FCNN: Fourier Convolutional Neural Networks

Harry Pratt(✉), Bryan Williams, Frans Coenen, and Yalin Zheng

University of Liverpool, Liverpool L69 3BX, UK
{sghpratt,bryan,coenen,yzheng}@liverpool.ac.uk

Abstract. The Fourier domain is used in computer vision and machine learning as image analysis tasks in the Fourier domain are analogous to spatial domain methods but are achieved using different operations. Convolutional Neural Networks (CNNs) use machine learning to achieve state-of-the-art results with respect to many computer vision tasks. One of the main limiting aspects of CNNs is the computational cost of updating a large number of convolution parameters. Further, in the spatial domain, larger images take exponentially longer than smaller image to train on CNNs due to the operations involved in convolution methods. Consequently, CNNs are often not a viable solution for large image computer vision tasks. In this paper a Fourier Convolution Neural Network (FCNN) is proposed whereby training is conducted entirely within the Fourier domain. The advantage offered is that there is a significant speed up in training time without loss of effectiveness. Using the proposed approach larger images can therefore be processed within viable computation time. The FCNN is fully described and evaluated. The evaluation was conducted using the benchmark Cifar10 and MNIST datasets, and a bespoke fundus retina image dataset. The results demonstrate that convolution in the Fourier domain gives a significant speed up without adversely affecting accuracy. For simplicity the proposed FCNN concept is presented in the context of a basic CNN architecture, however, the FCNN concept has the potential to improve the speed of any neural network system involving convolution.

1 Introduction

Convolutional Neural Networks (CNNs) [1] are a popular, state-of-the-art, deep learning approach to computer vision with a wide range of application in domains where data can be represented in terms of three dimensional matrices. For example, in the case of image and video analysis. Historically, CNNs were first applied to image data in the context of handwriting recognition [2]. Since then the viability of CNNs, and deep learning in general, has been facilitated, alongside theoretical improvements, by significant recent advancements in the availability of processing power. For example, Graphics Processing Units (GPUs) allow us to deal with the heavy computation required by convolution.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-319-71249-9_47) contains supplementary material, which is available to authorized users.

M. Ceci et al. (Eds.): ECML PKDD 2017, Part I, LNAI 10534, pp. 786–798, 2017.
https://doi.org/10.1007/978-3-319-71249-9_47

However, there are increasingly larger datasets to which we wish to apply deep learning to [3] and, in the case of deep learning, a growing desire to increase the depth of the networks used in order to achieve better results [4,5]. This not only increases memory utilisation requirements, but also computational complexity. In the case of CNNs, the most computationally expensive element is the calculation of the spatial convolutions. The convolution is typically conducted using a traditional sliding window approach across the data matrix together with the application of a kernel function of some kind [6]. However, this convolution is computationally expensive, which in turn means that CNNs are often not viable for large image computer vision tasks. To address this issue, this paper proposes the idea of a using the Fourier domain. More specifically this paper proposes the Fourier Convolution Neural Network (FCNN) whereby training is conducted entirely in the Fourier domain. The advantage offered is that there is a significant speed up in training time without loss of effectiveness. Using FCNN images are processed and represented using the Fourier domain to which a convolution mechanism is applied in a manner similar to that used in the context of more traditional CNN techniques. The proposed approach offers the advantage that it reduces the complexity, especially in the context of larger images, and consequently provides for significant increase in network efficiency.

The underlying intuition given by the Convolution Theorem which states that for two functions κ and u, we have

$$\mathcal{F}(\kappa * u) = \mathcal{F}(\kappa) \odot \mathcal{F}(u) \tag{1}$$

where $\mathcal{F}$ denotes the Fourier transform, $*$ denotes convolution and $\odot$ denotes the Hadamard Pointwise Product. This allows for convolution to be calculated more efficiently using Fast Fourier Transforms (FFTs). Since convolution corresponds to the Hadamard product in the Fourier domain and given the efficiency of the Fourier transform, this method involves significantly fewer computational operations than when using the sliding kernel spatial method, and is therefore much faster [7]. Working in the Fourier domain is less intuitive as we cannot visualise the filters learned by our Fourier convolution; this is a common problem with CNN techniques and is beyond the scope of this paper. While the Fourier domain is frequently used in the context of image processing and analysis [8–10], there has been little work directed at adopting the Fourier domain with respect to CNNs. Although FFTs, such as the Cooley-Tukey algorithm [11], have been applied in the context of neural networks for image [12] and time series [13] analysis. These applications date from the embryonic stage of CNNs and, at that time, the improvement was minimal.

The concept of using the Fourier domain for CNN operations has been previously proposed [7,14,15]. In both [7,14] the speed-up of convolution in the Fourier domain was demonstrated. Down-sampling within the Fourier domain was used in [15] where the ability to retain more spatial information and obtain faster convergence was demonstrated. However, the process proposed in [7,14,15] involved interchanges between the Fourier and spatial domains at both the training and testing stages which added significant complexity. The FFT required is

the computationally intensive part of the process. FFTs, and inverse FFTs, needed to be applied for each convolution; thus giving rise to an undesired computational overhead. In the case of the proposed FCNN the data is converted to the Fourier domain before the process starts, and remains in the Fourier domain; no inverse FFTs are required at any point.

Instead of defining spatial kernel functions, which must then be transformed to the Fourier domain, as in the case of [7], using the proposed FCNN, a bespoke Fourier convolution mechanism is also proposed whereby convolution kernels are initialised in the Fourier domain. This method saves computation time during both the training and utilisation. Pooling in the Fourier domain is implemented in a similar fashion to that presented in [15] with truncation in the Fourier domain. This is not only more efficient than max-pooling, but can achieve better results [15]. The other layers implemented within the FCNN are dense layers and dropout. These Fourier layers are analogous to the equivalent spatial layers. Dropout randomly drops nodes within our network at a probability of p to stop over-fitting. This applies in the Fourier domain as it does in the spatial domain. Likewise, dense layers for learning abstract links within convolved image data operates with respect to Fourier data in the same manner as for spatial data.

The layout of the rest of the paper is as follows. In Sect. 2, we present our method of implementation of the specific layers that constitute our FCNNs, in Sect. 3 we present our experimental results. In Sects. 4 and 5 we present a discussion together with conclusions concerning abilities of the FCNN.

2 The Fourier Convolution Neural Network (FCNN) Approach

The FCNN was implemented using the deep learning frameworks *Keras* [16] and *Theano* [17]. *Theano* is the machine learning backend of *Keras*. This backend was used to code the Fourier layers. The Theano FFT function *Theano* was used to convert our training and test data. The *Theano* FFT function is a tensor representation of the multi-dimensional Cooley-Tukey algorithm. This function is the n-dimensional discrete Fourier transform over any number of axes in an m-dimensional array by using FFT. The multi-dimensional discrete Fourier transform used is defined as:

$$A_{kl} = \sum_{\ell_1=0}^{m-1} \sum_{\ell_2=0}^{n-1} a_{\ell_1 \ell_2} e^{-2\pi i \left(\frac{\ell_1}{m} + \frac{\ell_2}{n}\right)} \tag{2}$$

where the image is of size $m \times n$. The comparative methods of spatial convolution and max-pooling used throughout this paper relate to *Keras* and *Theano*'s implementations. To demonstrate the ability of the FCNNs implementation of all the core CNN layers in the Fourier domain we use the network architectures shown in supplementary.

The well used network architecture from AlexNet [1] was adopted because it provides a simple baseline network structure to compare the results of our equivalent Fourier and spatial CNNs on the MNIST [18] and Cifar10 datasets [19].

The MNIST dataset contains 60,000 grey scale images, 50,000 for training and 10,000 for testing, of hand written numeric digits in the form of 28×28 pixel images, giving a 10 class classification problem. The Cifar10 [19] dataset contains 60,000, 32×32 pixel, colour images containing 10 classes. These datasets are regularly used for standard CNN baseline comparison [4,20]. Experiments were also conducted using a large fundus image Kaggle data set [3]. This dataset comprised 80,000 RGB fundus images, of around 3M pixels per image, taken from the US diabetic screening process. The images are labelled using five classes describing level of diabetic retinopathy. These images are currently down-sampled during training using established CNN techniques because of the size of the images; this seems undesirable.

2.1 Fourier Convolution Layer

In traditional CNNs discrete convolutions between the images $\mathbf{u}^j$ and kernel functions κ^i are carried out using the sliding window approach. That is, a window the size of the kernel matrix is moved across the image. The convolution is computed as the sum of the Hadamard product $\odot$ of the image patch with the kernel:

$$\mathbf{z}^{i,j}_{k_1,k_2} = \sum_{\ell_1=\lfloor -m_\kappa/2 \rfloor}^{\lfloor m_\kappa/2 \rfloor} \sum_{\ell_2=\lfloor -n_\kappa/2 \rfloor}^{\lfloor n_\kappa/2 \rfloor} \kappa^i_{\ell_1,\ell_2} \odot \mathbf{u}^j_{k_1-\ell_1,k_2-\ell_2} \tag{3}$$

which results in an $(m_\mathbf{u} - m_\kappa) \times (n_\mathbf{u} - n_\kappa)$ image $\mathbf{z}$ since the image is usually re-sized to avoid including boundary artefacts in calculations. At each point (k_1, k_2), there are $m_k n_k$ operations required and so $(m_\mathbf{u} - m_{\kappa+1})(n_\mathbf{u} - n_{\kappa+1}) m_k n_k$ operations are needed for a single convolution.

We intend to replace, in the first instance, the sliding window approach with the Fourier transform using the discrete analogue of the convolution theorem:

$$\mathcal{F}(\kappa * \mathbf{u}) = \mathcal{F}(\kappa) \odot \mathcal{F}(\mathbf{u}) \tag{4}$$

where $\mathcal{F}$ denotes the two dimensional discrete Fourier transform:

$$\tilde{\mathbf{u}}_{i_1,i_2} = \sum_{j_1=1}^{m_\mathbf{u}} \sum_{j_2=1}^{n_\mathbf{u}} e^{-2\imath\pi\left(\frac{i_1 j_1 n_\mathbf{u} + i_2 j_2 m_\mathbf{u}}{m_\mathbf{u} n_\mathbf{u}}\right)} \mathbf{u}_{j_1,j_2} \tag{5}$$

The computation of the discrete Fourier transform for an $n \times n$ image $\mathbf{u}$ involves n^2 multiplications and $n(n-1)$ additions, but this can be reduced considerably using an FFT algorithm, such as Cooley-Tukey [11] which can compute the Direct Fourier Transform (DFT) with $n/2 \log_2 n$ multiplications and $n \log_2 n$ additions. This gives an overall improvement from the $O(n^2)$ operations required to calculate the DFT directly to $O(n \log n)$ for the FFT.

Thus, for a convolutional layer which has N^κ kernels κ^i in a network training $N^\mathbf{u}$ images $\mathbf{u}^j$, the output is the set $\mathbf{z}^{i,j} = \kappa^i * \mathbf{u}^j$ where $*$ denotes convolution. The algorithm is then:

1. $\tilde{\kappa}^i = \mathcal{F}\left(\kappa^i\right),\ i = 1, \ldots, N^{\kappa}$
2. $\tilde{\mathbf{u}}^i = \mathcal{F}\left(\mathbf{u}^i\right),\ i = 1, \ldots, N^{\mathbf{u}}$
3. $\tilde{\mathbf{z}}^{i,j} = \tilde{\kappa}^i \odot \tilde{\mathbf{u}}^j,\ i = 1, \ldots, N^{\kappa},\ j = 1, \ldots, m^{\mathbf{u}}$
4. $\mathbf{z}^{i,j} = \mathcal{F}^{-1}\left(\tilde{\mathbf{z}}^{i,j}\right),\ i = 1, \ldots, N^{\kappa},\ j = 1, \ldots, N^{\mathbf{u}}$

This decrease in the number of operations gives an increasing relative speed-up for larger images. This is of particular relevance given that larger computer vision (image) datasets are increasingly becoming available [3].

With respect to the proposed FCNN the N^k complex Fourier kernels are initialised using glorot initialisation [21]. The parameter n is equivalent to the number of kernel filters in the spatial network. Glorot initialisation was adopted because it is more efficient than doing FFT transformations of spatial kernels as this would require lots of FFTs during training to update the numerous convolution kernels. The weights for our Fourier convolution layer are defined as our initialised Fourier kernels. Hence, the Fourier kernels are trainable parameters optimised during learning, using back propagation, to find the best Fourier filters for the classification task with no FFT transformations relating to the convolution kernels required. Another benefit of Fourier convolutions is not only the speed of the convolutions, but that we can perform pooling during the convolution phase in order to save more computation cost.

A novel element of our convolution kernels is that, because they remain in the Fourier domain throughout, they have the ability to learn the equivalent of arbitrarily large spatial kernels limited only by initial image size. The image size is significantly larger than the size selected by spatial kernels. That is, our Fourier kernels which match the image size can learn a good representation of a 3×3 spatial kernel or a 5×5 spatial kernel depending on what aids learning the most. This is a general enhancement of kernel learning in neural networks as most networks typically learn kernels of a fixed size, reducing the ability of the network to learn the spatial kernel of the optimal size. In the Fourier domain, we can train to find not only the optimal spatial kernel of a given size but the optimal spatial kernel size and the optimal spatial kernel itself.

2.2 Fourier Pooling Layer

In the Fourier domain, the image data is distributed in a differ manner to the spatial. This allows us to reduce the data size by the same amount that it would be reduced by in the spatial domain but retain more information. High frequency data is found towards the centre of a Fourier matrix and low frequency towards the boundaries. Therefore, we truncate the boundaries of the matrices as the high frequency Fourier data contains more of the spatial information that we wish to retain. Our Fourier pooling layer shown in Fig. 1, operates as follows. Given a complex 3 dimensional tensors of $X \times Y \times Z$ dimensions, and AN arbitrary pool_size variable relating to the amount of data we wish to retain. For $x \in X$,:

$$x_{y\,\min} = (0.5 - \frac{\text{pool\_size}}{2}) \times Y, \quad x_{y\,\max} = (0.5 + \frac{\text{pool\_size}}{2}) \times Y \tag{6}$$

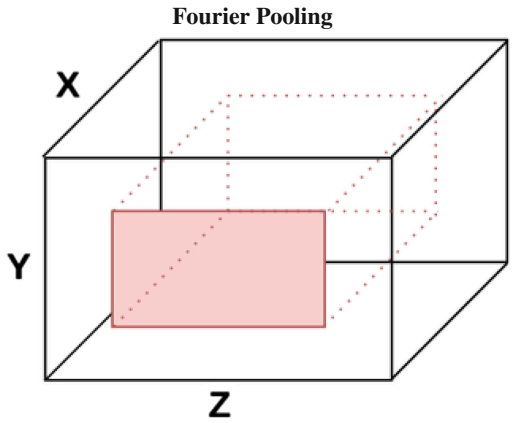

Fig. 1. Our layer initially contains an $X \times Y \times Z$ voxel. The truncation runs through the x-axis of the Fourier data (thus truncating the Y and Z axis).

$$x_{z\,\min} = (0.5 - \frac{\text{pool\_size}}{2}) \times Z, \quad x_{z\,\max} = (0.5 + \frac{\text{pool\_size}}{2}) \times Z \tag{7}$$

This method provides a straightforward Fourier pooling layer for our FCNN. It has a minimal number of computation operations for the GPU to carry out during training.

The equivalent method in the spatial context is max-pooling, which takes the maximum value in a $k \times k$ window where k is a chosen parameter. For example if $k = 2$, max-pooling reduces the data size by a quarter by taking the maximum value in the 2×2 matrices across the whole data. Similarly, in our Fourier pooling we would take pool_size $= 0.25$ which, using Eqs. 6 and 7, gives us:

$$x_{y\ \min} = 0.375 \times Y, x_{y\ \max} = 0.625 \times Y \tag{8}$$

$$x_{z\ \min} = 0.375 \times Z, x_{z\ \max} = 0.625 \times Z \tag{9}$$

which also reduces our data by a quarter.

3 Evaluation

The evaluation was conducted using an Nvidia K40c GPU that contains 2880 CUDA cores and comes with the Nvidia CUDA Deep Neural Network library (cuDNN) for GPU learning. For the evaluation both the computation time and the accuracy of the layers in the spatial and Fourier domains was compared. The FCNN and its spatial counterpart were trained using the 3 datasets introduced above: MNIST, Cifar10 and Kaggle fundus images. Each dataset was used to evaluate different aspects of the proposed FCNN. The MNIST dataset allows us to compare high-level accuracy while demonstrating the speed up of doing convolutions in the Fourier domain. The Cifar10 dataset was used to show that

the FCNN can learn a more complicated classification task to the same degree as a spatial CNN with the same number of filters. The results are presented below in terms of speed, accuracy and propagation loss. Finally, the large fundus Kaggle dataset was used to show that the FCNN is better suited to dealing with larger images, than spatial CNNs, because of the nature of the Fourier convolutions.

Table 1. Computation time for the convolution of a single images of varying size, using both Fourier and spatial convolution layers.

Size	FourierConv	SpatialConv	Ratio increase
2^{10}	5×10^{-2}	N/A	N/A
2^{9}	1×10^{-2}	N/A	N/A
2^{8}	2.67×10^{-3}	1.48×10^{-1}	55.43
2^{7}	7.74×10^{-4}	8.4×10^{-2}	10.85
2^{6}	2.85×10^{-4}	1.74×10^{-3}	6.10
2^{5}	1.78×10^{-4}	2.51×10^{-4}	1.41
2^{4}	1.36×10^{-4}	1.56×10^{-4}	1.14

3.1 Fourier Convolution

The small kernels used in neural networks mean that when training on larger images the amount of memory required to store all the convolution kernels on the GPU for parallel training is no longer viable. Using the Nvidia K40c GPU and a spatial convolution with 3×3 kernels the feed forward process of our network architecture cannot run a batch of images once image size approaches 2^9. The proposed Fourier convolution mechanism requires less computational memory when running in parallel. The memory capacity is not reached using the Fourier convolution mechanism until images of a size four times greater to the maximum size using the spatial domain are arrived at. This is due to the operational memory required for spatial convolution compared to the Fourier convolution.

The FCNN is able to train much larger images of the same batch size because the kernels are initialised in the Fourier domain, we initialise a complex matrix with the size matching the image size. Our convolutions are matrix multiplications and we are not required to pass across the image in a sliding window fashion, where extra storage is needed. The only storage we require is for the Fourier kernels, which are the same size as the images.

Table 1 presents a comparison of computation times, using Fourier and spatial convolution, for a sequence of single images of increasing size. From the table it can been seen that the computation time for a small images ($2^4 \times 2^4$ pixels) is similar for spatial and Fourier data in both cases. However, as the image size increases, the spatial convolution starts to become exponentially more time-consuming whereas the Fourier convolution scales at a much slower rate and allows convolution with respect to a much larger image size.

3.2 Fourier Pooling

Table 2 gives a comparison of the computation time, required to process a sequences of images of increasing size using, using the proposed Fourier pooling method in comparison with Max-pooling and Down-sampling. Fourier pooling is similar in terms of computational time to the max-pooling method which is the most basic down-sampling technique. This speed increase is for the same reason as the increase in convolution speed. Max-pooling requires access to smaller matrices within the data and takes the maximum value. On the other hand, in the Fourier domain, we can simply truncate in manner such that spatial information throughout the whole image is retained.

Table 2. Computation time for pooling an image of the given size using: (i) Down-sampling, (ii) Max pooling and (iii) Fourier pooling.

Size	Down-sampling	Max-pooling	Fourier pooling
2^{12}	2.77e-2	9.01	9.42e-2
2^{11}	7.93e-3	2.07	2.44e-2
2^{10}	2.19e-3	4.96e-1	5.30e-3
2^{9}	2.33e-4	1.26e-1	5.27e-4
2^{8}	2.70e-5	3.14e-2	1.01e-5
2^{7}	1.73e-5	6.80e-3	3.20e-6
2^{6}	3.67e-6	1.65e-3	5.29e-6
2^{5}	2.71e-6	3.82e-4	6.03e-6
2^{4}	2.46e-6	8.55e-5	5.35e-6

Figure 2 shows a comparison of pooling using down sampling, max pooling and Fourier pooling. In the figure the images in each image subsequent to the top row were reduced to half the size of the previous row and then up-scaled to the original image size for down-sampling and max-pooling. For Fourier pooling, the Fourier signal was embedded into a zero matrix of the same size as the original image and the Fourier transform is presented. Figure 3 shows how the Fourier pooling retains more spatial information as the best result in terms of visual acuity retained during pooling using mean squared error is the Fourier pooled image. All output images are the same size, but the Fourier retains more information. From the figures it can be seen that the Fourier pooling retains more spatial information than on the case of max-pooling when down-sampling the data by the same factor. This is because of the nature of the Fourier domain, the spatial information of the data is not contained in one specific point.

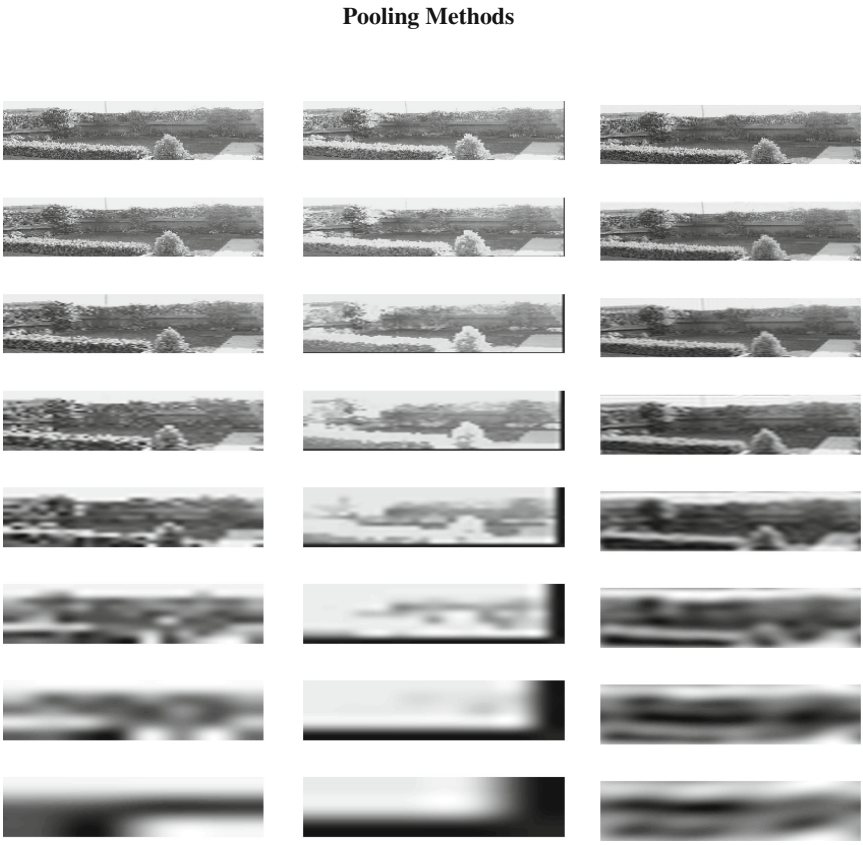

Fig. 2. Comparison of pooling using: (i) down-sampling (col. 1), (ii) max-pooling (col. 2) and (iii) Fourier pooling (col. 3).

3.3 Network Training

The baseline network is trained on both the MNIST and Cifar10 datasets to compare networks. Training was done using the categorical cross-entropy loss function and optimised using the rmsprop algorithm. The results are presented in Figs. 4 and 5 using network one. The fundus training was carried out on network two and epoch speeds were recorded see Table 3. The accuracy achieved on the MNIST and Cifar10 test sets using the FCNN is only marginally below the spatial CNN but the results are achieved with a significant speed up. The MNIST training was twice as fast on the FCNN in comparison the spatial CNN and the Cifar10 dataset was trained in 6 times the speed. This is due to the Cifar dataset containing slightly larger images than MNIST and demonstrates how our FCNN scales better to large images.

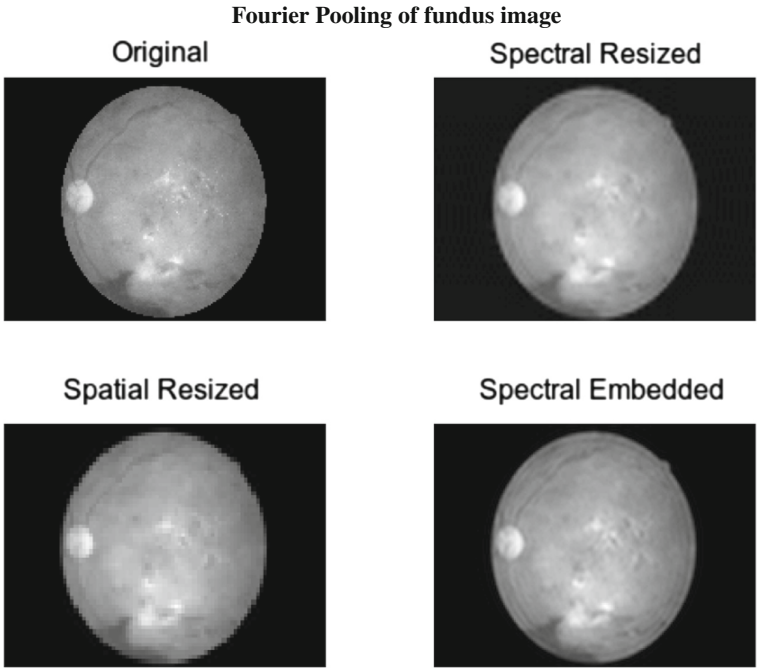

Fig. 3. (Top-left) Original fundus image, (Bottom-left) normal max-pooling and then resizing to original size; (Top-right) Fourier pooling, back to spatial domain and resize to original size; (Bottom-right) Fourier pooling, embed in a zero matrix and convert back to spatial

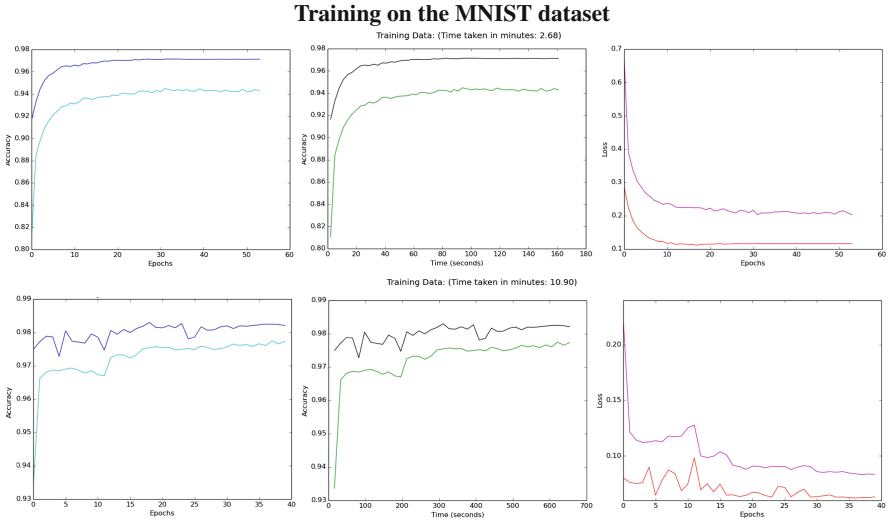

Fig. 4. (Top) FCNN (Bottom) Spatial CNN. Dark blue, black and red are validation values, lighter colours are training values. (Color figure online)

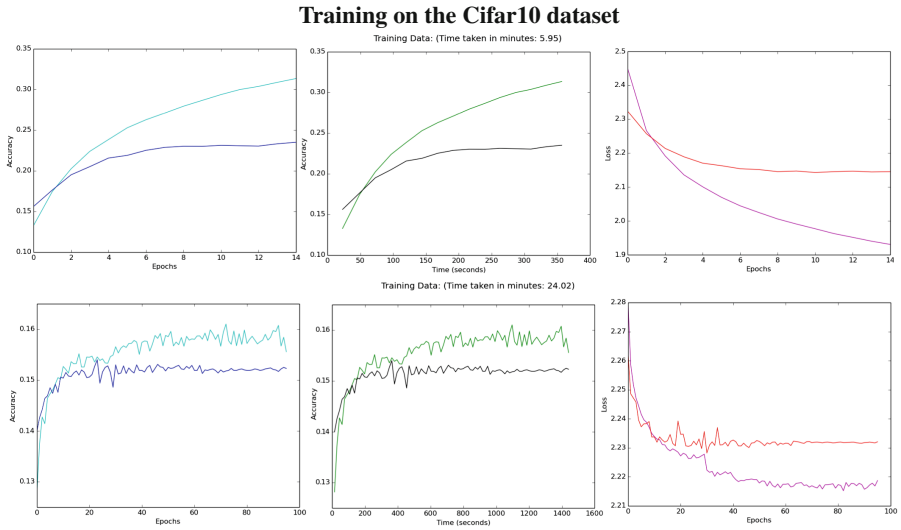

Fig. 5. Training on the Cifar10 dataset: (top) FCNN (bottom) Spatial CNN. Dark blue, black and red are validation values, lighter colours are training values. (Color figure online)

Table 3. Computation time in seconds for an epoch of re-sized fundus images. One epoch is 60,000 training images.

Image size	FCNN epoch	Spatial epoch
2^9	65.56	2435.93
2^8	30.42	1839.12
2^7	14.47	358.90
2^6	8.38	124.63
2^5	3.92	36.91
2^4	0.76	3.72

4 Discussion

The proposed FCNN technique allows training to be conducted entirely in the Fourier domain, in other words only one FFT is required throughout the whole process. The increase in computation time required for the FFT is recovered because of the resulting speed up of the convolution. Compared to spatial approach the evaluation results obtained evidence an exponential increase in efficiency for larger images. Given a more complex network, or a dataset of larger images, the benefit would be even more pronounced.

The results presented demonstrated that using the Fourier representation training time, using the same layer structure, was considerably less than when a spatial representation was used. The analogous Fourier domain convolutions and

more spatially accurate pooling method allowed for a retention in accuracy on both datasets introduced. It was conjectured that the higher accuracy achieved using the proposed FCNN on the Cifar10 dataset was due to the larger Fourier domain kernels within the Fourier convolution layer. Due to the Fourier kernel size, more parameters within the network were obtained than in the case of spatial window kernels. This allowed for more degrees of freedom when learning features of the images.

The reason for lower accuracy of the FCNN using the MNIST dataset is likely due to the network being trained on very small images. This creates boundary issues and information loss in the Fourier domain when converting from the spatial. This is particularly relevant with respect to smaller images; it is much less of an issue in larger images. Hence, when dealing with larger images we would expect no reduction in accuracy in the Fourier domain while achieving the speed-ups shown. To combat this, we could consider boundary conditions with respect to all of our Fourier layers, which is what is done in the spatial case.

5 Conclusion

This paper has proposed the idea of a Fourier Convolution Neural Network (FCNNs) which offers run-time advantages, especially during training. The reported performance results were comparable with standard CNNs but with the added advantage of a significant speed increase. As a consequence the FCNN approach can be used to classify image sets featuring large images; not possible using the spatial CNNs. The FCNN layers are not specific to any architecture and therefore can be extended to any network using convolution, pooling and dense layers. This is the case for the vast majority of neural network architectures. For future work the authors intend to investigate how the Fourier layers can be optimised and implemented with respect to other network architectures that have achieved state-of-the-art accuracies [4,5]. The authors speculate that, given the efficiency advantage offered by FCNNs, they would be used to address classification tasks directed at larger images, and in a much shorter time frames, than would be possible using standard CNNs.

Acknowledgement. The authors would like to acknowledge everyone in the Centre for Research in Image Analysis (CRiA) imaging team at the Institute of Ageing and Chronic Disease at the University of Liverpool and the Fight for Sight charity who have supported this work through funding.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates Inc. (2012)

2. LeCun, Y., Boser, B., Denker, J.S., Howard, R.E., Habbard, W., Jackel, L.D., Henderson, D.: Advances in neural information processing systems, vol. 2, pp. 396–404. Citeseer (1990)
3. Kaggle: Kaggle datasets. https://www.kaggle.com/datasets
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR, abs/1512.03385 (2015)
5. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR) (2015)
6. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: integrated recognition, localization and detection using convolutional networks. CoRR, abs/1312.6229 (2013)
7. Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S., LeCun, Y.: Fast convolutional nets when fbfft: a GPU performance evaluation (2015)
8. Chan, T.F., Wong, C.K.: Total variation blind deconvolution. IEEE Trans. Image Process. **7**(3), 370–375 (1998)
9. Persch, N., Elhayek, A., Welk, M., Bruhn, A., Grewenig, S., Böse, K., Kraegeloh, A., Weickert, J.: Enhancing 3-D cell structures in confocal and STED microscopy: a joint model for interpolation, deblurring and anisotropic smoothing. Measur. Sci. Technol. **24**(12), 125703 (2013)
10. Williams, B.M., Chen, K., Harding, S.P.: A new constrained total variational deblurring model and its fast algorithm. Numer. Algorithms **69**(2), 415–441 (2015)
11. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Math. comput. **19**(90), 297–301 (1965)
12. Campisi, P., Egiazarian, K.: Blind Image Deconvolution. CRC Press, Boca Raton (2007)
13. Kumar, R., Gothwal, H., Kedawat, S.: Cardiac arrhythmias detection in an ECG beat signal using fast fourier transform and artificial neural network. J. Biomed. Sci. Eng. **4**, 289–296 (2011)
14. LeCun, Y., Mathieu, M., Henaff, M.: Fast training of convolutional networks through FFTs (2014)
15. Adams, R.P., Rippel, O., Snoek, J.: Spectral representations for convolutional neural networks (2015)
16. Chollet, F.: Keras. https://github.com/fchollet/keras (2015)
17. Theano Development Team. Theano: a python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688, May 2016
18. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010)
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
20. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27, pp. 2672–2680. Curran Associates Inc. (2014)
21. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS 2010). Society for Artificial Intelligence and Statistics (2010)