first show that the Fourier transformation **as a system** has specific properties when the signal is shifted. If we shift a signal/function $g(x)$ by a in the spatial domain expressed by $g(x-a)$ this results in a linear phase shift in the Fourier domain:

$$\mathcal{F}(g(x-a)) = \mathcal{F}(g(x')) = \int_{-\infty}^{\infty} g(x') e^{-j2\pi u(x'+a)} dx' \tag{9}$$

where $e^{-j2\pi u(x'+a)} = e^{-j2\pi ua} e^{-j2\pi ux'}$ and $e^{-j2\pi ua}$ is a constant, such that

$$\mathcal{F}(g(x-a)) = e^{-j2\pi ua} G(u) \tag{10}$$

Using the shift property of the Fourier transform we can now prove the convolution theorem. The continuous convolution is defined as follows:

$$g(x) \circledast k(x) = \int_{-\infty}^{\infty} g(x) k(y-x) dx \tag{11}$$

The Fourier transformation of $g(x) \circledast k(x)$ is defined by:

$$\int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} g(x)k(y-x)dx \right] e^{-j2\pi uy} dy \tag{12}$$

By reversing the order of the integration we get

$$\int_{-\infty}^{\infty} g(x) \left[\int_{-\infty}^{\infty} k(y-x)e^{-j2\pi uy} dy \right] dx \tag{13}$$

where we can pull out $g(x)$. Given the shift property, the inner integration can be defined by:

$$\int_{-\infty}^{\infty} k(y-x)e^{-j2\pi uy} dy = \mathcal{F}(k(y-x)) = e^{-j2\pi ux} K(u) \tag{14}$$

such that

$$\begin{aligned} &\int_{-\infty}^{\infty} g(x) \left[\int_{-\infty}^{\infty} k(y-x)e^{-j2\pi uy} dy \right] dx \\ &= \int_{-\infty}^{\infty} g(x)e^{-j2\pi ux} K(u) dx \\ &= \left[\int_{-\infty}^{\infty} g(x)e^{-j2\pi ux} dx \right] K(u) \\ &= G(u)K(u) = \mathcal{F}(g)(u)\mathcal{F}(k)(u), \end{aligned} \tag{15}$$

so that for all spatial frequencies u, we have

$$\mathcal{F}(g \circledast k)(u) = \mathcal{F}(g)(u) \odot \mathcal{F}(k)(u). \tag{16}$$

K Fast Fourier Transform

The Discrete Fourier Transform (DFT) of an input signal $f(n)$ with N samples is defined as

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-j2\pi kn/N} \tag{17}$$

Executing the DFT directly would take $O(N^2)$. Thus Cooley & Tukey (1965) developed the Fast Fourier Transform, short FFT. Which builds upon a divide and concur strategy and reduces the runtime down to $O(NlogN)$.

They used the inherent symmetry which results from the period nature of the transformed signal. To give an intuition for this inherent symmetry lets explore what happens if we shift by N:

$$\begin{aligned} F(k+N) &= \sum_{n=0}^{N-1} f(n)e^{-j2\pi(k+N)n/N}, \\ &= \sum_{n=0}^{N-1} f(n)e^{-j2\pi n} e^{-j2\pi kn/N}, \\ &= \sum_{n=0}^{N-1} f(n)e^{-j2\pi kn/N}, \end{aligned} \tag{18}$$

as $e^{j2\pi n} = 1$ for any integer n. Thus one can see that

$$F(k+N) = F(k) \tag{19}$$

and also

$$F(k+iN) = F(k) \tag{20}$$

for any integer i holds.

Given this symmetry, Cooley & Tukey (1965) developed an algorithm which divides the DFT into smaller parts such that the DFT can be solved via divide and concur. Following we rearrange the DFT into two parts:

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N} \\ &= \sum_{m=0}^{N/2-1} f(2m) e^{-j2\pi k2m/N} \\ &+ \sum_{m=0}^{N/2-1} f(2m+1) e^{-j2\pi k(2m+1)/N} \\ &= \sum_{m=0}^{N/2-1} f(2m) e^{-j2\pi km/(N/2)} \\ &+ e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} f(2m+1) e^{-j2\pi km/(N/2)} \end{aligned} \tag{21}$$

Each part represents the even-numbered and odd-numbered values respectively. However, the runtime is still the same as each term consist of $O(N/2)N$ computations so in total still $O(N^2)$.

Luckily, this division into two parts can be continued in each part again. Hence, the range of k is $0 \leq k \leq N$ while m is now in the range of $0 \leq m \leq M$ where $M = N/2$. Thus, solving the problem only takes half of the computations as before, $O(N^2)$ becomes $O(M^2)$ where M is half the size of N. As long as M is even-valued, we can apply divide the problem in even smaller parts, applying the divide and concur strategy which in an recursive implementation takes only $O(NlogN)$.

L Code Base

Implementation code for our NIFF CNNs is provided at: `https://anonymous.4open.science/r/NIFF1528anonymous` and will be made publicly available upon acceptance.

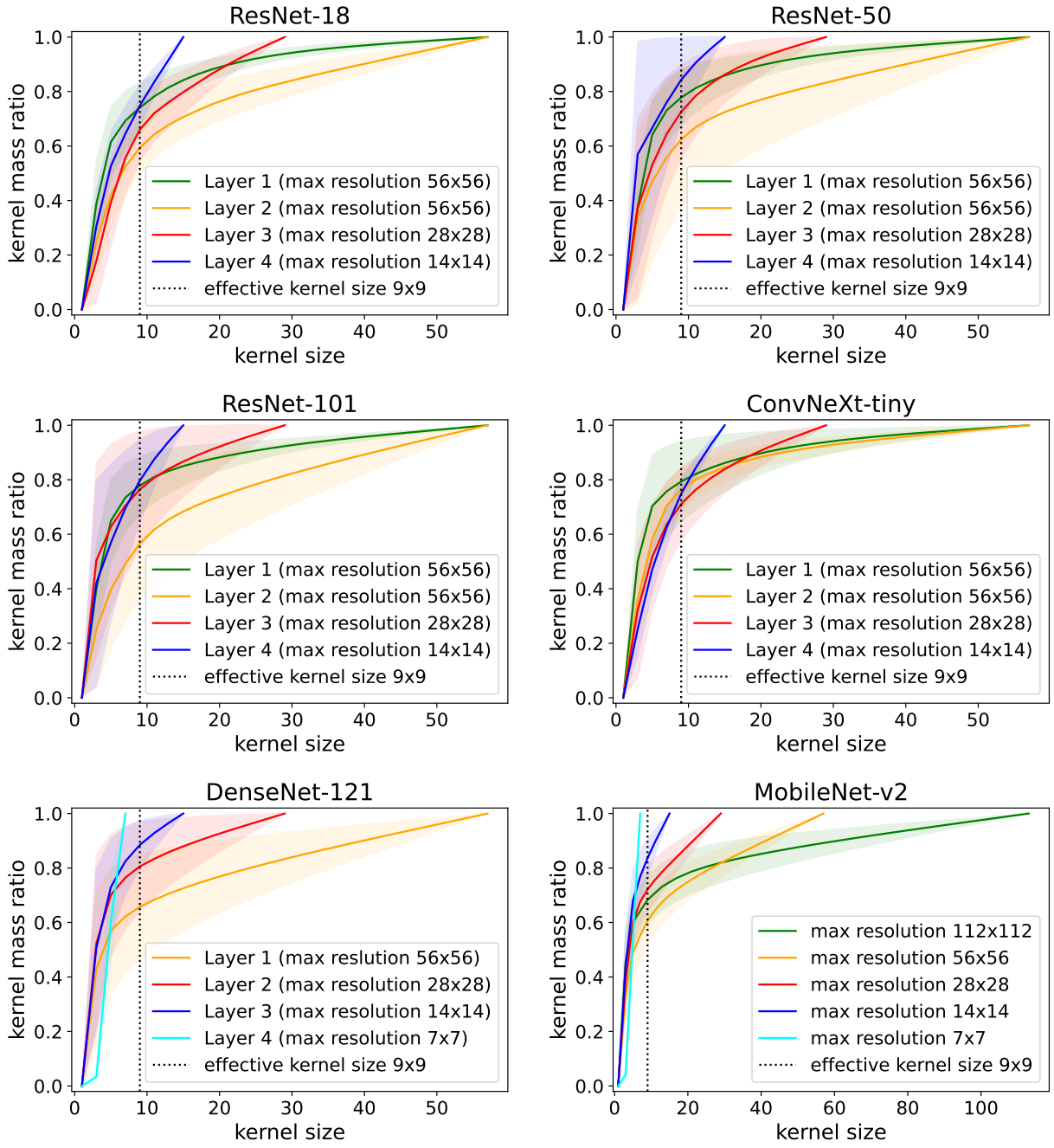

Figure A2: Effective kernel size evaluation on ImageNet-100. We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. For ResNet and ConvNeXt-tiny each layer encodes one resolution. Thus, the layers could be summarised (Layer 1 encoding 56×56, Layer 2 56×56, Layer 3 28×28 and Layer 4 14×14). For DenseNet-121 each layer can be summarised similarly, yet the after the first layer the feature maps are already downsampled resulting in the following: Layer 1 encoding 56×56, Layer 2 28×28, Layer 3 14×14 and Layer 4 7×7. However, for MobileNet-v2 the resolution is downsampled within a layer.

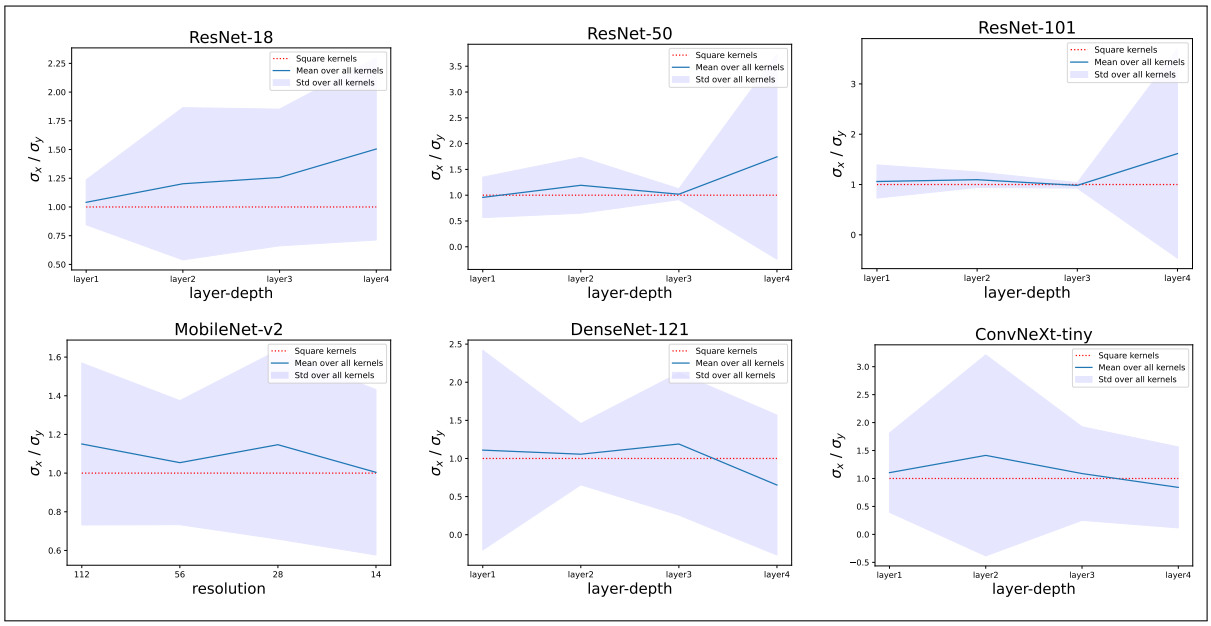

Figure A3: Analysis of non-square kernel shapes inspired by Romero et al. (2022a) on ImageNet-1k. We compare the variance σ_x and σ_y in x- and y-direction of a Gaussian fitted onto our learned spatial weights. The red dashed line indicates square-shaped kernels as the variance σ_x and σ_y are equal.

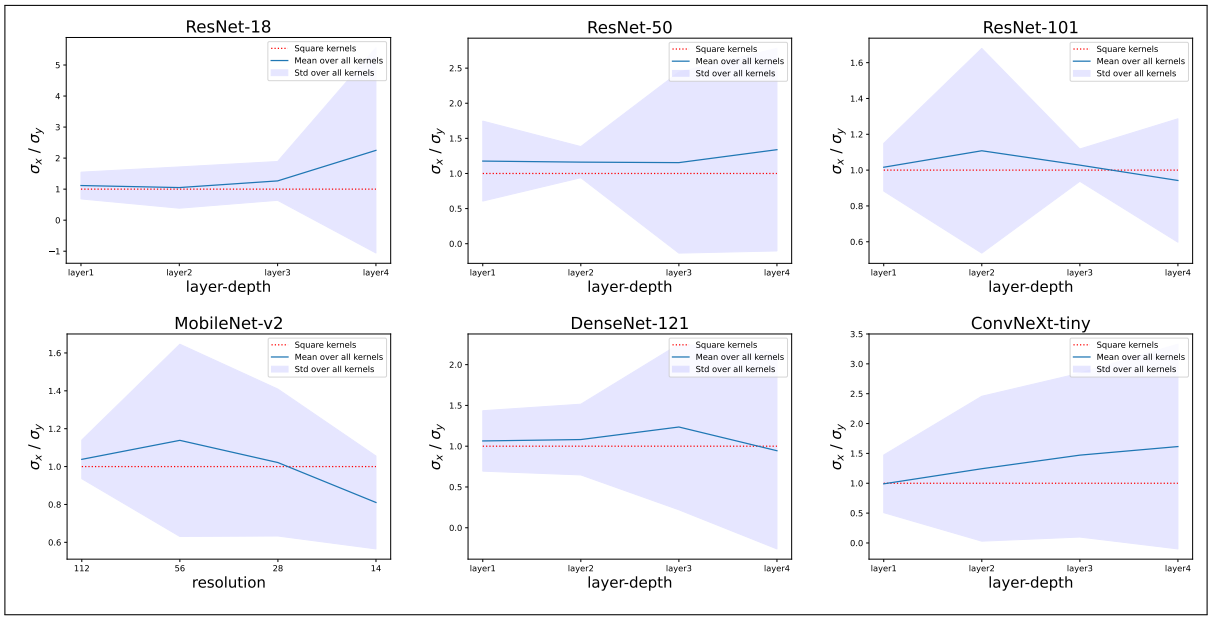

Figure A4: Analysis of non-square kernel shapes inspired by Romero et al. (2022a) on ImageNet-100. We compare the variance σ_x and σ_y in x- and y-direction of a Gaussian fitted onto our learned spatial weights. The red dashed line indicates square-shaped kernels as the variance σ_x and σ_y are equal.

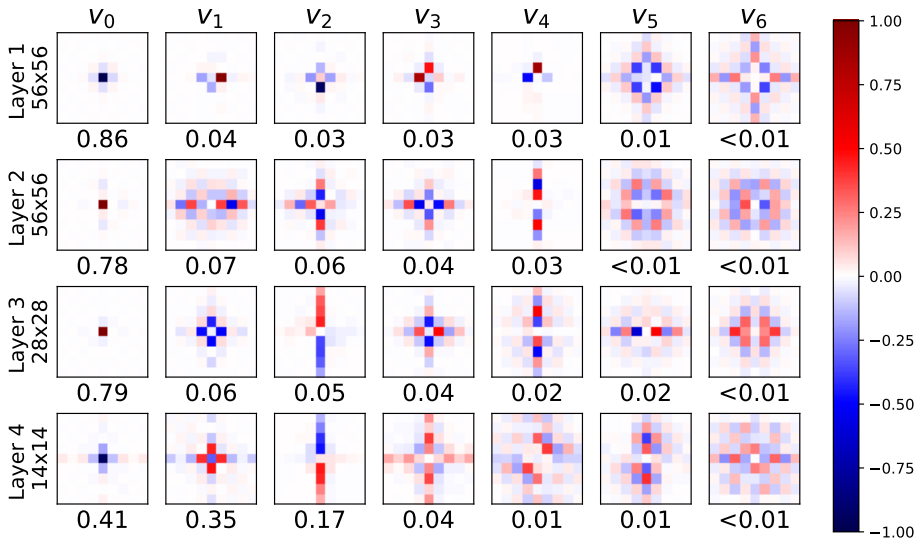

Figure A5: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. ConvNeXt convolutions are standardly equipped with larger kernel sizes than usual (7×7). However, our analysis reveals that the network barely uses large filters if it gets the opportunity to learn large filters. The learned filters in the first and third layer mostly use small (3×3), well-localized filters.

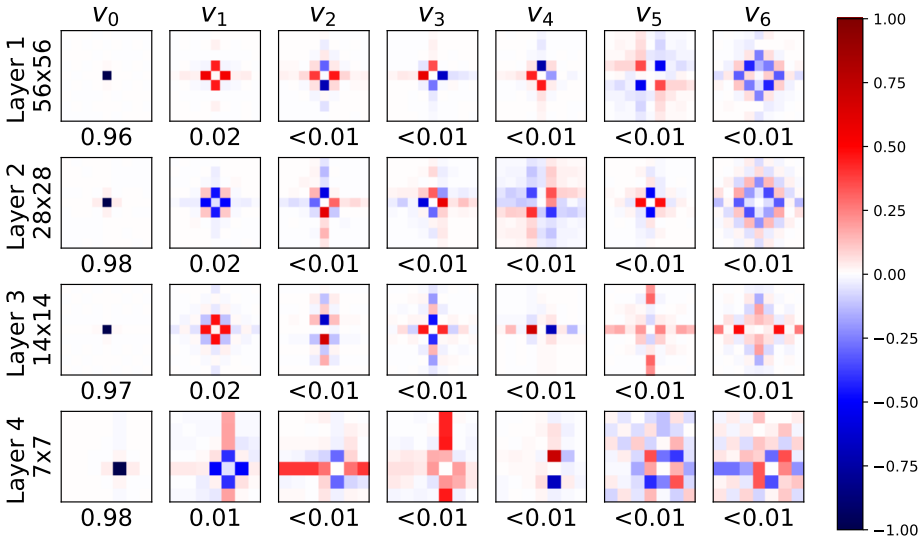

Figure A6: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

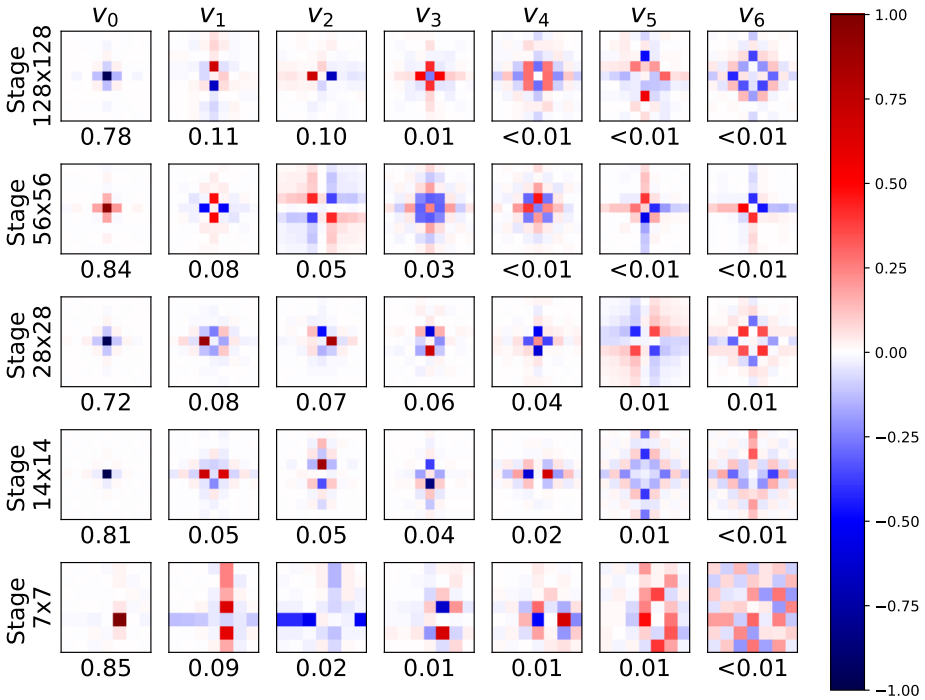

Figure A7: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding stage is given. For MobileNet-v2 the feature maps are downsampled within a layer, thus the stages are combine by feature maps size rather than the layers. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

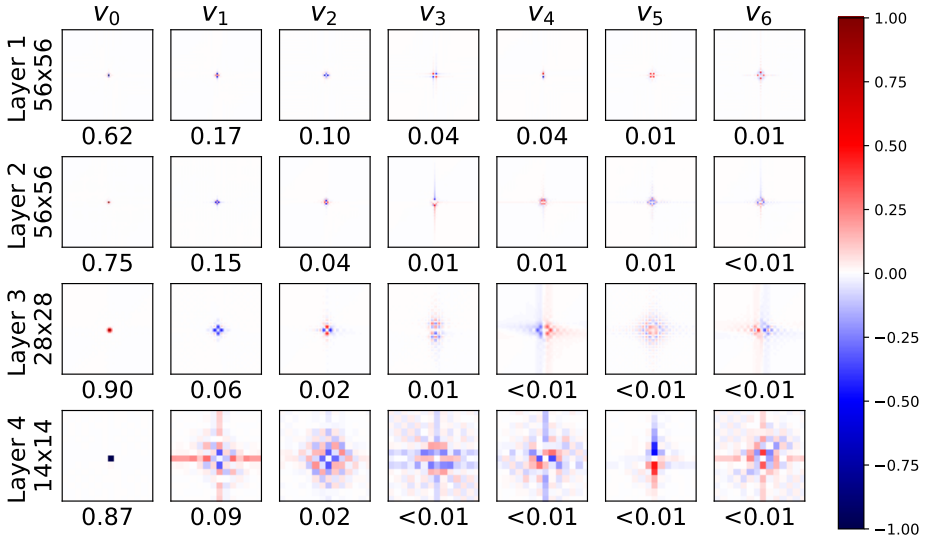

Figure A8: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-1k original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

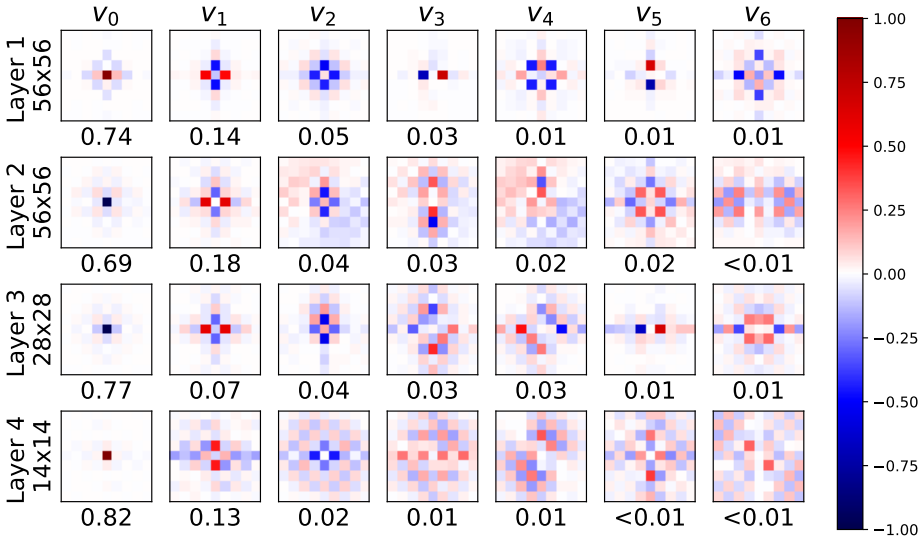

Figure A9: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a really small kernel size although they could use a much bigger kernel.

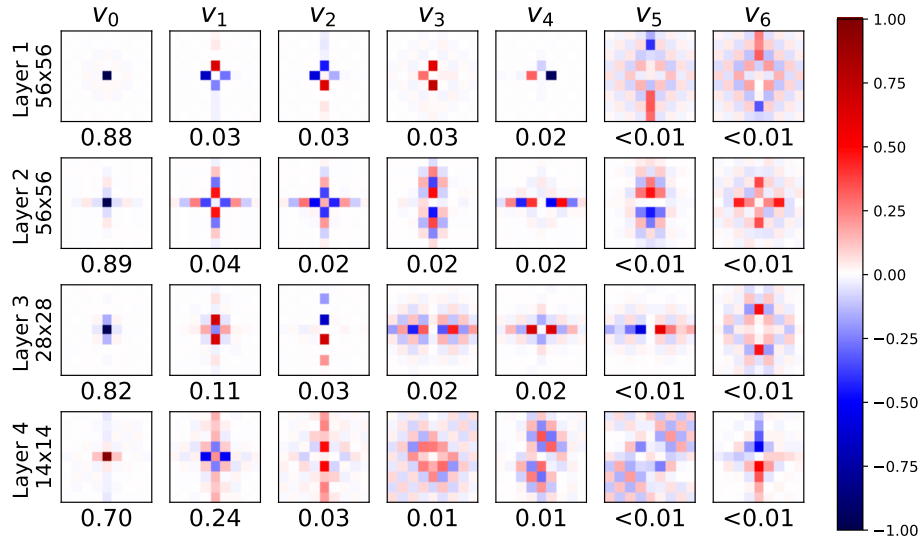

Figure A10: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a really small kernel size although they could use a much bigger kernel.

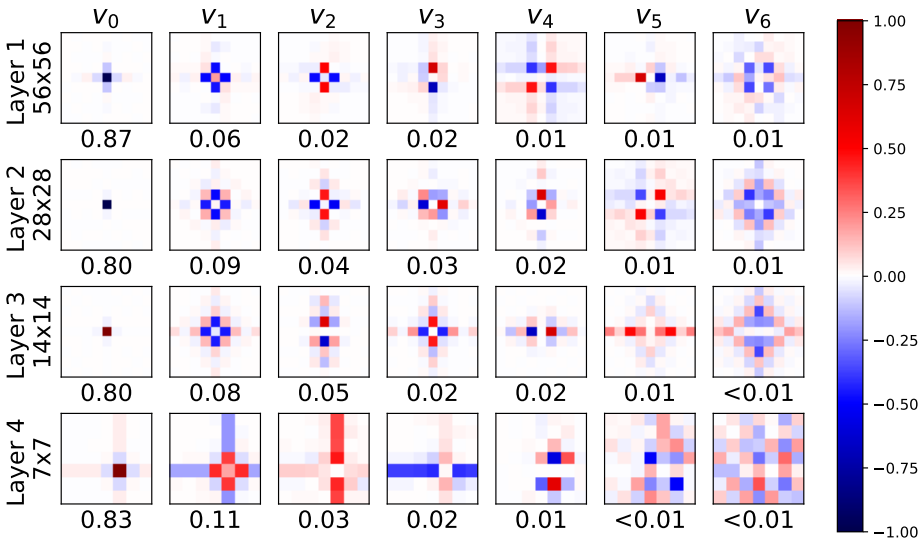

Figure A11: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

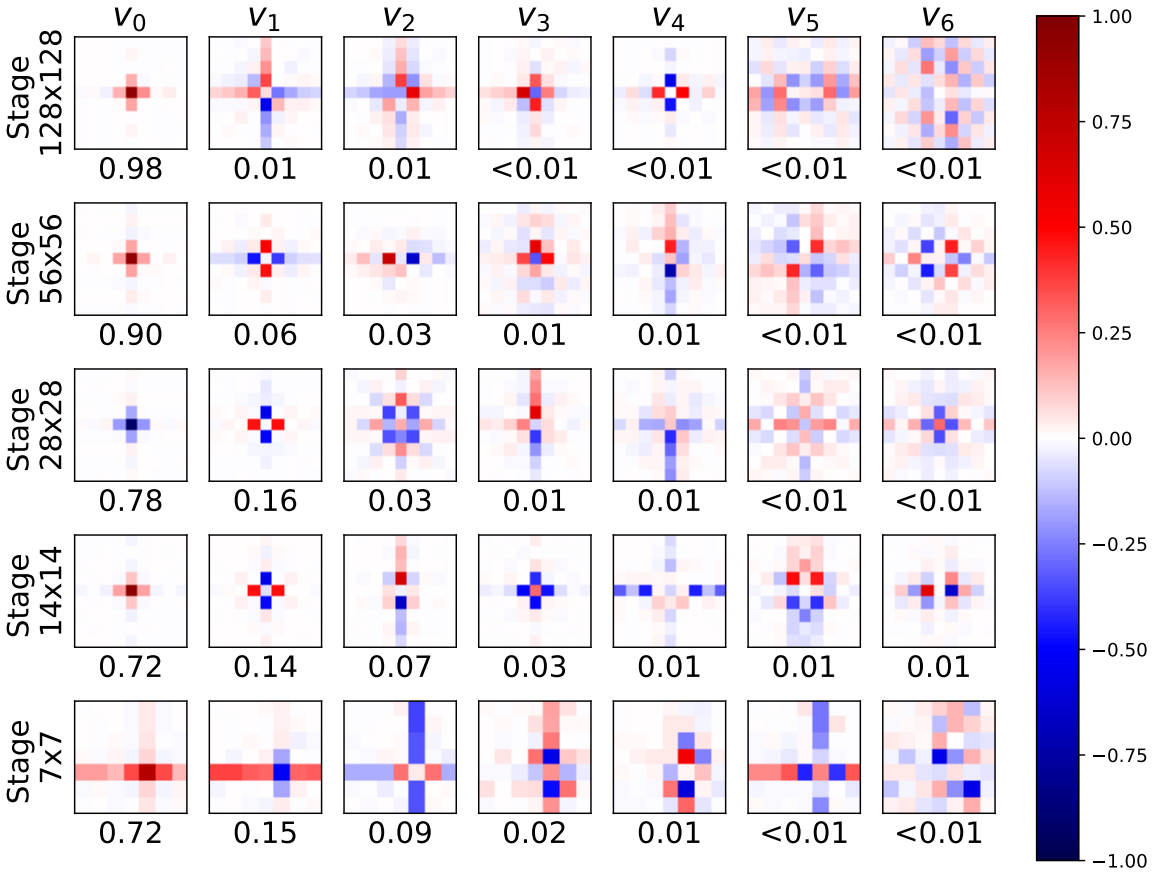

Figure A12: PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding stage is given. For MobileNet-v2 the feature maps are downsampled within a layer, thus the stages are combine by feature maps size rather than the layers. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

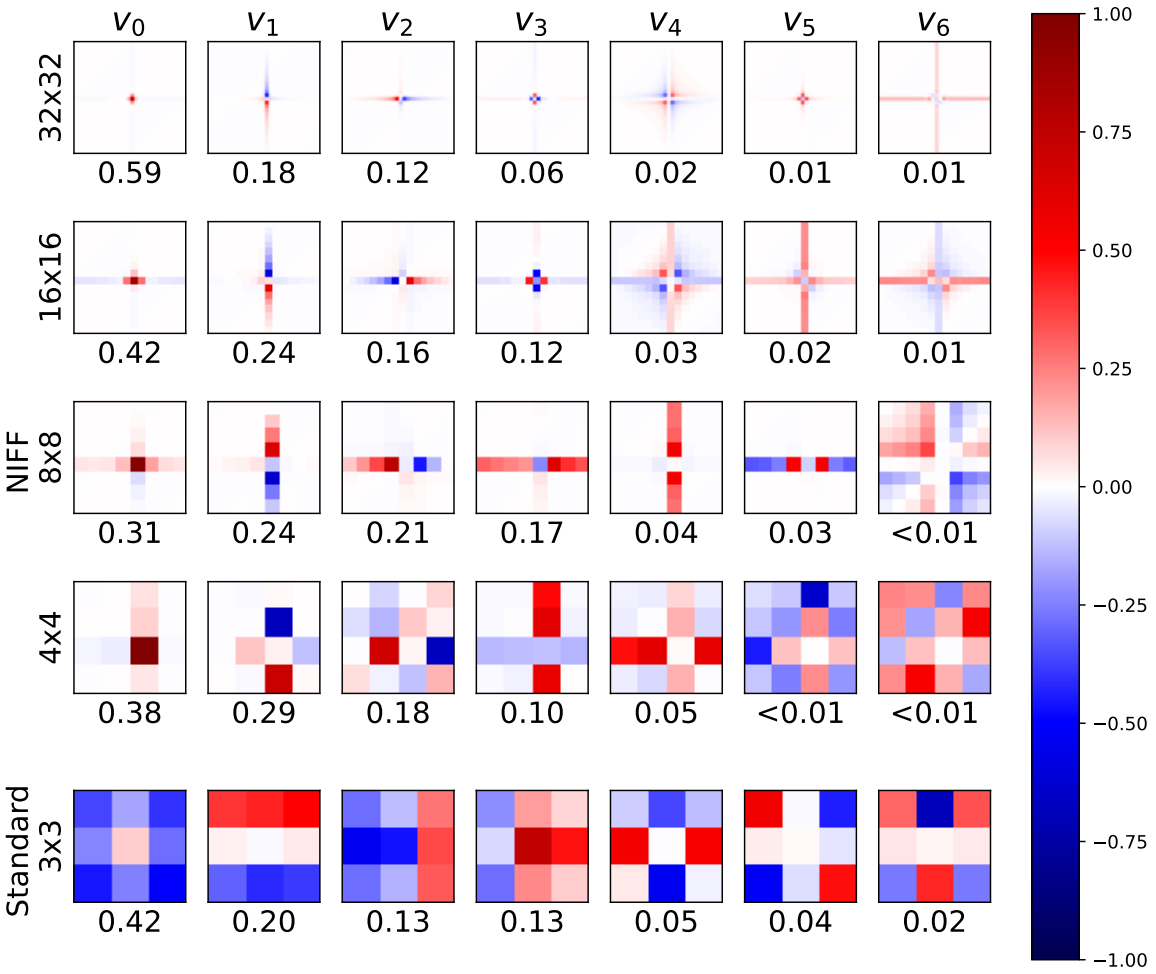

Figure A13: PCA basis and explained variance for each basis vector (below) of all spatial filters for each resolution for the NIFF convolutions of a MobileNet-V2 trained on CIFAR-10 as well as the learned filters for the third layer of a standard MobileNet-V2 trained on CIFAR-10 (bottom row). On the right, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use much bigger kernels.

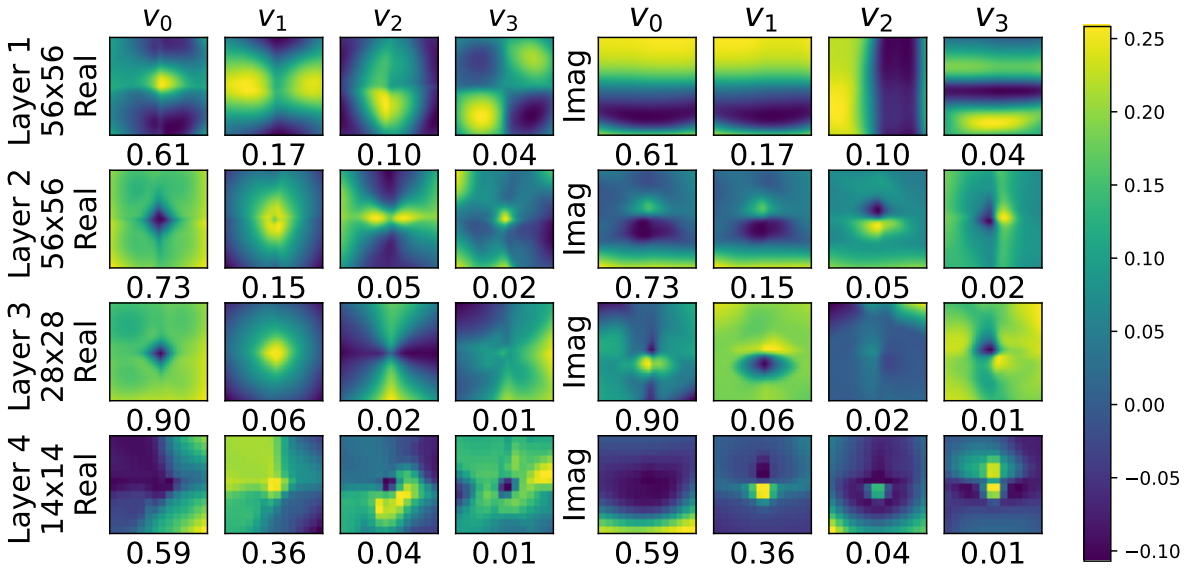

Figure A14: PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a ResNet-50 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

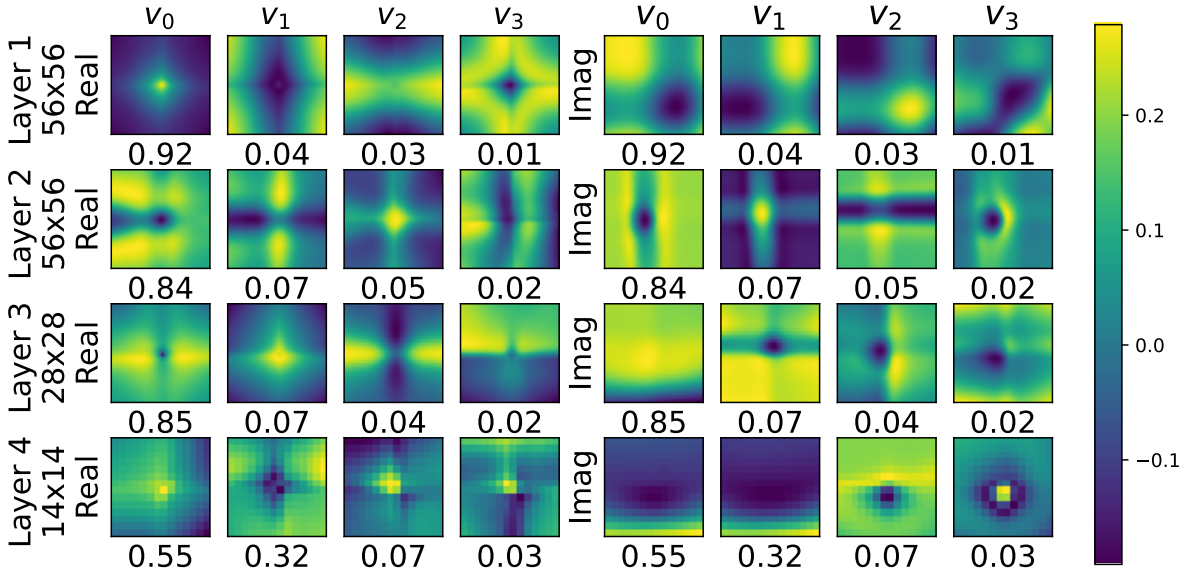

Figure A15: PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a ConvNeXt-tiny trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

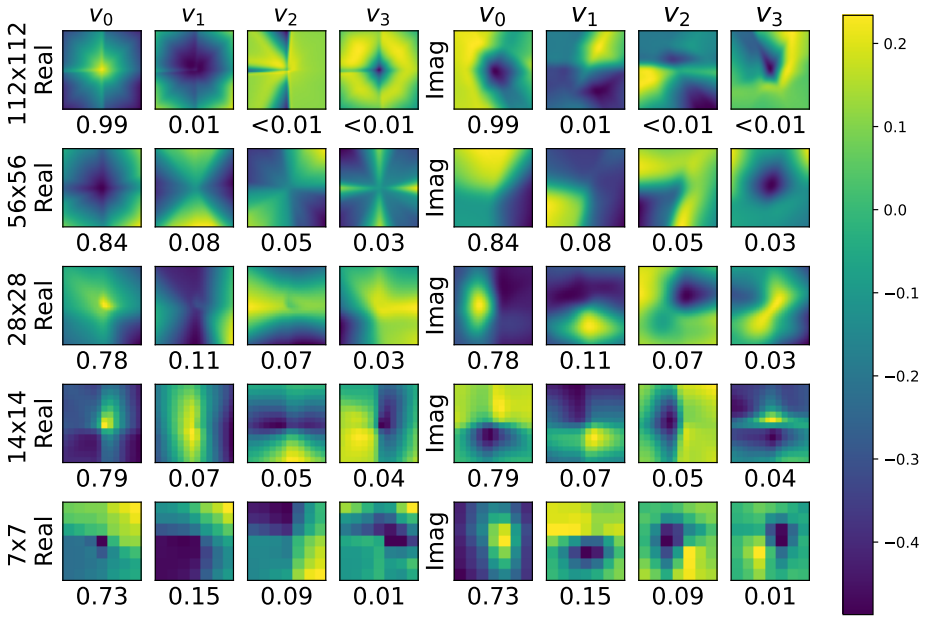

Figure A16: PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a MobileNet-v2 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

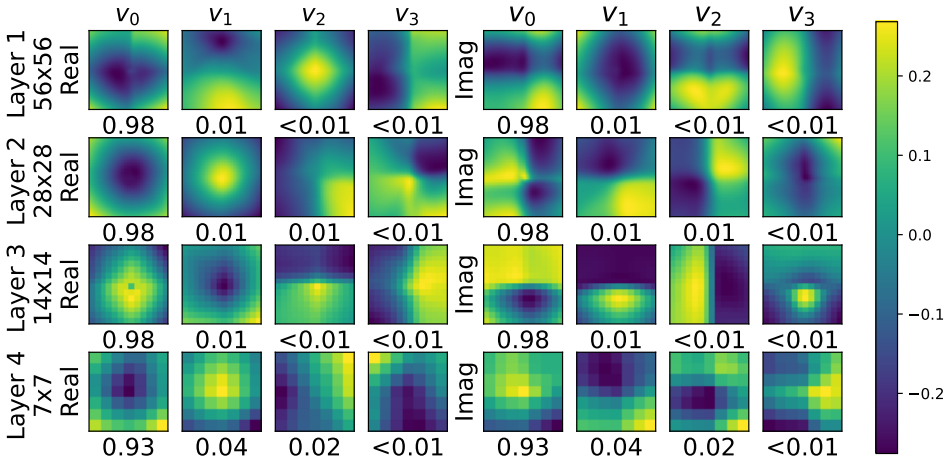

Figure A17: PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a DenseNet-121 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

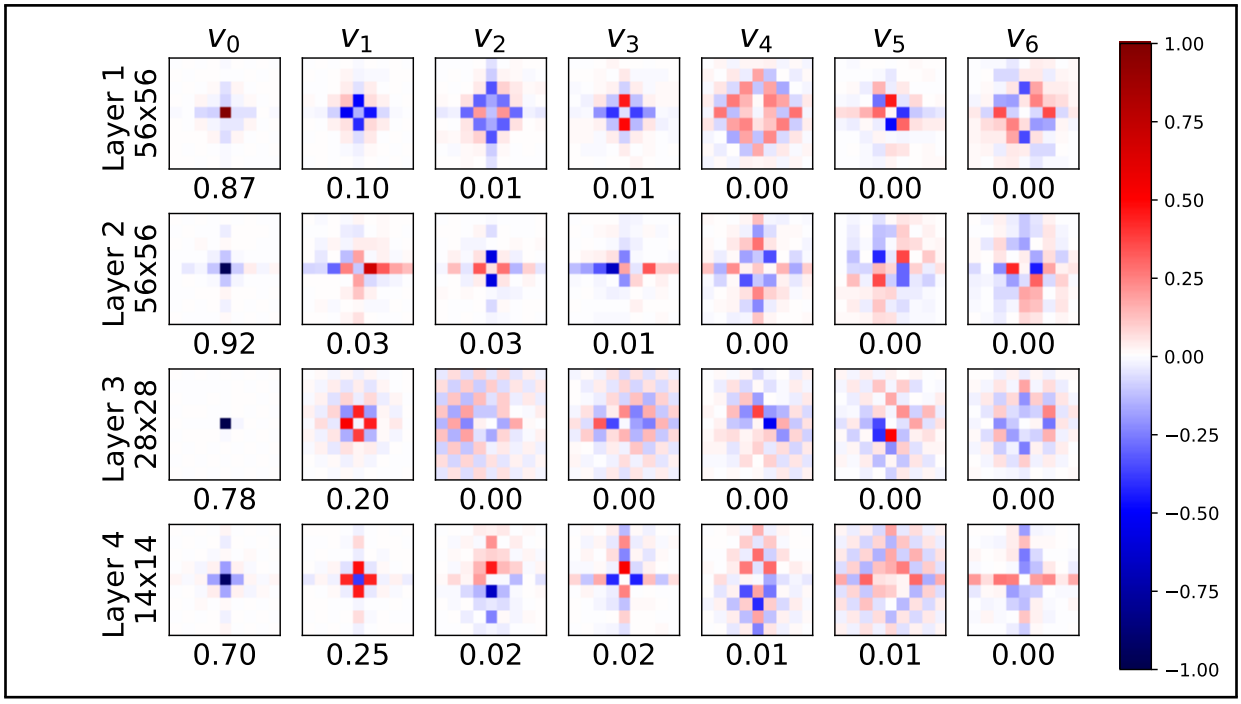

Figure A18: Actual kernels in the spatial domain of a ResNet-18 with additional zero padding before our NIFF trained on ImageNet-100. We plot for each kernel the zoomed-in (9×9) version below for better visibility. Still, most kernels exhibit well-localized, small spatial kernels.

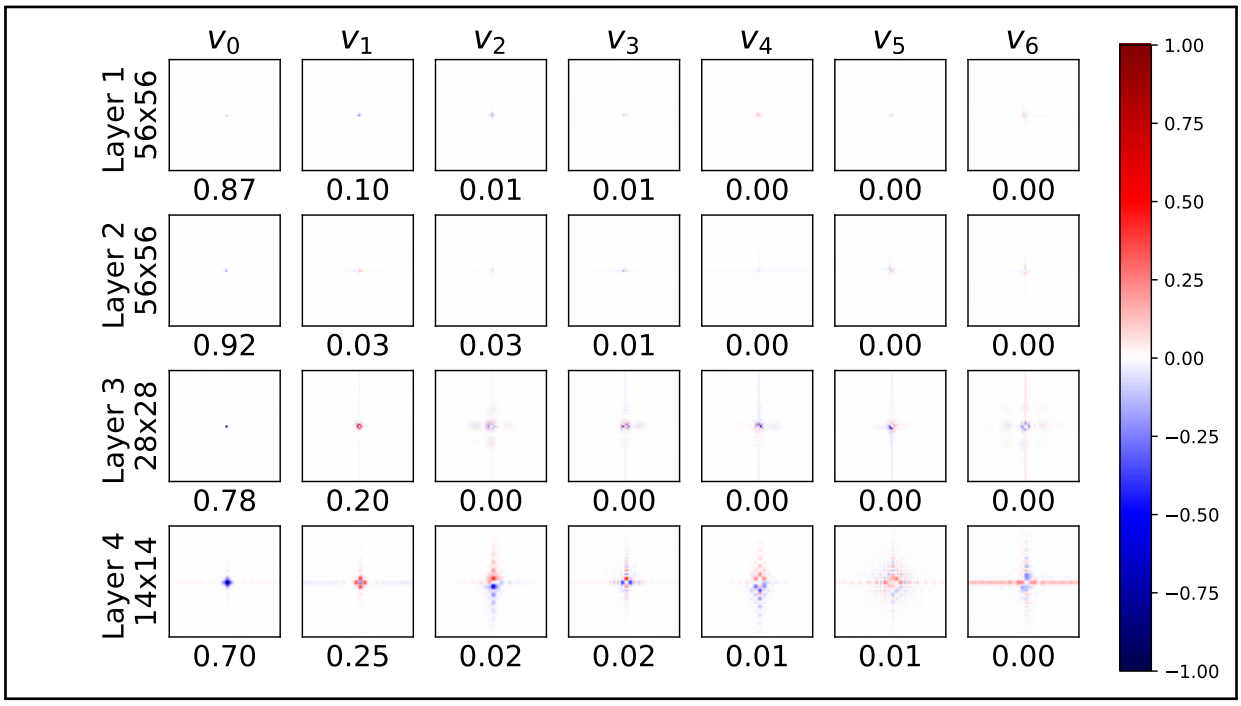

Figure A19: Actual kernels in the spatial domain of a ResNet-18 with additional zero padding before our NIFF trained on ImageNet-100. Still, most kernels exhibit well-localized, small spatial kernels.

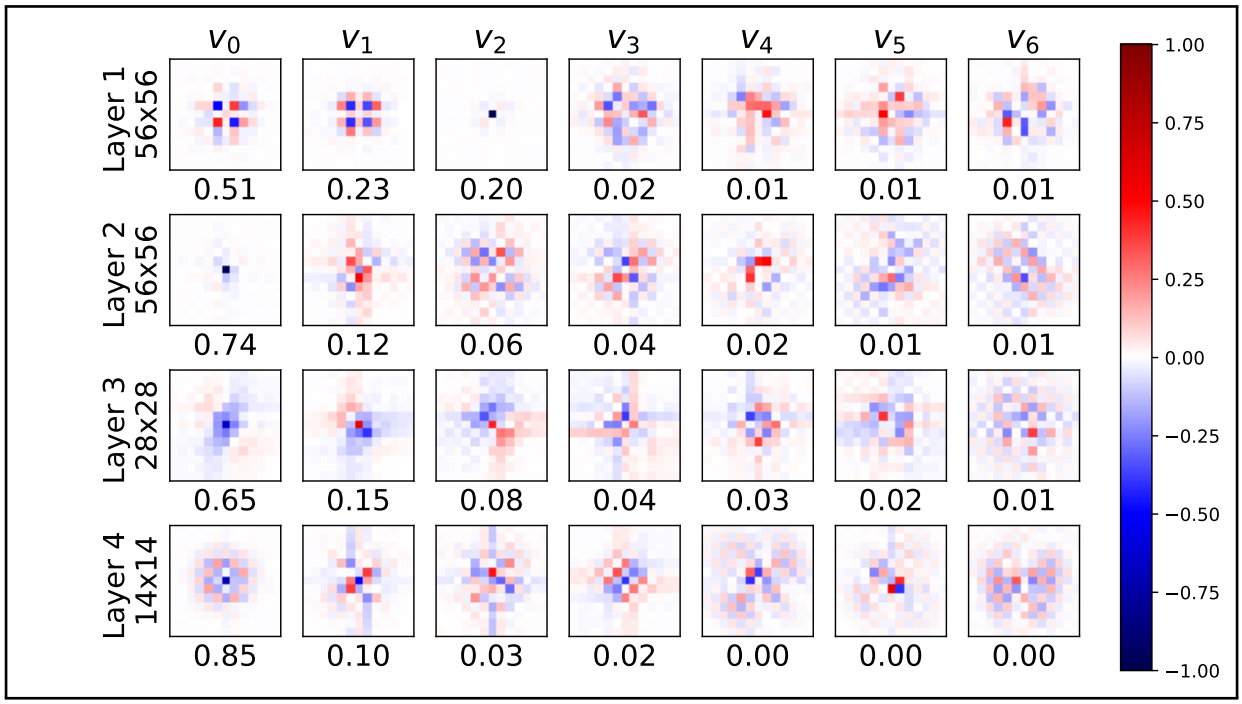

Figure A20: Actual kernels in the spatial domain of a ResNet-18 which mimics linear convolutions with our NIFF trained on ImageNet-100. We plot for each kernel the zoomed-in (13×13) version below for better visibility. Still, most kernels exhibit well-localized, small spatial kernels. However, they are slightly larger than the kernels learned without padding and cropping.

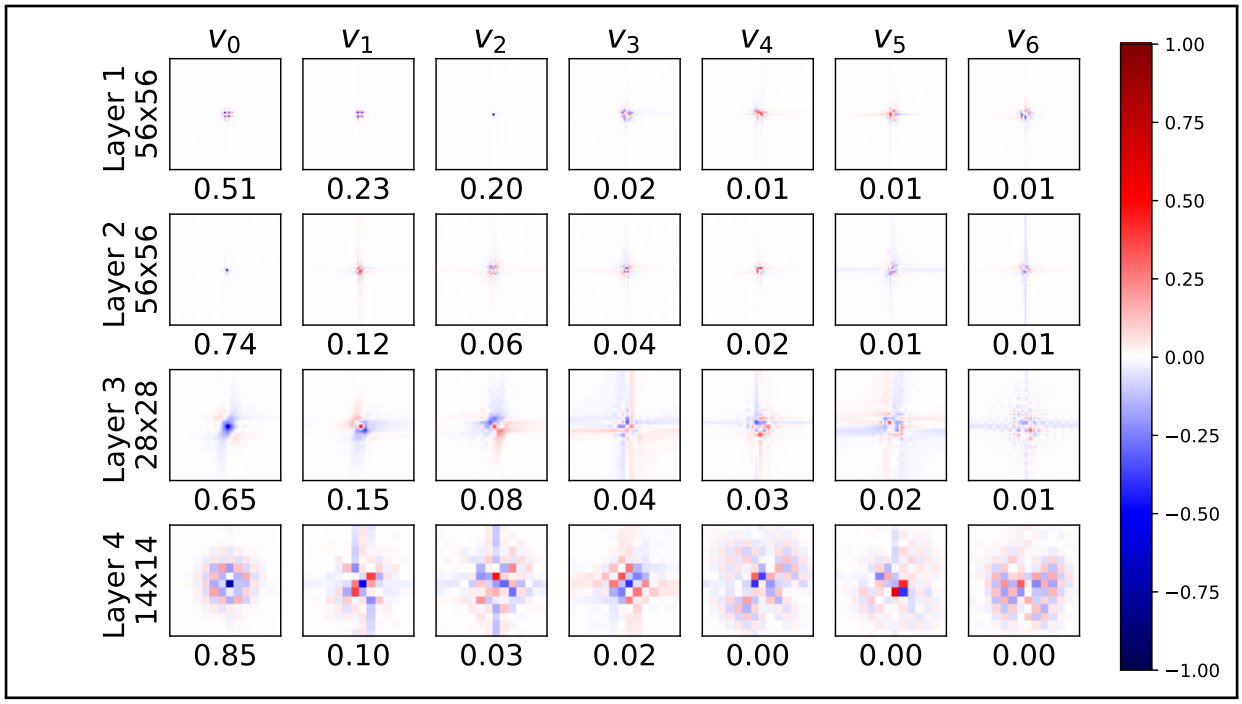

Figure A21: Actual kernels in the spatial domain of a ResNet-18 which mimics linear convolutions with our NIFF to mimic linear convolutions trained on ImageNet-100. Still, most kernels exhibit well-localized, small spatial kernels. However, they are slightly larger than the kernels learned without padding and cropping.

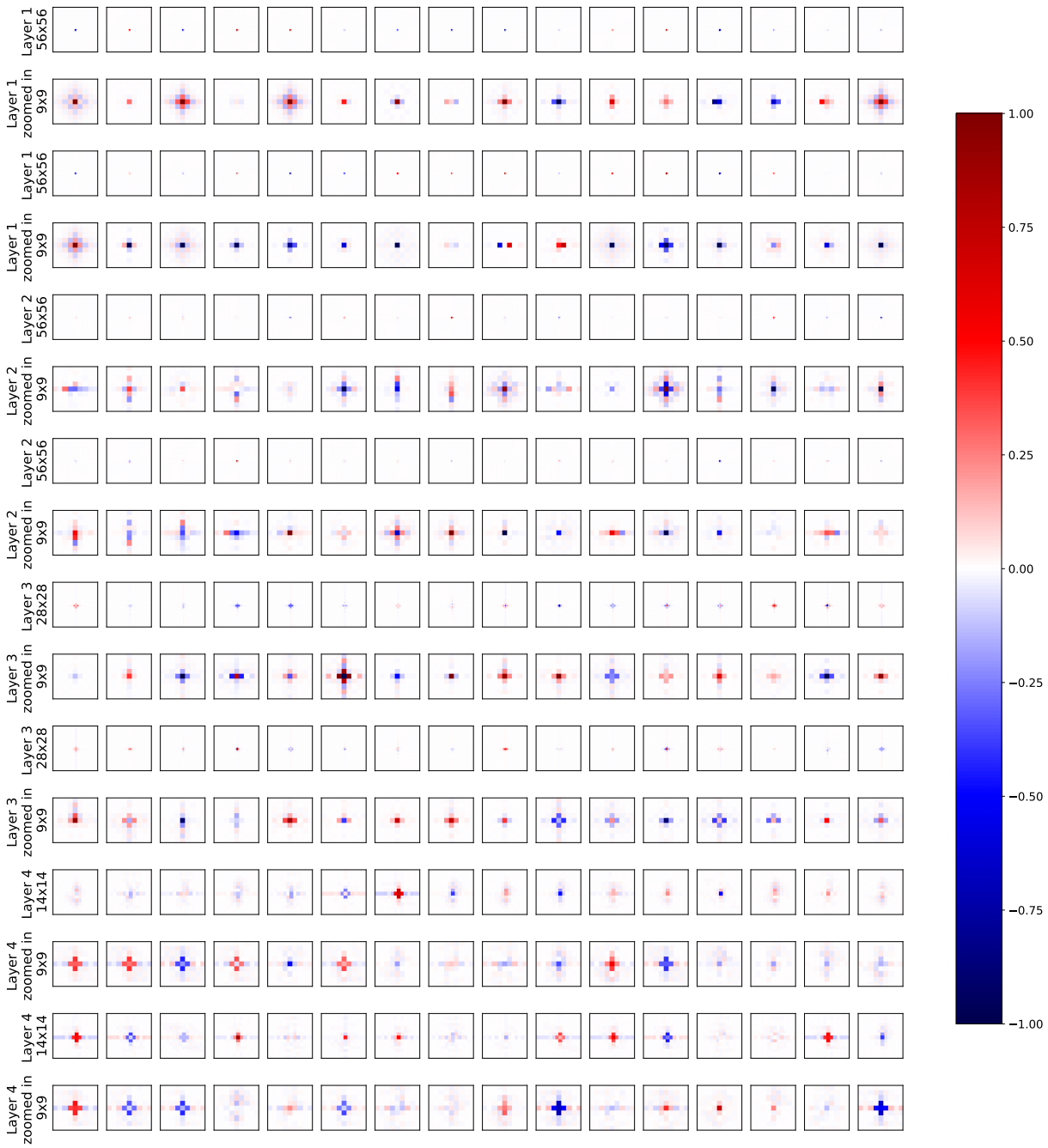

Figure A22: Actual kernels in the spatial domain of a ConvNeXt-tiny including our NIFF trained on ImageNet-1k. We plot for each kernel the zoomed-in (9×9) version below for better visibility. Overall, most kernels exhibit well-localized, small spatial kernels.

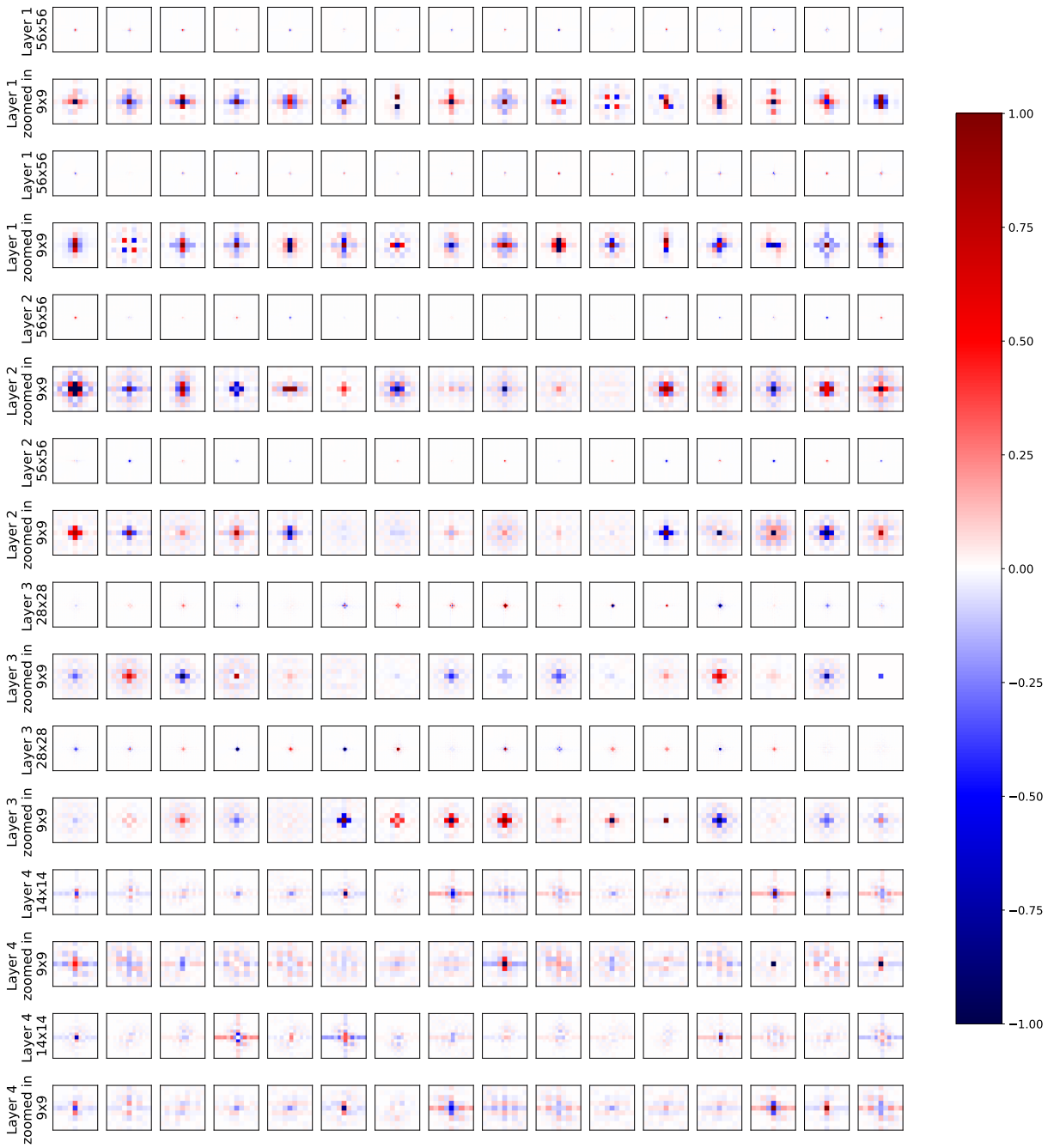

Figure A23: Actual kernels in the spatial domain of a ResNet-50 including our NIFF trained on ImageNet-1k. We plot for each kernel the zoomed-in (9×9) version below for better visibility. Overall, most kernels exhibit well-localized, small spatial kernels.

References

Shashank Agnihotri, Julia Grabinski, and Margret Keuper. Improving stability during upsampling–on the importance of spatial context. *arXiv preprint arXiv:2311.17524*, 2023.

Sayed Omid Ayat, Mohamed Khalil-Hani, Ab Al-Hadi Ab Rahman, and Hamdan Abdellatef. Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. *Neurocomputing*, 364: 152–167, 2019.

Ron Bracewell and Peter B Kahn. The fourier transform and its applications. *American Journal of Physics*, 34(8):712–712, 1966.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4479–4488. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf.

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21638–21652. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f69e505b08403ad2298b9f262659929a-Paper.pdf.

Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11963–11975, June 2022.

Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12124–12134, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

George H Dunteman. *Principal components analysis*, volume 69. Sage, 1989.

Ricard Durall, Margret Keuper, and Janis Keuper. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. An Alan R. Apt book. Prentice Hall, 2003. ISBN 9780130851987. URL https://books.google.de/books?id=VAd5QgAACAAJ.

Julia Grabinski, Steffen Jung, Janis Keuper, and Margret Keuper. Frequencylowcut pooling – plug & play against catastrophic overfitting. In *European Conference on Computer Vision*, 2022. URL https://arxiv.org/abs/2204.00491.

Julia Grabinski, Janis Keuper, and Margret Keuper. Fix your downsampling asap! be natively more robust via aliasing and spectral artifact free pooling. *arXiv preprint arXiv:2307.09804*, 2023.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Bochen Guan, Jinnian Zhang, William A Sethares, Richard Kijowski, and Fang Liu. Specnet: spectral domain convolutional neural network. *arXiv preprint arXiv:1905.10915*, 2019.

Meng-Hao Guo, Cheng-Ze Lu, Qibin Hou, Zhengning Liu, Ming-Ming Cheng, and Shi-Min Hu. Segnext: Rethinking convolutional attention design for semantic segmentation. *arXiv preprint arXiv:2209.08575*, 2022.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Steffen Jung and Margret Keuper. Spectral distribution aware image generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 1734–1742, 2021.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Mykola Pechenizkiy, Decebal Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. *arXiv preprint arXiv:2207.03620*, 2022a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

Jovita Lukasik, Paul Gavrikov, Janis Keuper, and Margret Keuper. Improving native cnn robustness with filter frequency regularization. *Transactions on Machine Learning Research*, 2023.

Tianyu Ma, Adrian V Dalca, and Mert R Sabuncu. Hyper-convolution networks for biomedical image segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1933–1942, 2022.

Tianyu Ma, Alan Q. Wang, Adrian V. Dalca, and Mert R. Sabuncu. Hyper-convolutions via implicit kernels for medical image analysis. *Medical Image Analysis*, 86:102796, 2023. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2023.102796. URL https://www.sciencedirect.com/science/article/pii/S1361841523000579.

Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

Hengyue Pan, Yixin Chen, Xin Niu, Wenbo Zhou, and Dongsheng Li. Learning convolutional neural networks in the frequency domain, 2022. URL https://arxiv.org/abs/2204.06718.

Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Silvia L Pintea, Nergis Tömen, Stanley F Goes, Marco Loog, and Jan C van Gemert. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, 30: 8342–8353, 2021.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng. Fcnn: Fourier convolutional neural networks. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 786–798, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71249-9.

Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=K_Mnsw5VoOW.

David W. Romero, Robert-Jan Bruintjes, Jakub Mikolaj Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=3jooF27-0Wy.

David W. Romero, Anna Kuzina, Erik J Bekkers, Jakub Mikolaj Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=8FhxBtXSl0.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. *arXiv preprint arXiv:1910.11093*, 2019.

Nergis Tomen and Jan C van Gemert. Spectral leakage and rethinking the kernel size in cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5138–5147, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1T2hmZAb.

Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zelong Wang, Qiang Lan, Dafei Huang, and Mei Wen. Combining fft and spectral-pooling for efficient convolution neural network model. In *2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016)*, pp. 203–206. Atlantis Press, 2016.

Thomio Watanabe and Denis F Wolf. Image classification in frequency domain with 2srelu: a second harmonics superposition activation function. *Applied Soft Computing*, 112:107851, 2021.

Shmuel Winograd. On computing the discrete fourier transform. *Mathematics of computation*, 32(141): 175–199, 1978.

Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.