Delft University of Technology

Luuk Haarman (4931173)

Master thesis

Data-efficient resolution transfer with Continuous Kernel Convolutional Neural Networks

MSc Computer Science, Data Science & Technology Track

Faculty Electrical Engineering, Mathematics, and Computer Science

Delft, September 11th 2023

Contents

1 Part 1: Introduction 1

2 Part 2: Scientific Article 3

3 Part 3: Supplemental Materials 14

Acknowledgements

I would like to thank my supervisor Robert-Jan Bruintjes and responsible professor Jan van Gemert for their feedback and guidance throughout the thesis. I would also like to thank fellow student Mark Basting for our insightful discussions.

1 Part 1: Introduction

This part aims to outline the structure of the thesis and explain the general storyline of the thesis.

Outline thesis

The outline of the thesis is as follows: the first part lays out the structure of the thesis and contains a brief general overview of the storyline of the thesis. The second part is a scientific article that presents the work done during the thesis. The third part contains supplemental material that explains the core concepts of deep learning, convolutional neural networks as well as continuous kernel convolutional neural networks that help understand part 2.

General storyline thesis

The thesis is about "data-efficient resolution transfer with continuous kernel CNNs". The term resolution transfer is used to refer to the process of pre-training a deep learning image model on low-resolution images before fine-tuning on high-resolution images to improve the model's high-resolution performance. The thesis contains the results of an investigation into the use of continuous kernel CNNs (Convolutional Neural Networks) for resolution transfer, and to improve performance when there is not a lot of high-resolution data available. The term "zero-shot" is used to refer to evaluating the model on high-resolution images without fine-tuning, "few-shot" refers to evaluating the model after fine-tuning with just a few samples, and "many-shot" refers to evaluating the model after fine-tuning with many samples.

Conventional CNNs are discrete. Discrete CNNs assign a learnable parameter to each value in the kernel that is used for convolution. Continuous kernel CNNs (CKCNNs) operate differently from discrete kernel CNNs. CKCNNs construct kernels by sampling a continuous function that is modelled by a small neural network. The sampling rate can be arbitrarily chosen, and so the kernel size is arbitrary while the amount of model parameters remains fixed, unlike normal discrete CNNs where the parameter amount depends on the kernel size.

I investigated the use of CKCNNs for resolution transfer. The fact that continuous kernel models can change their kernel size means that CKCNNs can adapt the kernel size respective to the change in image resolution. Adapting the kernel size to the change in resolution is called kernel resolution adaptation. A CKCNN can be trained on one resolution and perform almost as well on a resolution unseen during training. Conventional CNNs perform significantly worse on resolutions unseen during training. The hypothesis of the thesis is that kernel resolution adaptation can be used to transfer models to the new high resolution and improve the model with the limited amount of high-resolution images available in a way conventional CNNs can not.

Existing work has already demonstrated the ability of continuous kernel CNNs (CKCNNs) to perform well on image resolutions unseen during training and when fine-tuned on the entire dataset. I also investigate the data-efficiency of CKCNNs by fine-tuning on a limited amount of data. For kernel resolution adaptation to work well, the frequencies of the continuous functions of the CKCNN kernels need to be constrained to below the Nyquist Frequency of the kernel to avoid artefacts being introduced when the kernel size is changed. The Nyquist Frequency of the CKCNN kernel depends on the kernel size of the model.

Despite the existing work's good performance on zero-shot resolution transfer, training the models is slow and expensive in terms of memory compared to conventional CNNs. The reason for this is because the models use very large kernels that make convolution costly and do not lower the resolution of the images during the network inference, while conventional CNN models use small kernels and use down-sampling layers to lower the resolution of images during the network inference to lower memory costs. Next to this, the method for constraining the frequencies of the continuous functions in the kernels lacks an anti-aliasing guarantee. Aliasing occurs when the frequencies of the kernels are too high and significantly affect kernel resolution adaptation performance. A lack of anti-aliasing guarantee means that kernel resolution adaptation does not always work well.

In my thesis, I investigated the use of CKCNNs to use their good zero-shot resolution transfer performance to improve the performance when there is a low amount of high-resolution data available, referred to as few-shot resolution transfer. Next to this, I aimed to make the models faster and lower computational costs to make training the models on higher resolutions more comparable to training conventional models. I resolved the issue of a lack of anti-aliasing guarantee with an alternative parameterization for the continuous function that constructs the kernels. This parameterization has a fixed frequency range with a maximum frequency that can be set when the model is created. I also improved the fine-tuning of the models with an additional module that complements the low-frequency kernels by focusing on high-frequency features. With these changes, I improved the results of CKCNNs with small kernels and demonstrated that continuous kernel models can be used to pre-train models on low-resolution images, and achieve better results than conventional CNNs when fine-tuned with a limited amount of high-resolution data.

Data-efficient resolution transfer with Continuous Kernel Convolutional Neural Networks

Luuk Haarman

TU Delft

Abstract

Convolutional Neural Networks (CNNs) benefit from fine-grained details in high-resolution images, but these images are not always easily available as data collection can be expensive or time-consuming. Transfer learning pre-trains models on data from a related domain before fine-tuning on the main domain, and is a common strategy to deal with limited data. However, transfer learning requires a similar domain with enough available data to exist, and transferability varies from task to task. To deal with limited high-resolution data we propose resolution transfer: using low-resolution data to improve high-resolution accuracy. For resolution transfer, we use Continuous kernel CNNs (CKCNNs) that can adapt their kernel size to changes in resolution and perform well on unseen resolutions. Training CKCNNs on high-resolution images is currently significantly slower than CNNs. We lower the inference costs of CKCNNs to enable training on high-resolution data. We introduce a CKCNN parameterization that constrains the frequencies of kernels to avoid distortions when the kernel size is changed, improving resolution transfer accuracy. We improve fine-tuning with a High-Frequency Adaptation module that complements our constrained kernels. We demonstrate that CKCNNs with kernel resolution adaptation outperform CNNs for resolution transfer tasks with no fine-tuning or with limited fine-tuning data. We compare to transfer learning, and achieve competitive classification accuracy with an ImageNet pre-trained ResNet-18. Our method provides an alternative to transfer learning that uses low-resolution data to improve classification accuracy when high-resolution data is limited.

1. Introduction

With the improving performance of deep learning models comes a demand for high-resolution images in training Convolutional Neural Networks (CNNs) for tasks that require the fine-grained details absent in low-resolution images to attain good performance [4, 6, 27]. However, acquiring large amounts of high-resolution data for certain domains can be costly or difficult. These domains could require specific cameras and sensors or face logistic challenges such as satellite imagery. Transfer learning, a common approach to deal with limited data, pre-trains on data from a related domain and fine-tunes the pre-trained model on available main task data. Domain adaptation, a sub-field of transfer learning, focuses specifically on adapting the model to the target domain. A challenge with these methods is when a compatible domain with sufficient data does not exist or when the model can not adapt well during fine-tuning due to a large domain shift. We propose resolution transfer to use low-resolution data from the same domain for pre-training the model before fine-tuning on the available high-resolution data. We refer to evaluating on the high-resolution data without fine-tuning as zero-shot resolution transfer, fine-tuning with limited data as few-shot resolution transfer, and with many samples as many-shot resolution transfer.

We propose the use of Continuous Kernel CNNs (CKCNNs) for data-efficient resolution transfer. CKCNNs construct arbitrary-size kernels by discretely sampling an underlying kernel generator network. Existing CKCNN works [23, 24] have demonstrated that CKCNNs can adapt to unseen resolutions with minimal accuracy loss using kernel resolution adaptation: changing their kernel size respective to the change in resolution. Training CKCNNs on high-resolution data is currently significantly slower than CNNs due to the use of kernels that span the entire image and the lack of down-sampling layers. We significantly speed up the training of CKCNNs on high-resolution images by using smaller kernels and pooling layers.

Aliasing occurs when a signal's sampling rate is too low to properly capture all of its frequency components, and causes distortions that negatively affect kernel resolution adaptation performance. To deal with aliasing, CKCNNs constrain the frequency content of their ker-

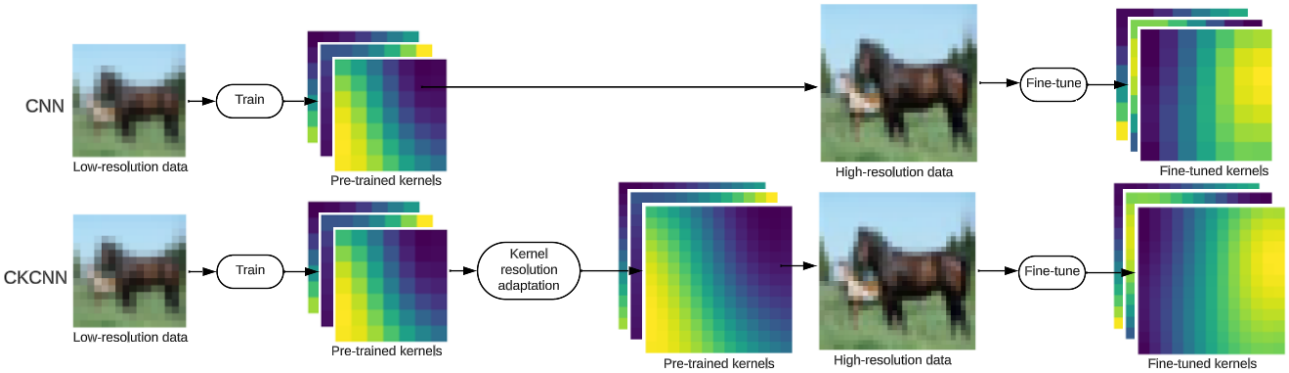


Figure 1. Resolution transfer pipeline for CNN (top row) and CKCNN (bottom row). Both models first train on low-resolution images, resulting in pre-trained kernels. Unlike CNNs, CKCNNs can adapt their kernel size respective to the change in resolution with minimal accuracy loss and no additional parameters. The adaptation in kernel size improves high-resolution accuracy with no fine-tuning or fine-tuned with limited data.

nels. Existing methods use regularisation or masking of high-frequency filters. We observe that the regularisation of kernels does not guarantee anti-aliasing and propose a parameterization with guaranteed anti-aliasing. Next to aliasing, spectral leakage creates unintended frequency components in kernels. We reduce spectral leakage with a Hamming window before convolution. We add a High-Frequency Adaptation module to improve fine-tuning on high-frequency features found in high-resolution images. We demonstrate that the addition of our module improves few-shot and many-shot classification accuracies.

In this paper, we investigate the use of CKCNNs with kernel resolution adaptation for data-efficient resolution transfer. We evaluate the resolution transfer performance of conventional discrete CNNs and CKCNNs, increasing the amount of fine-tuning data in each stage. We demonstrate that CKCNNs achieve higher classification accuracies than CNNs when high-resolution data is limited during fine-tuning, or not used at all.

In summary, our contributions are:

- We demonstrate that current CKCNNs are significantly slower than CNNs. We speed up the training of CKCNNs on high-resolution images with smaller kernels and pooling layers.
- We improve kernel resolution adaptation with a kernel parameterization that provides anti-aliasing guarantees and the use of a Hamming window to reduce spectral leakage. We also improve fine-tuning with the use of an additional High-Frequency Adaptation module.
- We demonstrate that CKCNNs with kernel resolution adaptation outperform conventional CNNs for zero-shot, few-shot resolution transfer tasks.

2. Related work

2.1. Transfer Learning

Transfer learning methods use a model pre-trained on data from a similar or related task to improve performance on the main task [16, 32]. Transfer learning methods have demonstrated improved model performance when the main task has limited data available [30, 35]. A common approach is to use a model pre-trained on ImageNet [7] and only fine-tune the last fully connected layer. Transfer learning requires that data from a domain similar to the main task is available, and if the main task is not similar to a commonly used dataset such as ImageNet transfer, learning might not work as well [20, 35]. In contrast to transfer learning, we propose resolution transfer that uses low-resolution from the same domain to improve high-resolution performance.

2.2. Domain Adaptation

Domain adaptation [3, 33] is a sub-field of transfer learning that focuses on adapting a pre-trained model to the main domain by mapping from the source domain to the target domain [9, 18, 28] or by finding a shared domain that both domains can be mapped to [1, 2]. Similar to transfer learning, domain adaptation methods have been used to address limited labelled data in the target domain [22, 33]. The kernel resolution adaptation method that we propose could be seen as a form of domain adaptation, with the domain being the shift in image resolution.

2.3. Conventional Resolution Transfer

Conventional CNNs lose a sizeable portion of their accuracy when evaluated on a different resolution than the resolution seen during training (zero-shot resolution transfer) [23, 24]. Some approaches have been proposed to enable CNNs to generalise to unseen resolutions, particularly for

facial recognition. Some use super-resolution to upsample low-resolution images to high-resolution [21, 34], or focus on robustness to unseen resolution representations [11]. These methods focus on improving low-resolution accuracy to achieve good accuracies across resolutions, while our approach focusses on improving high-resolution accuracy by integrating low-resolution information when the amount of high-resolution images is limited.

2.4. Continuous Kernel CNNs

Continuous kernel CNNs (CKCNNs) construct arbitrary-size kernels by discretely sampling an underlying continuous function modelled by a kernel generator network [10, 12, 23–26, 29], while CNNs [19] assign a weight to each kernel coordinate. CKConv [26] uses an MLP to generate large convolutional kernels with a parameter count much lower than conventional CNNs. FlexConv [24] extends on CKConv and replaces the MLP with a Multiplicative Filter Network [8] and learns optimal kernel sizes during training with a learnable Gaussian Mask. Later work [25] adds depth-wise separable convolutions and a different initialization strategy. S4 [12] and its multi-dimensional follow-up S4ND [23] construct kernels by State Space Models. CKCNNs commonly use global kernels that span the entire input signal to improve classification accuracy while maintaining a low parameter count. Using global kernels has associated high inference costs, especially for high-resolution data. Due to high inference costs on high-resolution data, we move away from global kernels and use smaller kernels instead.

Previous work has shown that continuous kernels can be inferred at new resolutions due to their variable kernel size [26]. This makes CKCNNs naturally suited to deal with changes in image resolution. FlexConv [24] shows that aliasing artefacts negatively affect the accuracies of kernels inferred at new resolutions. Aliasing occurs when the kernel's frequency components are too high respective to the kernel generator's sampling rate. For anti-aliasing, FlexConv uses regularisation of kernel generators and S4ND [23] uses masking. We observe that the regularisation in FlexConv lacks an anti-aliasing guarantee which becomes a problem for small kernels. Therefore, we propose an alternative parameterization that removes the need for the regularisation of CKConv kernels and guarantees anti-aliasing.

3. Method

In this section, we cover the changes made to CKConv [26] to lower the inference costs of training CKCNNs on larger inputs, and improvements made to kernel resolution adaptation with small kernels.

3.1. Lowering costs of CKCNNs

We observe that training CKCNNs on high-resolution images is significantly slower than training CNNs. CKCNNs have higher inference costs than CNNs by nature due to the need to generate kernels before convolution. However, we identify the cause of the currently high inference costs of CKCNNs as the use of large kernels in combination with the lack of down-sampling of feature maps. In contrast, conventional CNN models mainly use small kernels and can use multiple down-sampling layers. Due to the lack of down-sampling in CKCNNs, feature maps remain at full resolution during inference. The exclusive use of high-resolution feature maps results in costly convolutions and high memory costs. Our approach to lower these costs is simple: we use smaller kernels and add pooling layers to down-sample feature maps, making our network architecture more similar to conventional models. We use 7x7 kernels, which in combination with pooling layers result in a significant speedup in training time, as we demonstrate in Section 4.3.

3.2. Kernel resolution adaptation

CKCNNs are naturally suited for resolution transfer tasks due to their ability to adapt to unseen resolutions. Existing work has shown that continuous kernels can be inferred at new resolutions [26]:

$$K_{new} \approx \frac{r_{new}(K_{old} - 1)}{r_{old}} + 1, \tag{1}$$

where K depicts the kernel size, assuming square kernels for two-dimensional and higher, and r depicts the image resolution in pixels. Eq. (1) only holds approximately due that the kernel size must be an integer in practice, while inferred kernel size is not always integer if the ratio between r_{old} and r_{new} is not integer.

3.3. Guaranteed Anti-Aliasing with Constrained CKConv (cCKConv)

We propose a new method for constraining the frequencies of CKConv [26] layers to prevent aliasing. Aliasing occurs for CKCNNs when the sampling rate of the kernel generator is insufficient to accurately capture all of the kernel's frequency components. With an insufficient sampling rate, distortions will occur when the kernel is sampled during kernel construction. These distortions cause substantial accuracy loss for CKCNNs with kernel resolution adaptation. The kernel's sampling rate is the number of samples (the kernel size) over the sampling period. For CKConv the sampling period is fixed to [-1, 1]. The Nyquist Frequency is the maximum frequency that the kernel can contain without aliasing occurring. FlexConv [24] derives the Nyquist Frequency f_{Nyq} for CKConv [26] kernels as:

$$f_{Nyq} = \frac{\mathrm{k} - 1}{4}, \tag{2}$$

where k is the kernel size. When the frequency constraint is met, kernel resolution adaptation can be done with minimal accuracy loss.

In FlexConv, MAGNet (Multiplicative Anisotropic Gabor Net) layers are used to construct the kernels for CKConv and are shown to achieve higher classification accuracies than MLPs. A MAGNet layer's output is:

$$f(x) = exp(-\frac{1}{2}(\gamma(x-\mu)^2)) \cdot sin(\mathbf{W}x + \mathbf{b}). \quad (3)$$

μ and γ construct a Gaussian window, which is applied to the output of a sine function and a linear layer with weights $\mathbf{W}$ and bias $\mathbf{b}$ to construct the 2D Gabor filter. The maximum frequency of a MAGNet layer is composed of the frequency terms of the Gaussian window f^{γ} and the sine function f^{sin}:

$$f_{max}^{MAGNet} = max_i(f_i^{\gamma} + f_i^{sin}) = \quad (4)$$

$$max_i(\frac{min(\gamma_{X,i}, \gamma_{Y,i})}{\pi} + max_j \frac{\mathbf{W}_{i,j}}{2\pi}). \quad (5)$$

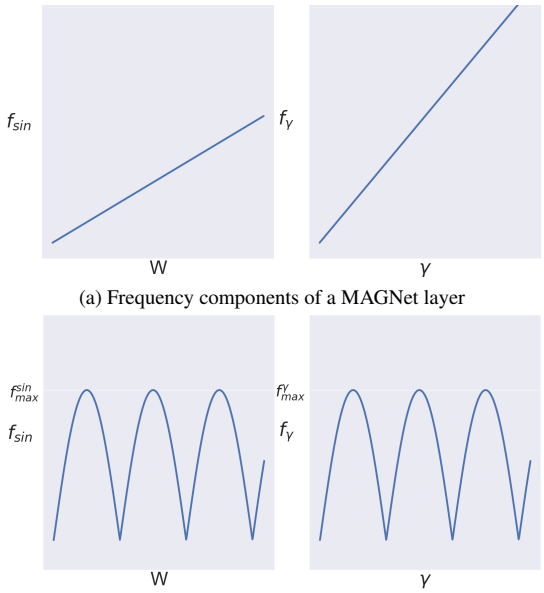
f^{γ} increases linearly with the magnitude of γ, and f^{sin} increases linearly with the magnitude of $\mathbf{W}$. FlexConv [24] uses regularisation of the parameters γ and $\mathbf{W}$ to steer the model towards kernels with frequencies below a set maximum frequency f_{max} (Eq. (2)) to avoid aliasing.

We observe that regularisation of γ and $\mathbf{W}$ introduces a trade-off between learning kernels with a maximum frequency and learning optimal kernels. There is no guarantee that learned kernels will have a maximum frequency below the Nyquist Frequency. The lack of an anti-aliasing guarantee becomes a relevant issue for smaller kernels, as the Nyquist Frequency decreases and aliasing becomes more likely. We propose a MAGNet parameterization that provides an anti-aliasing guarantee and removes the need for regularisation of kernel weights:

$$f(x) = exp(-\frac{1}{2}(\pi f_{max}^{\gamma} sin(\gamma)(x-\mu)^2)) \cdot \quad (6)$$
$$sin(2\pi f_{max}^{sin} sin(\mathbf{W})x + \mathbf{b}).$$

We apply a sine function to γ and $\mathbf{W}$, which is enough to constrain the frequency content of the MAGNet kernels. f^{γ} and f^{sin} no longer increase linearly but move periodically between a set minimum and maximum. We use our parameterization to constrain f^{γ} and f^{sin} by setting the amplitude of the sine functions to πf_{max}^{γ} and $2\pi f_{max}^{sin}$. The differences between MAGNet parameterizations in frequency as the parameters γ and $\mathbf{W}$ increase in magnitude are visualized in Figure 2. f_{max}^{γ} and f_{max}^{sin} can be set manually, and make up the maximum frequency of the MAGNet layer:

$$f_{max}^{MAGNet} = f_{max}^{\gamma} + f_{max}^{sin} \quad (7)$$



(a) Frequency components of a MAGNet layer

(b) Frequency components of our constrained MAGNet layer

Figure 2. Frequency components of a MAGNet layer and our alternative parameterization. The frequency components of a MAGNet layer grow linearly with the magnitude of the parameters whereas the frequency components of our alternative parameterization stay below a set maximum.

The maximum frequency of the MAGNet layer needs to be divided between the frequency effect of the Gaussian window and the frequency of the sine function. In our current implementation, this divide is initialized to 25% to the Gaussian window and 75% to the sine function. We base this initialization on results from preliminary experiments and the divide is set to be a learnable parameter such that each MAGNet layer can optimize its split during training.

3.4. Spectral Leakage

Spectral leakage occurs when the frequency content of a signal spreads to unintended frequencies, which causes drops in resolution transfer accuracy in a similar way to aliasing (Sec. 3.3). Previous work [31] has demonstrated that small kernels in CNNs are susceptible to spectral leakage. We find that in zero-shot resolution transfer settings, smaller CKConv kernels lose more accuracy than larger kernels and link this to spectral leakage. Both S4ND [23] and FlexConv [24] use large kernels, reducing the risk of spectral leakage. Furthermore, we observe that in FlexConv the Gaussian mask used to learn the kernel size also inadvertently acts as a windowing function that reduces spectral leakage. Consequently, we apply a Hamming window function [31] to the kernel before convolution to reduce spectral leakage and improve resolution transfer for small kernels.

Model	Accuracy$_{36x36}$	Zero-Shot Accuracy$_{72x72}$	Many-Shot Accuracy$_{72x72}$
CNN	76.67 ± 0.10	52.24 ± 0.10	$\mathbf{100.0 \pm 0.0}$
CNN + Bilinear Upsampling	76.67 ± 0.10	53.12 ± 0.91	$\mathbf{100.0 \pm 0.0}$
CKConv	$\mathbf{79.64} \pm 1.49$	62.83 ± 7.69	$\mathbf{100.0 \pm 0.0}$
cCKConv	77.07 ± 1.18	$\mathbf{76.27 \pm 0.67}$	76.30 ± 0.92
cCKConv + HFA	77.07 ± 1.18	$\mathbf{76.27 \pm 0.67}$	$\mathbf{100.0 \pm 0.0}$

Table 1. Accuracies on 2D sine wave classification on low-resolution, zero-shot high-resolution and many-shot fine-tuned high-resolution. Our constrained CKConv variant (cCKConv) performs best zero-shot but does not learn to classify the high-frequency signals without the High-Frequency Adaptation module.

3.5. High-Frequency Adaptation

The anti-aliasing constraint causes the kernels to only capture low-frequency features. Ideally, the model should be able to capture high-frequency features as well during fine-tuning to benefit from the introduced high-frequency features found in high-resolution images. In a similar fashion to Low-Rank-Adaptation [15], we add a second kernel generator module during the fine-tuning stage of our resolution transfer experiments to learn additional filters so that the model can adapt to the introduced high-frequency features. The two generated kernels are summed before convolution. The second kernel is initialized such that its contribution is minimal at first to maintain pre-trained accuracy, but the model can learn to increase the second kernel's contribution during fine-tuning.

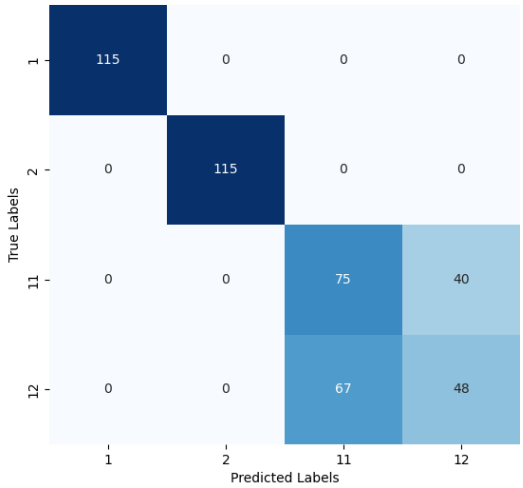


Figure 3. Confusion Matrix of the CNN model trained and evaluated on the lower resolution images of the toy experiment. The model fails to accurately classify the high-frequency classes.

4. Experiments

We first construct a toy setting in which we evaluate and compare the zero-shot resolution transfer of a CNN, CKConv with regularisation, and our cCKConv parameterization, as well as the effect of the High-Frequency Adaptation (HFA) module during fine-tuning. Next, we evaluate our changes outlined in Section 3 in a resolution transfer CIFAR10 [17] setting. Although the low-resolution setting of CIFAR10 does not fit all our contributions as these focus on high-resolution data, we use CIFAR10 to compare to existing work [23, 24]. Lastly, we evaluate in a hybrid dataset setting that reflects the realistic scenario of limited high-resolution data and pre-train on low-resolution data to demonstrate the data-efficiency of kernel resolution adaptation for resolution transfer as well as our changes to lower inference costs of CKCNNs. Whenever comparing CNN and CKConv, we use the same architecture and only differ in kernel type.

4.1. Experiment 1: Resolution transfer and High-Frequency Adaptation

We create an artificial image classification setting of several 2D sine waves, grouped by their frequency. A full description of how we construct our toy experiment setting can be found in Appendix A. When the image resolution is lowered, so is the sampling rate, which results in the loss of high-frequency components. The absence of high-frequency components in low-resolution images will make accurately classifying the high-frequency classes difficult. We compare conventional convolutional kernels with CKConv kernels in a two-layer convolutional model and evaluate their accuracy on low-resolution images, the high-resolution images before fine-tuning (zero-shot), and after fine-tuning (many-shot). We also include a variant of CNN where the kernels are up-sampled with bilinear interpolation before evaluation on the higher resolution to demonstrate that conventional kernels cannot easily change in size respective to resolution change and maintain accuracy, unlike CKCNNs.

Our results in Table 1 demonstrate that no model can classify the images without error on the low-resolution images and that all models can achieve zero error on the high-resolution images except for cCKConv without the

Model		#Params	Accuracy$_{16x16}$	Accuracy$_{32x32}$					Epoch time (s)
Type	Architecture			Zero-shot (0)	Few-shot (50)	Few-shot (250)	Many-shot (5000)	No pre-training (5000)	
Baselines									
CNN	FlexNet-16 [24]	904K	82.36 ± 0.2	62.43 ± 2.41	72.53 ± 0.87	78.99 ± 0.4	$\mathbf{89.94 \pm 0.13}$	89.51 ± 0.12	14
FlexConv	FlexNet-16	672K	$\mathbf{86.66 \pm 0.08}$	$\mathbf{84.51 \pm 0.96}$	$\mathbf{86.13 \pm 0.26}$	$\mathbf{86.59 \pm 0.30}$	88.94 ± 0.06	$\mathbf{91.61 \pm 0.14}$	111
S4ND	S4ND-ISO [23]	676K	81.36 ± 0.13	77.97 ± 0.15	80.35 ± 0.80	80.57 ± 0.39	85.93 ± 0.44	84.56 ± 0.84	110
CKConv	FlexNet-16	672K	83.77 ± 0.37	75.89 ± 1.22	82.27 ± 0.37	83.35 ± 0.42	85.28 ± 0.28	89.83 ± 0.18	95
CKConv	FlexNet-16$_{Pool}$	672K	82.53 ± 0.26	74.57 ± 2.21	81.54 ± 0.23	82.48 ± 0.62	84.74 ± 0.92	89.14 ± 0.28	94
Ours									
CKConv + Hamming	FlexNet-16	672K	85.04 ± 0.34	79.54 ± 0.62	84.28 ± 0.29	84.39 ± 0.24	86.49 ± 0.35	90.01 ± 0.17	96
CKConv + Hamming	FlexNet-16$_{Pool}$	672K	83.8 ± 0.23	80.40 ± 1.43	83.10 ± 0.20	83.94 ± 0.80	86.27 ± 0.58	89.20 ± 0.39	94
cCKConv	FlexNet-16	672K	82.46 ± 0.59	77.32 ± 2.10	82.2 ± 0.38	82.75 ± 0.37	83.61 ± 0.67	88.81 ± 0.05	100
cCKConv	FlexNet-16$_{Pool}$	672K	80.31 ± 0.51	76.47 ± 1.68	80.60 ± 0.20	79.78 ± 0.87	82.66 ± 0.4	87.49 ± 0.28	99
cCKConv + Hamming	FlexNet-16$_{Pool}$	672K	82.15 ± 0.08	79.22 ± 1.32	82.08 ± 0.32	82.10 ± 0.42	84.3 ± 0.28	87.96 ± 0.21	99
cCKConv + Hamming + HFA	FlexNet-16$_{Pool}$	672K	82.15 ± 0.08	79.22 ± 1.32	82.17 ± 0.16	82.48 ± 0.36	87.94 ± 0.24	87.96 ± 0.21	99

Table 2. Test accuracies on CIFAR10 on 16x16 images, zero-shot, few-shot, many-shot on 32x32 images as well as test accuracies when trained and evaluated on 32x32 images without pre-training. Epoch time is measured on a NVIDIA A40 during pre-training. All CKCNN models outperform CNNs on zero-shot and few-shot due to their kernel resolution adaptation.

High-Frequency Adaptation module. The fact that no model can achieve zero error on the low-resolution images is expected as the low-resolution images do not contain the high-frequency components necessary to accurately classify the high-frequency classes, while the high-resolution images do. We show the class-specific accuracies of the CNN model in Figure 3 to demonstrate that the model can separate the low-frequency classes but fails to distinguish the high-frequency classes well. In the zero-shot setting, we demonstrate the following: CNN models lose a sizeable portion of their accuracy, whereas CKConv can retain some of their accuracy and cCKConv retains nearly all of their accuracy. We demonstrate that, unlike CKCNNs, CNNs cannot simply up-sample their kernels and improve their zero-shot accuracy. CKConv with regularisation lacks an anti-aliasing guarantee which explains the worse zero-shot accuracy compared to cCKConv which does provide an anti-aliasing guarantee. However, cCKConv does not improve its accuracy during fine-tuning in the many-shot setting due to its constrained kernels that can not capture high-frequency features. With the addition of the High-Frequency Adaptation module, cCKConv can learn the high-frequency features and achieve zero error.

4.2. Experiment 2: CIFAR10

We evaluate several CKCNN variants and the conventional convolutional layer on CIFAR10 [17], as CIFAR10 has been used as a baseline for resolution transfer in previous works [23, 24]. We use the FlexNet-16 architecture as is used in FlexConv [24]. CIFAR10 uses low-resolution images exclusively and our changes to lower inference costs and improvements to fine-tuning are not necessarily as relevant as these are intended for high-resolution images that contain high-frequency features. Nevertheless, we use CIFAR10 to compare to existing baselines and demonstrate that CKCNNs with kernel resolution adaptation, including existing works, can achieve better zero-shot and few-shot resolution transfer performance than conventional CNNs.

We include several variants of CKConv [26] with MAGNet kernels [24] as well as a down-sized variant of S4ND for a fair comparison. Our configuration of the S4ND model is given in Appendix C. We evaluate the models on their test accuracies on the low resolution, as well as the zero-shot, few-shot, and many-shot accuracies on the high resolution. For the few-shot setting, we evaluate fine-tuning on a subset of CIFAR10, in this case 1% (50 samples per class), and 5% (250 samples per class). In the many-shot setting, we fine-tune on the full high-resolution dataset (5000 samples per class).

Our results are given in Table 2. All CKCNN models outperform CNNs significantly when evaluated zero-shot on the high-resolution data. However, CNNs quickly regain a large portion of their accuracy when fine-tuned with a limited amount of data in the few-shot setting. We observe that FlexConv loses less accuracy compared to CKConv. We attribute the lower loss of accuracy to the reduction of spectral leakage due to the use of larger kernels as well as the Gaussian mask used to learn the kernel size. We demonstrate that the addition of a Hamming window function decreases the amount of accuracy lost in zero-shot resolution transfer and consequently improves few-shot accuracy as well. The addition of our High-Frequency Adaptation model improves fine-tuning accuracy, especially many-shot.

FlexConv performs best on most metrics but has the highest epoch times together with S4ND due to the use of global kernels that span the entire image. Out of the models with 7x7 kernels, those that apply a Hamming window before convolution perform best on zero-shot and few-shot. Our constrained variant outperforms the regularised variant in zero-shot and few-shot, but with a Hamming window to reduce spectral leakage the performance between CKConv and cCKConv becomes more similar. The addition of pooling layers does not provide a notable speedup for

Model			#Params	Accuracy_{64x64}	$\text{Accuracy}_{256x256}$					Epoch time (s)
Type	Architecture	Pre-training			Zero-Shot	Few-Shot ($\sim$35)	Many-Shot ($\sim$700)	No Pre-training ($\sim$35)	No Pre-training ($\sim$700)	
Baselines										
CNN	FlexNet-16 [24]$_{Pool}$	SATMIX	0.90M	96.36 ± 0.22	59.00 ± 3.54	77.70 ± 2.58	96.07 ± 0.29	**66.03 ± 3.02**	**95.77 ± 0.52**	17
FlexConv	FlexNet-16	SATMIX	0.67M	96.64 ± 0.33	**76.57 ± 6.94**	-	-	57.8 ± 2.18	-	146
CKConv	FlexNet-16$_{Pool}$	SATMIX	0.67M	97.26 ± 0.09	24.20 ± 0.59	85.05 ± 1.40	95.87 ± 0.20	54.91 ± 1.39	95.18 ± 0.27	40
Ours										
cCKConv + Hamming	FlexNet-16$_{Pool}$	SATMIX	0.67M	96.69 ± 0.19	68.51 ± 7.02	83.42 ± 1.86	95.38 ± 0.33	59.02 ± 2.80	**95.83 ± 0.88**	41
cCKConv + Hamming + HFA	FlexNet-16$_{Pool}$	SATMIX	0.67M	96.69 ± 0.19	68.51 ± 7.02	86.52 ± 0.15	96.33 ± 0.48	59.02 ± 2.80	**95.83 ± 0.88**	41
Larger model										
CNN	FlexNet-28$_{Pool}$	SATMIX	16.42M	**97.48 ± 0.17**	60.83 ± 0.93	80.96 ± 0.78	96.41 ± 0.33	**64.44 ± 0.19**	**95.89 ± 0.13**	18
cCKConv + Hamming + HFA	FlexNet-28$_{Pool}$	SATMIX	11.18M	**97.36 ± 0.07**	68.65 ± 9.03	**89.72 ± 1.04**	96.41 ± 0.32	59.20 ± 3.35	93.23 ± 0.48	43
ResNet-18$_{7x7}$										
CNN	ResNet-18 [13]$_{7x7}$	SATMIX	60.0M	96.11 ± 0.17	60.99 ± 1.48	76.55 ± 3.25	**96.8 ± 0.11**	**66.57 ± 2.14**	94.76 ± 0.40	18
CNN	ResNet-18$_{7x7}$	ImageNet	60.0M	-	13.88 ± 1.99	80.13 ± 3.30	92.36 ± 0.54	-	-	-
cCKConv + Hamming + HFA	ResNet-18$_{7x7}$	SATMIX	39.32M	96.52 ± 0.13	63.23 ± 2.45	79.71 ± 0.63	94.88 ± 0.39	57.14 ± 2.66	93.23 ± 1.03	29

Table 3. Test accuracies on SATMIX for CNN and CKCNN. Epoch time is measured on a NVIDIA A40 during pre-training. Continuous kernels consistently outperform the conventional discrete kernels in zero-shot and few-shot due to kernel resolution adaptation. ImageNet pre-trained ResNet-18$_{7x7}$ performs similarly to CKConv ResNet-18$_{7x7}$ in few-shot resolution transfer, but CKCNNs with other architectures can outperform when pre-trained on low-resolution images.

training CKCNNs in CIFAR10. The lack of significant speedup makes sense, as CIFAR10 exclusively uses low-resolution images for which down-sampling does not remove significant inference costs. The difference in epoch times between the CNN and CKConv models is mostly due to the overhead of generating kernels. The kernel generation overhead depends on kernel size and becomes less relevant for high-resolution data. However, the difference in inference times between FlexConv, S4ND and CNNs will only become larger due to their use of global kernels, as inference costs of convolution increase massively with higher-resolution data.

4.3. Experiment 3: SATMIX

4.3.1 Setting: SATMIX

To evaluate resolution transfer in a practical setting with high-resolution images, we construct a hybrid dataset setting composed of two satellite datasets, which we refer to as SATMIX. SATMIX is composed of 6 overlapping classes between EUROSAT [14] and NWPU-RESISC45 [5]. EUROSAT contains 64x64 images and $\sim$3000 images per class. NWPU-RESISC45 contains 256x256 images and $\sim$700 images per class. The datasets have a similar domain and the transfer between the two datasets should be mostly in image resolution. We pre-train on SATMIX and fine-tune on the 6 corresponding classes in NWPU-RESISC45. We include a comparison between CNNs and CKCNNs with the FlexNet-16 [24] architecture, a larger FlexNet-28 architecture, and ResNet-18 [13] with 7x7 kernels including an ImageNet pre-trained model. For further details on the experiment configuration see Appendix D.

4.3.2 Results

Our results are shown in Table 3. Nearly all models' low-resolution accuracy is better than their high-resolution accuracy, which is unexpected as high-resolution images should contain more fine-grained details that improve accuracy. We hypothesize that the reason for this is the larger variety of meters per pixel in NWPU-RESISC45 images. We refer to the meters per pixel in satellite images as satellite resolution. In EUROSAT, every image has a fixed satellite resolution of 10m, but NWPU-RESISC45 has satellite resolutions ranging from 0.2m to 30m. Variety in satellite resolution makes NWPU-RESISC45 more difficult, which causes a larger gap between low and high-resolution accuracy. However, as we demonstrate in our few-shot results, models can quickly adapt to this variety in satellite resolution and improve their accuracy.

FlexConv [24] is significantly slower than the other models during pre-training. We were only able to complete pre-training as fine-tuning on the high-resolution data was too slow to consider viable. Training FlexConv on a single batch takes 3 minutes, and extrapolated to the entire dataset one epoch takes approximately 6 hours. We consider this too slow to be viable as this is roughly the time it takes to fully train the other models. In comparison to FlexConv, other CKCNN models are significantly faster as a result of smaller kernels and down-sampling layers. cCKConv models outperform the CNN models in zero-shot, and few-shot by a considerable margin. Due to the lack of an anti-aliasing guarantee for kernel resolution adaptation, the regularised CKConv model has the worst zero-shot accuracy. CNNs and CKCNNs perform roughly the same when fine-tuned with a large number of samples (many-shot). cCKConv benefits from the addition of our High-Frequency Adaptation module during fine-tuning, improving both few-shot and many-shot results.

The ImageNet pre-trained ResNet-18$_{7x7}$ performs considerably worse on zero-shot compared to its SATMIX pre-trained counterparts, but achieves better few-shot accuracy and worse many-shot accuracy. cCKConv ResNet-18$_{7x7}$ performs similarly to the ImageNet pre-

trained model in few-shot but performs significantly better in zero-shot and many-shot. However, our other CKCNN results demonstrate significantly better few-shot results with smaller models, which suggests that the ResNet-18 models were prone to over-fitting or that other architectures suit SATMIX better. The suggestion of ResNet-18_{7x7} models overfitting is further motivated by that the difference between many-shot and no pre-training is larger for these models than other models, which suggest that pre-training reduced overfitting. We argue that resolution transfer allows for the use of task-specific architectures without the need to pre-train on large general datasets such as ImageNet. Our results demonstrate that resolution transfer methods can significantly outperform transfer learning methods in zero-shot resolution transfer, and compete in few-shot and many-shot settings.

Pre-training on SATMIX did not have a significant effect on the accuracy for most models, as the many-shot accuracy is similar to simply training on the 6 classes of the NWPU-RESISC45 dataset. If the model is overfitting such as suggested for our ResNet-18_{7x7} models, pre-training with additional low-resolution data improves generalisation. With limited high-resolution data, pre-training significantly improves accuracy, with CKCNNs with kernel resolution adaptation performing best.

5. Discussion

We propose the use of Continuous Kernel CNNs (CKCNNs) with kernel resolution adaptation for data-efficient resolution transfer. We demonstrate that CKCNNs can adapt their kernel size respective to resolution change and achieve better zero-shot resolution transfer accuracy than conventional discrete CNNs. With smaller convolutional kernels and introducing pooling layers, we lower the high inference costs associated with current CKCNN work. We improve the zero-shot resolution transfer performance of CKCNNs with the use of an alternative parameterization that guarantees frequency constraints of kernels and apply a Hamming window function to reduce spectral leakage. We improve fine-tuning with the addition of a module to fine-tune specifically on high-frequency features introduced by the high-resolution images. We demonstrate that CKCNNs can use kernel resolution adaptation to achieve better few-shot results than conventional CNNs when pre-trained with low-resolution images.

The use of CKCNNs still has some limitations. CKCNNs have higher inference costs than CNNs due to the kernel generation overhead. Next to computational overhead, to apply kernel resolution adaptation the resolutions of all images must be known and if the ratio between low and high-resolution images is non-integer the kernel cannot always adapt its size optimally as the kernel size must be integer. The use of kernels smaller than 7x7 is common in current CNN models, but extremely small kernels are still at risk of spectral leakage significantly affecting kernel resolution adaptation performance. Lastly, the use of kernel generator networks adds hyperparameters to the already arduous task of model tuning.

We believe that continuous kernel CNNs could still be developed further. In our current implementation, increasing the maximum frequency of a constrained kernel requires increasing the sample amount, the kernel size, as the sampling period is fixed. Future work could look into adapting the sampling period rather than the kernel size by adjusting the sampling duration to fit the Nyquist Frequency. Additionally, we believe that the ability to constrain a CNN's kernel's frequency content could have relevant applications besides kernel resolution adaptation. Constrained kernels could lead to improvements in model generalisation and robustness with kernels constrained to be smooth, or avoid commonly occurring biases that negatively affect behaviour by constraining the kernel's range outside of these biases.

References

[1] Carlos J Becker, Christos M Christoudias, and Pascal Fua. Non-linear domain adaptation with boosting. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 4

[2] Alessandro Bergamo and Lorenzo Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. 4

[3] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July 2006. Association for Computational Linguistics. 4

[4] Dingding Cai, Ke Chen, Yanlin Qian, and Joni-Kristian Kämäräinen. Convolutional low-resolution fine-grained classification. *CoRR*, abs/1703.05393, 2017. 3

[5] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *CoRR*, abs/1703.00121, 2017. 9, 13

[6] M. Chevalier, N. Thome, M. Cord, J. Fournier, G. Henaff, and E. Dusch. Lr-cnn for fine-grained classification with varying resolution. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 3101–3105, 2015. 3

[7] 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*, pages 248–255. Ieee, 2009. 4

[8] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021. 5

[9] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Subspace alignment for domain adaptation. *CoRR*, abs/1409.5241, 2014. 4

[10] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data, 2020. 5

[11] Guangwei Gao, Yi Yu, Jian Yang, Guo-Jun Qi, and Meng Yang. Hierarchical deep cnn feature set-based representation learning for robust cross-resolution face recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2550–2560, May 2022. 5

[12] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *CoRR*, abs/2111.00396, 2021. 5

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 9, 13

[14] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. 08 2017. 9, 13

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. 7

[16] Rajdeep Kaur, Rakesh Kumar, and Meenu Gupta. Review on transfer learning for convolutional neural network. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 922–926, 2021. 4

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 7, 8

[18] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, page 1785–1792, USA, 2011. IEEE Computer Society. 4

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 5

[20] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2208–2217. PMLR, 06–11 Aug 2017. 4

[21] Priyank Makwana, Satish Kumar Singh, and Shiv Ram Dubey. Resolution invariant face recognition. In Massimo Tistarelli, Shiv Ram Dubey, Satish Kumar Singh, and Xiaoyi Jiang, editors, *Computer Vision and Machine Intelligence*, pages 733–745, Singapore, 2023. Springer Nature Singapore. 5

[22] Saeid Motiian, Quinn Jones, Seyed Mehdi Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. *CoRR*, abs/1711.02536, 2017. 4

[23] Eric Nguyen, Karan Goel, Albert Gu, Gordon W. Downs, Preey Shah, Tri Dao, Stephen A. Baccus, and Christopher Ré. S4nd: Modeling images and videos as multidimensional signals using state spaces, 2022. 3, 4, 5, 6, 7, 8, 12

[24] David W. Romero, Robert-Jan Bruintjes, Jakub M. Tomczak, Erik J. Bekkers, Mark Hoogendoorn, and Jan C. van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *CoRR*, abs/2110.08059, 2021. 3, 4, 5, 6, 7, 8, 9, 13

[25] David W Romero, David M Knigge, Albert Gu, Erik J Bekkers, Efstratios Gavves, Jakub M Tomczak, and Mark Hoogendoorn. Towards a general purpose cnn for long range dependencies in nd. *arXiv preprint arXiv:2206.03398*, 2022. 5

[26] David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. *CoRR*, abs/2102.02611, 2021. 5, 8

[27] Carl F. Sabottke and Bradley M. Spieler. The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, 2(1):e190015, 2020. PMID: 33937810. 3

[28] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 213–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 4

[29] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions, 2017. 5

[30] Srikanth Tammina. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. volume 9, page p9420, 10 2019. 4

[31] Nergis Tomen and Jan van Gemert. Spectral leakage and rethinking the kernel size in cnns, 2021. 6

[32] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications*, 01 2009. 4

[33] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018. 4

[34] Ling-Yi Xu and Zoran Gajic. Improved network for face recognition based on feature super resolution method, 2021. 5

[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. 4

Appendices

A. Toy Setting

We create a toy setting of 2D sinewaves to classify by their frequency. We use Python and matplotlib to create the plots. The equation of the 2D sine wave is as follows:

$$f(x, y) = sin(2\pi \cdot frequency \cdot x + 2\pi \cdot frequency \cdot y). \quad (8)$$

We have 4 frequency classes (1, 2, 11, 12 Hertz) with amplitude 1.0. We increase the number of samples by having 5 different phase shifts, 0%, 20%, 40%, 60%, and 80% of a full phase. We also increase the number of samples by changing the colormap used to plot the sine function. The 23 colormaps used are:

```
viridis, plasma, inferno, magma,
cividis,Greys, Purples, Blues,
Greens, Oranges, Reds,YlOrBr,
YlOrRd, OrRd, PuRd, RdPu,
BuPu,GnBu, PuBu, YlGnBu,
PuBuGn, BuGn, YlGn
```

The use of 23 colormaps and 5 different phase shifts results in 115 samples per frequency class.

We use a sampling density of 36 and measure the sine wave between X and Y values [-1, 1]. This results in 72x72 images. When the resolution is halved the images will be 36x36 pixels with a sampling density of 18. This sampling density is enough to accurately represent signals with a frequency up to 9 Hertz. This means high-frequency components of the classes belonging to 11 and 12 Hertz are lost on low-resolution. This explains why the models cannot perfectly classify the signals on the lower-resolution. The necessity of the high-frequency components for perfect classification also explain why the constrained CKConv model cannot reach zero error without the High-Frequency Adaptation (HFA) module during fine-tuning.

For the experiment we used a two layer CNN model which was sufficient to achieve zero error on the high-resolution images. Only kernel type (CNN, CKCNN) and fine-tuning (no up-sampling of CNN, bilinear up-sampling, no HFA, HFA) differ between experiments. Our training configuration:

```
learning rate: 0.01
batch size: 64
scheduler: cosine
scheduler warm-up epochs: 3
normalization: Batch Norm
optimizer: Adam
kernel size: 7
epochs: 25
fine-tune epochs: 25
```

B. CIFAR10

We train for 100 epochs and fine-tune for 50 epochs. When only trained on the high-resolution images, models were trained for 100 epochs. Our training configuration:

```
learning rate: 0.01
batch size: 64
scheduler: cosine
scheduler warm-up epochs: 5
normalization: Batch Norm
optimizer: Adam
kernel size: 7
epochs: 100
fine-tune epochs: 50
```

Each model except for S4ND used the same network architecture, only varying in their kernel type (Conv2d, CKConv), whether pooling layers used, and fine-tuning strategy. FlexConv and S4ND used global kernels rather than kernel size 7, corresponding to 17 during pre-training and 33 during fine-tuning.

C. S4ND configuration

We compared several configurations with a similar parameter count to our own models, and selected the best performing configuration. We ran our S4ND experiments with code from the authors: https://github.com/HazyResearch/state-spaces/tree/main/configs/experiment Our configuration lowers the model dimension width, and the number of channels in the hidden state of the kernel generator compared to the model in the paper [23].

```
defaults:
  - /pipeline: cifar-2d
  - /model: s4
  - override /model/layer: s4nd
  - override /scheduler: cosine_warmup
dataset:
  augment: false
loader:
  batch_size: 50
  img_size: 32
  train_resolution: 2
  eval_resolutions: [1, 2]
model:
  dropout: 0.1
  tie_dropout: true
  n_layers: 6
  d_model: 192
```

Model	#Params	Accuracy$_{16x16}$	Accuracy$_{32x32}$				Epoch time (s)
			Zero-shot (0)	Few-shot (50)	Few-shot (250)	Many-shot (5000)	
cCKConv$_{Pool}$ + Hamming + HFA$_{Frozen,FixedSum}$	672K	82.15 ± 0.08	79.22 ± 1.32	82.15 ± 0.18	81.94 ± 0.29	86.95 ± 0.33	148
cCKConv$_{Pool}$ + Hamming + HFA$_{Frozen,WeighedSum}$	672K	82.15 ± 0.08	79.22 ± 1.32	81.94 ± 0.44	81.90 ± 0.20	86.27 ± 0.16	148
cCKConv$_{Pool}$ + Hamming + HFA$_{Unfrozen,FixedSum}$	672K	82.15 ± 0.08	79.22 ± 1.32	82.07 ± 0.37	82.30 ± 0.66	88.44 ± 0.54	148
cCKConv$_{Pool}$ + Hamming + HFA$_{Unfrozen,WeighedSum}$	672K	82.15 ± 0.08	79.22 ± 1.32	82.17 ± 0.16	82.48 ± 0.36	87.94 ± 0.24	148

Table 4. High-Frequency Adaptation ablations on CIFAR10.

Model	#Params	Accuracy$_{64x64}$	Accuracy$_{256x256}$			Epoch time (s)
			Zero-Shot	Few-Shot ($\sim$35)	Many-Shot ($\sim$700)	
cCKConv$_{Pool}$ + Hamming + HFA$_{Frozen,FixedSum}$	0.67M	96.69 ± 0.19	68.51 ± 7.02	85.15 ± 1.03	96.09 ± 0.06	75
cCKConv$_{Pool}$ + Hamming + HFA$_{Frozen,WeightedSum}$	0.67M	96.69 ± 0.19	68.51 ± 7.02	84.02 ± 2.15	95.53 ± 0.64	75
cCKConv$_{Pool}$ + Hamming + HFA$_{Unfrozen,FixedSum}$	0.67M	96.69 ± 0.19	68.51 ± 7.02	85.29 ± 0.26	96.33 ± 0.27	75
cCKConv$_{Pool}$ + Hamming + HFA$_{Unfrozen,WeightedSum}$	0.67M	96.69 ± 0.19	68.51 ± 7.02	$\mathbf{86.52 \pm 0.15}$	96.33 ± 0.48	75

Table 5. High-Frequency Adaptation ablations on SATMIX.

```
  prenorm: true
  layer:
    init: diag-lin
    d_state: 48
    bidirectional: true
    final_act: glu
    n_ssm: 1
    dt_min: 0.1
    dt_max: 1.0
    l_max: [32, 32]
    bandlimit: 0.2
optimizer:
  lr: 0.01
  weight_decay: 0.03
trainer:
  max_epochs: 100
scheduler:
  num_warmup_steps: 900
  num_training_steps: 90000
train:
  seed: 2222
```

D. SATMIX

We construct SATMIX by combining the following classes from EUROSAT [14] and NWPU-RESISC45 [5]:

SATMIX Class	EUROSAT Class(es)	NWPU-RESIC45 Class(es)
Agricultural	Annual Crop, Permanent Crop	Rectangular farmland
Forest	Forest	Forest
Highway	Highway	Freeway
Industrial	Industrial	Industrial Area
Residentual	Residentual	Dense residentual, sparse residentual
River	River	River

Table 6. Classes in SATMIX and their corresponding class(es) in EUROSat and NWPU-RESIS45.

We use a 70/30 split for training/validation. During pre-training, models were trained for 100 epochs and fine-tuned for 100 epochs. Our training configuration:

```
learning rate: 0.01
batch size: 64
fine-tune batch size: 32
scheduler: cosine
scheduler warm-up epochs: 5
normalization: Batch Norm
optimizer: Adam
kernel size: 7
epochs: 100
fine-tune epochs: 100
```

When comparing models, we use the same architecture and only differ in kernel type and fine-tuning strategy. We use the FlexNet-16 architecture from FlexConv [24], and our larger models use our architecture FlexNet-28 which have 13 blocks rather than 7, 64 hidden channels rather than 24, and a different block width distribution: [64, 3, 96, 3, 128, 5, 160, 2] rather than: [24, 2, 36, 3, 48, 2]. We also use a ResNet-18 [13] with 7x7 kernels rather than 3x3 kernels.

E. High-Frequency Adaptation ablations

We perform ablations on our High-Frequency Adaptation (HFA) module. Our results are in Tab. 4 and Tab. 5. We compare several variants that differ in whether or not we freeze the weights of the pre-trained kernels, and to use a fixed sum or learnable weighted sum. Our results show that fine-tuning the pre-trained weights as well improves performance, and generally weighted sum performs better than fixed sum.

3 Part 3: Supplemental Materials

Deep learning & Convolutional Neural Networks (CNNs)

Deep learning

Deep learning is a branch of machine learning. In general, machine learning tries to teach machines how to accomplish tasks by providing training them with data. Deep learning uses artificial neural networks (ANNs) to learn representations of data to perform a specific task such as classification of images. Artificial neural networks are composed of many small "neuron" units that take an input and apply a function to it before outputting it to the next layer of the network. Neural networks typically learn, or train, by propagating the input through their network, and then comparing the output of the network to the input's paired desired output label. Using this comparison a loss value is formulated as a result of each of the network's weights (learnable parameters) and by a process called back-propagation each weight is updated with as objective to reach a minimum total loss. Some deep learning task have more or less data available than other tasks as input data can be hard to require, and labeling can be expensive or require expert knowledge. Deep neural networks have many layers of neurons and possibly millions of learnable parameters. However, the larger the amount of parameters in the network, the more data is typically required to train the model and achieve good performance.

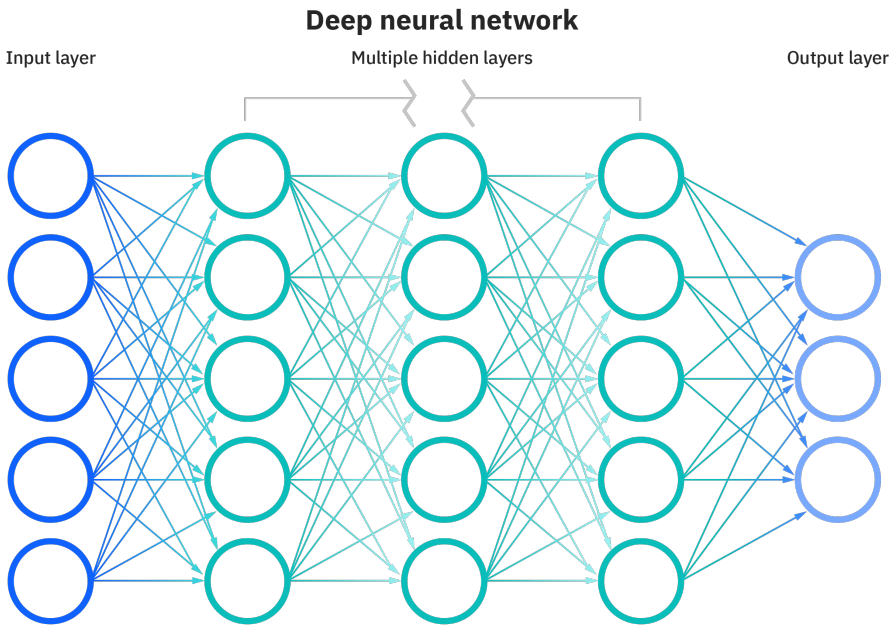


Figure 1: A (Fully Connected) Neural Network architecture. The input layer consists of neurons that take in the input, their output is processed by several hidden layers before the output layer outputs the model output. Source: https://www.ibm.com/topics/neural-networks

The hypothesis of the thesis is that a specific parameterization of a neural network could be used to use the weights trained on low-resolution image data and perform better on high-resolution data than the current conventional method. The reason this parameterization works better is because it can efficiently adapt to the high-resolution data without fine-tuning, and use this as a better kick-off for fine-tuning.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specific type of artificial neural network. CNNs are usually used for image processing with deep learning. Fully connected networks would require too many parameters as images contain too many input values. CNNs learn convolutional kernels that act as filters to extract features that it can pass to fully-connected layers at the last layer of the network. A convolutional kernel is a matrix of weights (learnable parameters) that acts as sliding window. This way the model can reuse parameters rather than assign a parameter for each pixel in the image. The sliding window is moved across the two-dimensional image,

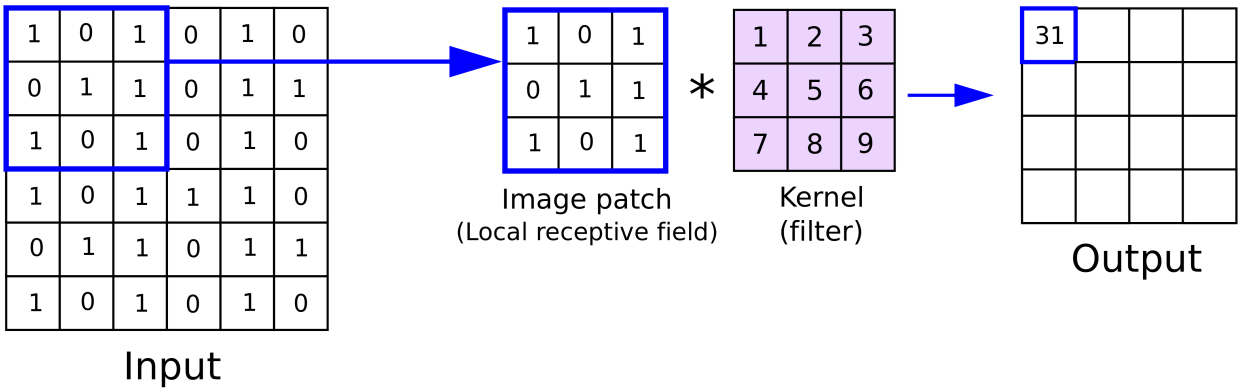


Figure 2: An example of convolution given the two-dimensional input and the kernel. The kernel covers a section of the image (local receptive field) and performs convolution over this area, the result corresponding to one value in the input. Source: https://anhreynolds.com/blogs/cnn.html

resulting in a two-dimensional feature map. Each value in the kernel is multiplied by the respective value in the input and the sum of these values corresponds to one value in the feature map. A single convolutional layer can have many kernels, to account for input-dimensions and the desired output dimensionality. This means each kernel acts as a filter for a specific feature.

Continuous Kernel Convolutional Neural Networks (CKConv)

Continuous Kernel Convolutional Neural Networks (CKConv, or CKCNNs) are a variant of the conventional Convolutional Neural Network (CNN). Conventional CNNs assign a learnable parameter (weight) to each value in the convolutional kernel. Due to this, small kernels are preferred as larger kernels would require too many learnable parameters, which could affect model learning negatively. Continuous kernel models construct their kernels differently. The model has a smaller kernel generator neural network, that learns a continuous function which is discretely sampled. Thus the network learns an underlying function rather than a kernel directly. The kernel can be constructed by sampling the underlying function, and thus the kernel size does not affect the amount of weights needed. This can be used to construct CNN models with very large kernels and still maintain a low amount of learnable parameters. An alternative application of this is that the kernel size is arbitrary, and can be changed at any time during training the network. In the thesis, this is used to be able to adapt to changes in image resolution better than conventional CNNs can, and train on low-resolution data before adapting to high-resolution data.