Implementations and Optimisations of Optical Conv2D Networks Designs

Philip Birch, Navid Rahimi, Peter Overburry, Rupert Young, Chris Chatwin

Department of Engineering and Design, University of Sussex, Falmer, BN1 9QT, UK

ABSTRACT

Deep learning for object detection offers the advantage of very low electrical power requirements but with the potential of a very large computation bandwidth due to the ability of Fraunhofer diffraction to perform correlation operations. However, many of the current designs of deep learning networks are not easily implemented in the optical domain. In this paper we develop a modified version of the deep learning library, Keras, that can accurately model optical systems. This allows the discovery of the optimal weights by calculating them on a realistic optical system. Noise sources, speckle models, and calibration errors can be accounted for. The effect of using readily realisable filters such as nematic liquid crystal phase only spatial light modulators is investigated. The effect of multiplexing a number of correlations in order to replicate the Conv2D multiple channel input is assessed. The effect of an optically implementable bias and activation functions are examined and compared to the state-of-the-art software implementations. We show that object recognition can be achieved using spatial light modulator technology and give comparable results to digital implementations.

Keywords: Optical Correlation, Deep Learning, Convolution

1. INTRODUCTION

Deep learning algorithms running on graphical processor units (GPUs) have completely revolutionised computer vision. Convolutional 2D (Conv2D) networks are a major new tool,[1–3] and form the backbone of many novel network designs. They are often used as a pre-filter before a fully connected network is used to perform the classification. The Conv2D layers are actually a stacked array of cross-correlation filters that act on multiple channels and sum the result. This is then followed by a scalar bias addition and a non-linear activation function.

There have been several successful implementations of deep learning systems using optics that have the potential of a massive bandwidth for very little electrical power. Diffractive deep learning systems[4] have been developed that implement the entire system optically. Other authors have implemented optical activation layers[5,6] and trained the network optically.[7] A detector and laser only consume milliwatts of power compared to the hundreds of Watts of a GPU. However, GPUs have the advantage of reconfigurability and a high dynamic range when compared to optical solutions. For some applications, the power requirement of GPUs is too large. For example, data centres that classify images do not require retraining very often but maybe running massively parallel tasks, consuming a considerable amount of electrical power. A low powered optical solution could well be attractive in this scenario.

In GPU libraries, such as Tensorflow,[8] the Conv2D is implemented as a space domain correlation operator. This is correlates a set of real only data over multiple channels and sums the result to provide an output. To implement this optically we can use SLMs and Fourier transform lenses. There are multiple arrangements that are potentially useful.[9–11] A reflective nematic SLM with a polariser at 45 degrees, will modulate along the real axis, both positively and negatively . Aligning the polariser at 0 degrees will give phase only modulation. This gives several possible configurations that can be implemented optically. The correlation is implemented with an optical Fourier transform lens in a 4-f system but joint transform correlation filters are also possible.

There are, however, several key differences with electronic Conv2D layers:

Further author information: (Send correspondence to P.B.)
P.B : E-mail: p.m.birch@sussex.ac.uk, Telephone: +44 (0)1273 678553

1. The multiple channels of the Conv2D have to be either spatially or temporally multiplexed. If the output activations are electronically captured, we cannot normally capture the phase information. In spatial multiplexing, the complex fields can be added but, again, the phase is lost and we capture the modulus square of the amplitude. This can be thought of as a type of an extreme version of a squared leaky ReLU[12] activation with a negative slope below zero.

2. The bias function cannot be implemented as a purely additive operator for a single SLM. Real only configurations must restrict the total range of the data to between -1 and +1. Phase only configurations cannot directly add in the bias.

3. The signal to noise ratio is different. SLMs have calibration errors and limited dynamic ranges, sensors introduce numerous sources of noise and the coherent light source causes speckle effects.

To test these effects, this paper will discuss the development a modified version of Keras[8,13] which was developed with a Tensorflow back-end to simulate an optical system. The correlation operators were modelled with Fraunhofer diffraction, and have the ability to add in optical noise sources and limited dynamic ranges. A two layer Conv2D network, followed by a fully connected layer was modelled. The actual gradient calculations and optimisation was carried out on the optical model, rather than the more conventional approach of training the network and adapting the results. Both methods are compared, and the optical modelling shows improved accuracy.

Training was performed on the MNIST dataset. From this the effect of biasing, unconventional activations, phase only arrangement and complex correlation addition modes can be assessed. We show that with the correct design arrangements, state of the art object detection results can be performed.

2. BACKGROUND

2.1 Convolutional Layers

Conventional neural networks have multiple layers of neurons, each neuron is typically connected to all the neurons in the adjacent layers. Each connection has a unique weight.

The output from single neuron will be:

$$o = \sum_{n}^{N} w_n x_n + b \tag{1}$$

Extending this to a layer, it becomes:

$$\mathbf{o} = \mathbf{w}\mathbf{x} + \mathbf{b} \tag{2}$$

This is then passed through some non-linear activation function $\eta(\cdot)$. For the m^{th} layer this is:

$$\mathbf{a}^m = \eta^m\left(\mathbf{o}^m\right) = \eta^m\left(\mathbf{w}^m\mathbf{x}^m + \mathbf{b}^m\right) \tag{3}$$

which are the activations for the specific layer. The problem with these networks is that as they become larger, the number of weights becomes too large to successfully train. Convolutional networks attempt to overcome this network problem by the use of machine learning concepts such as sparse connectivity, parameter sharing and equivariant representation. Sparse connectivity reduces the number of connections, each neuron being no longer connected to every other neuron. The connectivity can be designed by hand, so some regions of the network do not connect with others. This can considerably speed up the computation involved in calculating a layer output.

Parameter sharing means that that the weight parameters are reused and repeated. This reduces the number of weights to be discovered. Equivariance means that these weights do not change depending on position.

These techniques could be implemented by placing constraints on $\mathbf{w}$ in eq. (2), but in practice it is actually easier to use a correlation operator. In this case, a small network is used (typically it is between 3×3 and 7×7 neurons in size). This is then scanned across the image and the output is calculated for each position. Equation (2) becomes:

$$\mathbf{o}_{i,j} = \left(\mathbf{w} * \mathbf{x}_{i,j}\right) \tag{4}$$

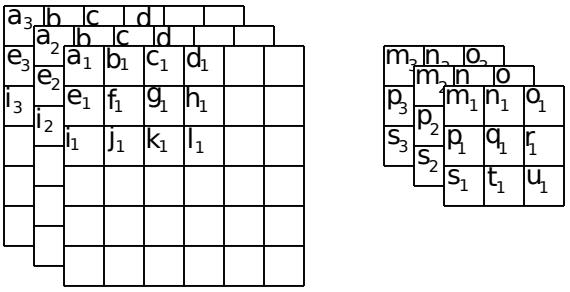

Figure 1. The left hand image illustrates the input into a Conv2D layer. The right hand image is the kernel. The kernel must have the same number of channels (i.e. depth) as the input. There are multiple kernels for each layer and the correlation operation is repeated for each one.

This is incorrectly called a convolution, but since the weights are discovered by a gradient descent operation, it does not actually mater if the process was implemented by a correlation or convolution since the algorithm would learn to flip the kernel.

The Conv2D network inputs may actually be 3D in shape. Multiple planes can be used, for example an RGB image has 3 channels. Each of these are stacked to produce a 3D input array. The kernel will therefore be 3D as well. A 3D element-wise multiplication is performed and the results summed to give a single number. This kernel is only moved in 2D, but there may well be multiple kernels. If an Conv2D layer has P kernels, the number of channels that are fed into the next layer are P. Typically numbers for P range from 64 to 256. Figure 1 illustrates the correlation process in a conv2d layer. Centering the kernel on pixel (1,1) results in an output of

$$o(1,1) = \sum_{i}^{3} a_i \times m_i + b_i \times n_i \ldots l_i \times u_i \tag{5}$$

2.2 Activation Functions

Following the output of each layer is a non-linear activation function. For Conv2D, this is normally a rectified linear function:

$$f(x) = x^{+} = \max(x, 0) \tag{6}$$

This has the advantage of being fast to compute, and does not saturate like some activation functions when the network is very deep.

2.3 Max-pooling

To reduce the spatial dimensions, a max-pooling layer is used. If an max-pooling(N, N) layer is created, it filters out all but the highest activations within the $N \times N$ region. Thus, a $N = 2$ will effectively reduce the size of the network by a half.

3. OPTICAL EQUIVALENCE

Optically, a correlation operation can be performed using the Fourier transforming properties of a lens of focal length f. One way to do this is with a $4 - f$ correlator. If the object Ξ is placed a distance f in front of a lens, and is illuminated with coherent light, its Fourier transform, $\mathcal{F}(\Xi) = \xi$, will occur a distance f behind the lens. Imposing a complex modulation filter, ζ at distance f behind the lens and then passing the light through a second Fourier transforming lens will result in a complex field:

$$\mathcal{F}^{-1}(\zeta \times \xi) = \Xi^{*} \star Z \tag{7}$$

Where $\star$ is the correlation function and is related to convolution via $f \star g = f^{*}(-x, -y) * g(x, y)$. Strictly speaking the output of eq. (7) is inverted since the optical lens can only perform a forward Fourier transform and not the inverse.

3.1 Conv2D layers

A conventional optical correlator, will only perform 2D correlations. In order, to perform the stacked correlation shown in fig. 1, a modification is proposed. Taking each correlation and performing them separately could be implemented. Thus, the Conv2D operation is achieved by temporal multiplexing. However, this is not the same since the intensity output will have been captured by the camera, not the complex fields. Thus, the addition will give an incorrect result. Instead, the correlations need to be spatially multiplexed and optically recombined so an optical addition is performed of the complex field. The addition will occur at the sensor, in addition to the activation function as described in the next section.

3.2 Activation Functions

It is not possible to implement eq. (6) optically. However, when using a sensor, only the intensity of the complex field can be detected, and we can use this as the activation function. If α is the complex field, the activation is

$$\eta(\alpha) = |\alpha|^2 \tag{8}$$

We term this the Squared Rectified Unit (SRU).

3.3 Bias weights

The bias adds a constant onto the output of eq. (7). An additive function is not easy to achieve optically, but a subtractive bias could be achieved with the use a liquid crystal intensity modulator.

$$(\Xi \star Z)\gamma \tag{9}$$

where $\{\gamma \in \mathbb{R} : 0 \leq \gamma \leq 1\}$

Another option would be to modulate the DC term of the filter in the Fourier domain.

$$= \Xi(f) \times Z(f)(1-\gamma)\delta(f-0)) \tag{10}$$
$$= \Xi(f) \times Z(f) - \gamma\Xi(0)Z(0)\delta(f-0) \tag{11}$$
$$= \xi \star \zeta - \gamma\Xi(0)Z(0) \tag{12}$$

This could be implemented on a single SLM, but unless $\Xi(0)Z(0)$ can be forced to be constants and not equal to zero, the modulation will be subject to the additional terms.

3.4 Max-pooling Layers

Max-pooling layers can not be easily implemented optically since they are a non-linear process. An average pooling layer could be approximated by the use of a convolutional filter or blur. This will not give the same result and is the subject of further investigation.

3.5 Flattening Layers

The role of flattening layers is purely to rearrange the data. For example, the output of a Conv2D layer maybe several 2D arrays. The flattening layer re-arranges the data into a single 1D array. This is not easy to optically implement, but may also not be required depending on the network design.

3.6 Dropout Layers

Dropout layers are only used during the training phase, and not in the evaluation of the network. There is therefore no need to optically implement dropout layers if the training is performed with a computer simulation.

3.7 Fully Connected Network

A fully connected layer performs the final output. There are several options to implement this. For now we have implemented the simplest: the data could be captured and the fully connected layer performed electronically. These layers typically are small compared to the Conv2D operations, so they consume little electrical power.

4. NUMERIC COMPUTATION

In Tensorflow, the input weights are stored in 4D tensor of dimension $(B, N_{in}, M_{in}, C_{in})$, where B is the batch size, N_{in} and M_{in} are the spatial dimensions and C_{in} is the number of channels. The weights are in a tensor of shape (n, m, C_{in}, C_{out}). The correlation is preformed with the first 3 dimensions of the weight and the last 3 of the input. This will produce an output tensor of (B, N_o, M_o, C_{out}).

Usually, the output size of the convolution is reduced since only the valid parts of the correlation data are returned. Some networks such as ResNet,[14] require this to remain the same, so the central part of the convolution that is the same shape as the input is returned.

This process is repeated over all the images in the batch size. For training, a large batch size is useful. For actually running the network, a batch size of 1 is common.

A pseudo code representation is shown in listing 1. This is looped over again for each image in the batch.

Listing 1. Conv2D code

```
output = zeros(B, N_o,M_o,C_o)
for p in C_o
        result = zeros()
        for q in C_in
                result = result + conv2d(w(:,:,q,p) , x (B, :, :, q))
        output(B,:,:,p) = result
```

The activation function is then applied to every element in `output`, and this result is passed to the next layer as its input.

To implement the Tensorflow `conv2d` function with an FFT, some data preparation is required. Since the FFT can only perform circular convolutions, zero padding is required. If the kernel size is square, of length K, the data arrays must be zero padded such that they are $(B, N_{in} + K - 1, M_{in} + K - 1, C_{in})$. The FFT is provided by `tensorflow.signal.fft2d`. This calculates the FFT on the inner most 2 dimensions, so a transpose operation is first required. We shift the data both before and after the FFT has been transformed to ensure the phase data is correct. Some of these shifts are unnecessary and will be removed in future optimisations.

Each channel is separately correlated and the results are summed to mimic a Conv2D layer. Since an optical system could be implemented in many different ways, we introduce a new activation function before the summation, we call *inter-activation*. Currently this is used to transform the data from `tensorflow.complex64` to `tensorflow.real`, but other optical setups will be modelled in the future.

To implement the whole simulation we implement a custom Keras layer called `Conv2DOptical` which is a subclass of `keras.layers.convolutional._Conv`. We override the `build` function, which sets up the layer, and the `call` function. The `call` function calculates the forward propagation of the layer, that is, it is passed the input tensor and calculates the output activations for the layer.

To optimise the code, the weights are stored as normal Keras weights, however, a new tensor is added to the class to hold the Fourier transformed version of these. This is only calculated when the weights are updated, which is normally done after a batch has been processed by the optimisation algorithm.

Keras also has a number of callback functions that are useful to reimplement. The `Constraint` function imposes a constraint limit on the Keras weights during training. We can override this to impose realistic optical constraints such as impose phase only filters, bit-depth limitations, etc.

5. EXPERIMENTS

The `Conv2DOptical` layer is currently considerably slower to calculate than a conventional Conv2D layer due to the FFT and other additional operations. Further work is required to optimise the code. Some of the algorithm is currently restricted to the CPU and the GPU/CPU memory transfer is inefficient. We therefore have restricted the network to relatively small sizes.

The MNIST handwriting dataset was used for training and testing. The images are 28×28 and monochromatic, and simple networks can produce reasonable results.

A network was trained with the following structure

- Conv2D - 32 filters of 3×3 in size.
- Conv2D - 32 filters of 3×3 in size.
- 2×2 max pooling layer
- 0.25 dropout layer
- Flatten
- Dense network size 128 with ReLU.
- 0.5 Dropout layer
- Dense layer 10 outputs softmax

This can achieve 0.9855 accuracy in only 2 epochs, with a speed of $385\mu S$ per step on a GeForce RTX 2070 with 8GB of RAM and an Intel Core i7-8750H @ 12x 4.1GHz and 16 GB of PC RAM. The batch size was 128. The validation accuracy is higher than the training accuracy. This is because of the large amount of dropout used in the training. Dropout is performed by disabling some neurons, so some of the information about each sample is lost. The training is therefore made artificially harder. The network was trained on 60000 samples and 10000 validation samples were used.

The Conv2DOptical was then used to replace the Conv2D. Various options were used to study the effect on the performance on the network. Using the FFT and phase only calculations increased the computation time to $46ms$ per step for the two layer system (about $\times 120$ slower than the conventional space domain correlation).

Table 1. Results of different models showing the final validation accuracy for 1 epoch.

1st Conv Layer	2st Conv Layer	Validation Accuracy
Conv2D with bias (ReLU)	Conv2D with bias (ReLU)	0.9789
Conv2DOptical with bias (real)	Conv2DOptical with bias (SRU)	0.9710
Conv2DOptical with bias (real)	Conv2DOptical no bias (SRU)	0.9711
Conv2DOptical with bias (real)	Conv2DOptical no bias (ReLU)	0.9653
Conv2DOptical with bias (real)	Conv2DOptical with bias (ReLU)	0.9720
Conv2DOptical with bias (real)	Conv2DOptical (phase only) no bias (SRU)	0.9717
Conv2DOptical with bias (real)	Conv2DOptical (phase only) with bias (SRU)	0.9737
Conv2DOptical (phase only) no bias (SRU)	Conv2DOptical (phase only) with bias (SRU)	0.9683

Table 1 shows the validation accuracy after various experiments. Different activations have been used (Real only, ReLU, and SRU). Also, the result of running the Conv2DOptical network with phase only correlations has been shown. The results show that the effect of biasing on this particular problem is negligible. The bias has the effect of shifting the activation function's response. It may be in this case the bias values were near zero and the network has trained itself to compensate for this. Further testing on different datasets is required to fully model the effect. The effect of the SRU layer is less surprising since it is well known the many non-linear functions can be used as the activation function. Again, the network has adapted to the change.

Training with the phase only correlation works well. Here the network can be implemented with a cascade block consisting of a phase only SLM and CCD camera. This result is significantly better then when a normal Conv2D is trained and the phase is extracted. The ability of the network to work with different types of layer is useful. However, we note that changing the inter-activation layer to a SRU from a complex addition causes the network to fail to converge.

6. CONCLUSIONS

This paper reports on the development and design of optical convolutional layers for implementing deep learning networks a partial optical domain. A custom Keras layer has been developed that can simulate an optical correlation in the Fourier domain. This is capable of simulating full complex filters and phase only filters. The network has been tested on a small handwriting dataset and shown to have very similar results to a conventional Conv2D based network.

REFERENCES

[1] LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y., "Object recognition with Gradient-Based learning," in [*Shape, Contour and Grouping in Computer Vision*], Forsyth, D. A., Mundy, J. L., di Gesú, V., and Cipolla, R., eds., 319–345, Springer Berlin Heidelberg, Berlin, Heidelberg (1999).

[2] Schmidhuber, J., "Deep learning in neural networks: An overview," *Neural Netw.* **61**, 85–117 (Jan. 2015).

[3] Fukushima, K., "Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.* **36**(4), 193–202 (1980).

[4] Lin, X., Rivenson, Y., Yardimci, N. T., Veli, M., Luo, Y., Jarrahi, M., and Ozcan, A., "All-optical machine learning using diffractive deep neural networks," *Science* **361**, 1004–1008 (Sept. 2018).

[5] Zuo, Y., Li, B., Zhao, Y., Jiang, Y., Chen, Y.-C., Chen, P., Jo, G.-B., Liu, J., and Du, S., "All-optical neural network with nonlinear activation functions," *Optica, OPTICA* **6**, 1132–1137 (Sept. 2019).

[6] Miscuglio, M., Mehrabian, A., Hu, Z., Azzam, S. I., George, J., Kildishev, A. V., Pelton, M., and Sorger, V. J., "All-optical nonlinear activation function for photonic neural networks [invited]," *Opt. Mater. Express, OME* **8**, 3851–3863 (Dec. 2018).

[7] Hughes, T. W., Minkov, M., Shi, Y., and Fan, S., "Training of photonic neural networks through in situ backpropagation and gradient measurement," *Optica, OPTICA* **5**, 864–871 (July 2018).

[8] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015). Software available from tensorflow.org.

[9] Birch, P. M., Young, R., and Chatwin, C., "Spatial light modulators," in [*Optical and Digital Image Processing*], Cristobal, G., Schelkens, P., and Thienpont, H., eds., 179–200, Wiley-vch, Weinheim (Jan. 2011).

[10] Birch, P. M., Young, R., Chatwin, C., Farsari, M., Budgett, D., and Richardson, J., "Fully complex optical modulation with an analogue ferroelectric liquid crystal spatial light modulator," *Opt. Commun.* **175**, 347–352 (Jan. 2000).

[11] Birch, P. M., Young, R., Budgett, D., and Chatwin, C., "Two-pixel computer-generated hologram with a zero-twist nematic liquid-crystal spatial light modulator," *Opt. Lett.* **25**, 1013–1015 (July 2000).

[12] Maas, A. L., Hannun, A. Y., and Ng, A. Y., "Rectifier nonlinearities improve neural network acoustic models," in [*in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*], (2013).
[13] Chollet, F. et al., "Keras." https://keras.io (2015).
[14] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," (Dec. 2015).