International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

Implementation of FFT-Based Convolutional Neural Networks on Medical Imagery for Studying Performance Boost

Rohit Bokade[1], Dr. Namrata Gharat[2]*
[1] PG - Artificial intelligence, K J Somaiya Institute of Technology, Sion, Mumbai, India.
[2] Assistant Professor -Artificial intelligence, K J Somaiya Institute of Technology, Sion, Mumbai, India.
Emails: *rohit.bokade@somaiya.edu[1], ngharat@somaiya.edu[2]*
****Corresponding Author Orcid ID:*** *0009-0005-5893-0969*

Abstract
This study focuses on the implementation of CNNs based on the Fast Fourier Transform (FFT) in medical images to study the performance boost compared to traditional neural networks. Using a Kaggle dataset of brain tumour imagery, we examine the effectiveness of FFT-based CNNs for the detection and classification of brain tumours. By leveraging the advantages of FFT, we aim to improve the efficiency and accuracy of CNNs in analyzing medical images. Our results demonstrate that FFT-based CNNs outperform traditional CNNs in terms of accuracy and computational efficiency. This performance boost is attributed to the inherent properties of FFT, which enable faster convolution operations and reduced complexity. In order to ensure a fair comparison between traditional CNNs and Fourier Transform-based convolutions, the forward pass for both methods was implemented from scratch using NumPy and SciPy libraries. By doing so, any optimization benefits gained by either technique through specific libraries or computation platforms are eliminated. This approach allows for a more accurate evaluation of the relative performance of each method in terms of inference time and accuracy, while also providing insights into potential areas for improvement and optimization.

Keywords: *FFT-CNN, CNN, FFT, Fourier transform for CNN*

1. Introduction
The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT) of a signal or image. The Convolution Theorem, a fundamental mathematical result, states that under appropriate conditions, the Fourier transform of a convolution of two functions is the pointwise product of their Fourier transforms. In other words, convolution in one domain (e.g., time domain) is equivalent to pointwise multiplication in the other domain (e.g., frequency domain). This property of the FFT enables a significant reduction in the computational complexity of the convolution operation in CNNs. These techniques can help to address the challenge of computational complexity in Convolutional neural networks and make them more deployable. [1] In this study, we aim to conduct a comparative analysis between the performance of FFT-based convolutions and standard convolutions. We have chosen medical imagery as the target domain for our imagery dataset, as medical images tend to exhibit complex feature sets and characteristics that are favorable for examining the performance of algorithms. By focusing on medical imagery, we can gain valuable insights into the computational complexity and efficiency of FFT-based convolutions versus standard convolutions in real-world applications. Furthermore, this comparison will help us determine the potential benefits of employing FFT-based convolutions in the medical imaging domain, which could lead to improved performance and faster processing times for image

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

reconstruction processes in these critical healthcare applications. The proposed study intends to quantify the acceleration or gain factor achieved by the technique of Fast Fourier Transform over conventional convolutional neural networks by creating a common platform implementation and running both the technique in forward propagation of neural network model with same training loop operating with same weights and biases so as to avoid any source of error. The dataset to be chosen needs to be simpler dataset to work on since the custom implementation of forward propagation will be computed without any extra acceleration like gpu or tpu modules.

2. Methodology

Theoretical Analysis of Convolution theorem

Implementation would first require a theoretical basis of the convolution theorem which governs the proposed methodology. The underlying intuition given by the Convolution Theorem states that for two functions v and u, we have

F (v∗u) = F (v) xF (u) where F denotes the Fourier transform, * denotes convolution and x denotes the Hadamard Pointwise Product. [1] 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. [2] The network architecture table is shown in Table 1.

Selection of Base Dataset

Several parameters for the selection of the dataset to be considered for the proposed study are as follows:

1. The selected dataset should have images with complex patterns that need to be learned.
2. The dataset should not be complicated in terms of having multiple output classes or even complex classification, which may introduce classification level errors in the measurement of speed difference since any significant time added by non-convolutional layers in the network may add up and distort the results.
3. Since the study needs to be conducted on machines with limited resources, the proposed dataset should preferably be monochrome to avoid multiple parallel layers being involved causing an increase in the number of weights and biases that need to be maintained during the training.
4. Dataset should have been worked upon earlier to ensure sufficient reference models are available for the same. Based on these criteria, Brain Tumor Dataset from Kaggle. The selected dataset has a binary classification of monochrome brain scan imagery data which has complex patterns to learn from while maintaining the smaller size of images which means the model could be trained over general-purpose hardware. The model also has several legacy models trained on top of it.

Training of base model

The base model was designed to have minimal complexity yet factual accuracy. This was designed as follows:

Table 1 Network Architecture Table

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 223, 223, 1)	10
max pooling2d (MaxPooling2D)	(None, 111, 111, 1)	0
conv2d 1 (Conv2D)	(None, 109, 109, 1)	10

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

max pooling2d 1 (MaxPooling2D)	(None, 54, 54, 1)	0
conv2d 2 (Conv2D)	(None, 52, 52, 1)	10
max pooling2d 2 (MaxPooling2D)	(None, 26, 26, 1)	0
flatten (Flatten)	(None, 676)	0
dense (Dense)	(None, 640)	433280
dense 1 (Dense)	(None, 1)	641

Implementation of Base Model

The base model described above was implemented on a Kaggle kernel since the dataset was directly available through the Jupyter Notebook for training.

1. **The following code was used to do Data parsing**

```
#the two categories present in our dataset categories= ['no','yes']
#Display some of the images from the original datasetfor category in categories:
Win=plt.figure (fig size= (20, 10))
If category=='no':
win.suptitle ("Original Data\n\nBrain MRI Images without Tumor", fontsize=15)
Else:
win.suptitle ("Brain MRI Images with Tumor", fontsize=15)
Index=1
for img in os.listdir(os.path.join("/kaggle/input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset",category)):
image=cv2.imread(os.path.join("/kaggle/input/brain-mri-images-for-brain-tumor-detection/brain_tumor_dataset",category,img))
Try:
    a. win.add_subplot (6, 6, index)
    b. plt.imshow (image)
    c. plt.axis ('off')
    d. Index=index+1
Except:
    e. Break
plt.show ()
```

2. **To reduce the overfitting of model, the data set was extended by augmenting the data pipeline in the runtime.**

```
#Create an augmented image generatordatagen=ImageDataGenerator (width_shift_range=0.1, height_shift_range=0.1, zoom_range=0.1, horizontal_flip=True, shear_range=0.15, rotation_range=10)
```

3. **Training of Keras model:**

```
Model = keras.models.Sequential ([
keras.layers.Conv2D (1, (3, 3), activation='relu', input shape= (225, 225, 1)),
keras.layers.MaxPooling2D (2, 2),
keras.layers.Conv2D (1, (3, 3), activation='relu'),
keras.layers.MaxPooling2D (2, 2),
keras.layers.Conv2D (1, (3, 3), activation='relu'),
keras.layers.MaxPooling2D (2, 2),
keras.layers.Flatten (),
keras.layers.Dense (640, activation='relu'),
keras.layers.Dense (1, activation='sigmoid')])
From tensorflow.keras.optimizers import RMSprop
model.compile (loss='binary_crossentropy',
Optimizer=RMSprop (learning_rate=0.0001),
Metrics= ['acc']
```

4. **Implementation of platform independent convolutions:**

```
Def fft_forward (img, w, b, hparameters1):
l = img. Shape [1]
Pa = (l - 3) // 2
f2 = np.pad (w.reshape (3, 3), (pa, pa))
Img = img.reshape (l, l)
If l % 2 == 0:
l = l-1
Img = img [: l,: l]
```

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

```
f = np.fft.fft2 (img)
f21 = np.fft.fft2 (f2)
AAA = f * f21
Final = np.fft.ifft2 (AAA)
Zeros = np.zeros ((l, l))
Zeros [1: l // 2+1, 1: l // 2+1] = final. Real [l // 2+1: l // 2+1:]
Zeros [1: l // 2+1, l // 2+1:] = final. Real [l // 2+1: 1: l // 2+1]
Zeros [l // 2+1: 1: l // 2+1] = final. Real [1: l // 2+1, l // 2+1:]
Zeros [l // 2+1: l // 2+1:] = final. Real [1: l // 2+1, 1: l // 2+1]
Zeros = zeros [1: l, 1: l] + b
Else:
f = np.fft.fft2 (img)
f21 = np.fft.fft2 (f2)
AAA = f * f21
Final = np.fft.ifft2 (AAA)
Zeros = np.zeros ((l, l))
Zeros [1: l // 2+1, 1: l // 2+1] = final. Real [l // 2+1: l // 2+1:]
Zeros [1: l // 2+1, l // 2+1:] = final. Real [l // 2+1: 1: l // 2+1]
Zeros [l // 2+1: 1: l // 2+1] = final. Real [1: l // 2+1, l // 2+1:]
Zeros [l // 2+1: l // 2+1:] = final. Real [1: l // 2+1, 1: l // 2+1]
Zeros = zeros [1: l-1, 1: l-1] + b
Return zeros
Def conv_forward (A_prev, W, b, hparameters):
(M, n_H_prev, n_W_prev, n_C_prev) = A_prev.shape [0], A_prev.shape [1], A_prev.shape [2], A_prev.shape [3]
(F, f, n_C_prev, n_C) = W.shape [0], W.shape [1], W.shape [2], W.shape [3]
Stride = hparameters ["stride"]
Pad = hparameters ["pad"]
n_H = int (((n_H_prev - f + (2 * pad)) / stride) + 1)
n_W = int (((n_W_prev - f + (2 * pad)) / stride) + 1)
Z = np.zeros ((m, n_H, n_W, n_C))
A_prev_pad = zero_pad (A_prev, pad)
For i in range (m):
a_prev_pad = A_prev_pad[i]
For h in range (n_H):
a. for w in range(n_W):
b. for c in range(n_C):
   i. vert_start = h * stride
   ii. vert_end = vert_start + f
   iii. horiz_start = w * stride
   iv. horiz_end = horiz_start + f
   v. a_slice_prev = a_prev_pad[vert_start:vert_end,horiz_start:horiz_end,:]
   vi. Z[i, h, w, c] = conv_single_step(a_slice_prev, W[:,:,:,c], b[:,:,:,c])
Assert (Z.shape == (m, n_H, n_W, n_C))
Cache = (A_prev, W, b, hparameters)
Return Z, cache
```

Performance Evaluation:

For enabling comparison of performance, forward passes of both traditional CNN and FFT CNN were written from scratch, so as to avoid any undue boost to one due to implementation details. To get timed output of these sections following code blocks were used. [4]

```
conv_out = []

Start = time. Time ()
For i in range (len(x_test)):
 Img = cv2.resize (x_test[i], (225,225)).reshape (1, 225, 225, 1)
 Output = conv_model (img, weights, hparameters1, hparameters2)
 If output [0] >= 0.5:
  conv_out.append (1)
 Else:
  conv_out.append (0)
End = time. Time ()
 Print ('Time taken by CNN to predict class for 253 images is', end - start)
fft_out = []
Start = time.time ()
For i in range (len (x_test)):
 Img = cv2.resize (x_test[i], (225,225)).reshape (1, 225, 225, 1)
 output1 = fft_model (img, weights, hparameters1, hparameters2)
 If output1 [0] >= 0.5:
```

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

```
    fft_out.append (1)
  Else:
    fft_out.append (0)
End = time. Time ()
Print ('Time taken by FFT to predict the class for 253 images is', end - start)
```

3. Results and Discussion

3.1 Evaluation of Confusion Matrix

Table 2 shows the comparison of the confusion matrix.

Table 2 Confusion Matrix Comparison

Attribute	Normal Convolution	FFT Convolution
Time Taken	200.331	38.61
True Positive	83	78
False Positive	25	30
False Negative	17	18
True Negative	128	127

3.2 Evaluation of Observation of Normal CNN Over FFT CNN

From these observations following evaluations were derived. Which points to a 5.1x increase in the speed of CNN due to the FFT technique. Observation of Normal CNN Over FFT CNN shown in Table 3. [6]

3.3 Discussion of Results

1. The proposed method showed around 5.1x increase in the speeds of forward pass.
2. Both the methods showed similar recall performances. With minor difference between recalls.
3. The precision performance of the proposed algorithm showed deterioration over the original method.
4. The accuracy had a Deterioration of around 3% which could be acceptable for general cases although could be a matter of discussion in critical domains like medical domains
5. The observations suggest that the proposed method provides boost of 1.64x with loss of every 1% in accuracy. Interpolating this one can understand that, one can expect loss of 1.2% for every 2x increase in speed by the proposed algorithm. [5]

Table 3 Observation of Normal CNN over FFT CNN

Metric	Value
Ratio of speeds	5.1
Recall of original CNN	0.83
Recall of FFT CNN	0.81
Precision of CNN	0.76
Precision of FFT CNN	0.72
F-measure Accuracy of CNN	0.793
F-measure Accuracy of FFT CNN	0.762
Average increase in speed per percent decrease in accuracy	1.64

Conclusion

The proposed method of using FFT-based convolutional neural networks showed promising improvements over traditional CNN methods. The results were calculated even without factoring in real-world boosting of FFT algorithms by GPUs increasing the real-world speeds further by optimization. While the speed boost factors have been impressive, one should be cautious while using these algorithms in accuracy intensive tasks like the medical domain as a decrease in accuracy may become a life-or-death situation. However, the proposed method is a promising prospect for applications requiring high-speed convolutions on very large kernel sizes like self-driving cars or planet-scale weather models. The study also successfully demonstrated the implementation of convolutional neural networks from scratch using the proposed method.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 03 March 2024
Page No: 508 - 513
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0073

References

[1]. Highlander, T. & Rodriguez, A. Very Efficient Training of Convolutional Neural Networks using Fast Fourier Transform and Overlap-and-Add. (2016)

[2]. Hsiao, V., Nau, D. & Dechter, R. Fast Fourier Transform Reductions for Bayesian Network Inference. Proceedings of the 25th International Conference on Artificial Intelligence and Statistics. 151 pp. 6445-6458 (2022, 3, 28), https://proceedings.mlr.press/v151/hsiao22a.html

[3]. Pratt, H., Williams, B., Coenen, F. & Zheng, Y. FCNN: Fourier Convolutional Neural Networks. Machine Learning and Knowledge Discovery in Databases. pp. 786-798 (2017)

[4]. Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S. & LeCun, Y. Fast convolutional nets with fbfft: A GPU performance evaluation. ArXiv Preprint ArXiv: 1412.7580. (2014)

[5]. Cooley, J. & Tukey, J. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation. 19, 297-301 (1965)

[6]. Wang, Z., LAN, Q., Huang, D. & Wen, M. Combining FFT and spectral-pooling for efficient convolution neural network model. 2016 2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016). pp. 203-206 (2016)