



Article

# Accelerated Diagnosis of Novel Coronavirus (COVID-19)—Computer Vision with Convolutional Neural Networks (CNNs)

Arfan Ghani <sup>1,2,\*</sup> , Akinyemi Aina <sup>3</sup>, Chan Hwang See <sup>4</sup> , Hongnian Yu <sup>4</sup> and Simeon Keates <sup>5</sup>

<sup>1</sup> Department of Computer Science and Engineering, American University of Ras al Khaimah, Ras al Khaimah P.O. Box 10021, United Arab Emirates

<sup>2</sup> Research Centre for Future Transport and Cities, Coventry University, Coventry CV1 5FB, UK

<sup>3</sup> School of Computing, Electronics and Maths, Coventry University, Coventry CV1 5FB, UK; akinaina97@gmail.com

<sup>4</sup> School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK; c.see@napier.ac.uk (C.H.S.); h.yu@napier.ac.uk (H.Y.)

<sup>5</sup> Deputy Vice-Chancellor (Research), University of Chichester, College Lane, Chichester PO19 6PE, UK; s.keates@chi.ac.uk

\* Correspondence: arfan.ghani@aurak.ac.ae or ghani\_786@yahoo.com

**Abstract:** Early detection and diagnosis of COVID-19, as well as the exact separation of non-COVID-19 cases in a non-invasive manner in the earliest stages of the disease, are critical concerns in the current COVID-19 pandemic. Convolutional Neural Network (CNN) based models offer a remarkable capacity for providing an accurate and efficient system for the detection and diagnosis of COVID-19. Due to the limited availability of RT-PCR (Reverse transcription-polymerase Chain Reaction) tests in developing countries, imaging-based techniques could offer an alternative and affordable solution to detect COVID-19 symptoms. This paper reviewed the current CNN-based approaches and investigated a custom-designed CNN method to detect COVID-19 symptoms from CT (Computed Tomography) chest scan images. This study demonstrated an integrated method to accelerate the process of classifying CT scan images. In order to improve the computational time, a hardware-based acceleration method was investigated and implemented on a reconfigurable platform (FPGA). Experimental results highlight the difference between various approximations of the design, providing a range of design options corresponding to both software and hardware. The FPGA-based implementation involved a reduced pre-processed feature vector for the classification task, which is a unique advantage of this particular application. To demonstrate the applicability of the proposed method, results from the CPU-based classification and the FPGA were measured separately and compared retrospectively.

**Keywords:** Convolutional Neural Networks (CNN); computer vision; reconfigurable architectures; intelligent system design; COVID-19; embedded devices



**Citation:** Ghani, A.; Aina, A.; See, C.H.; Yu, H.; Keates, S. Accelerated Diagnosis of Novel Coronavirus (COVID-19)—Computer Vision with Convolutional Neural Networks (CNNs). *Electronics* **2022**, *11*, 1148. <https://doi.org/10.3390/electronics11071148>

Academic Editors: Jungong Han and Guiguang Ding

Received: 27 February 2022

Accepted: 3 April 2022

Published: 6 April 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

COVID-19 is a respiratory disease that has spread across the globe. It resulted in the prohibition of social gatherings and the implementation of harsh measures such as social distancing in order to lower infection rates. Its financial impact has had a significant impact on numerous industries and learning paradigms have been drastically altered as a result. COVID-19 is still being cured due to problems posed by an evolving virus, and existing testing methodologies are insufficient [1]. This is because the virus can mimic the symptoms of other, more dangerous, diseases, such as pneumonia. As a result, new strategies must be proposed and researched to speed-up the research process for COVID-19 diagnosis, therapies, and vaccinations.

In many areas impacted by the epidemic, the RT-PCR test, which consists of a swab collected from individuals after insertion into respiratory sources, such as the nose and mouth, is commonly used to check whether or not they have contracted COVID-19. The chest CT scan test, however, is an important complement to RT-PCR tests because of its high sensitivity for the diagnosis of coronavirus disease. The detection rates of CT- and RT-PCR-based techniques are compared in [2,3]. CT scans were found to be more likely than RT-PCR methods to detect COVID-19 symptoms earlier, as reported in [4,5]. The findings also indicated that a RT-PCR test is still needed to confirm a COVID-19 positive case. It could be inferred that non-invasive CT scans could be used as an additional diagnostic tool to confirm the need for cautionary action in people who have symptoms. This would help to address the issue of a scarcity of RT-PCR test kits for large-scale testing in both developing and developed countries. Due to the nature of COVID-19 and how it manifests, the longer it remains untreated the more signatures appear in radiology scans, such as glass opacities and other linear opacities [2]. Because of this complication, the symptoms are not definitive enough to be classified. Meanwhile, many hospitals do not disclose patients' medical imaging scans for research purposes due to GDPR (General Data Protection Regulation) requirements, a regulation in EU law on data protection and privacy [2]. Due to the required complex image recognition and lack of a large image dataset, research facilities will have to develop methods with limited samples. However, a well-designed image classification neural network could provide a solution with limited samples for identification. Machine Learning (ML) adopts several mathematical calculations to learn and identify different sets of data to make a prediction. There are several types of neural networks reported in the literature, however, due to the high accuracy of image processing capabilities. CNN-based techniques have been widely used to classify digital images compared to their counterparts [2,4,5].

CNNs are widely used in the machine vision system; however, their use in real-life applications incurs high computational costs. Therefore, implementing CNNs on embedded devices is a major challenge. A solution to this challenge is to take advantage of the fine-grain parallelism offered by reconfigurable hardware, such as FPGAs, to exploit the inherent concurrency exhibited by CNN-based algorithms. To accelerate machine learning algorithms, hardware implementations on FPGAs have been reported in the literature [4–8], where CNN-based algorithms were applied to classify images. FPGAs have been used as hardware accelerators because of their flexibility in the way in which neural network models can be implemented and are modified. Authors in [4] focused on the efficiency across different FPGA architectures, whereas authors in [5] compared an FPGA architecture to other computational methods (multiple CPUs). Both works elaborated that FPGAs could offer competitive performance in making predictions with CNNs.

As reported in [9,10], FPGAs have been utilized to accelerate machine learning algorithms, albeit for different applications. The fully parallel processing capabilities of FPGAs were utilized to reduce the overall MAC (multiply-accumulate) operations needed during training. Authors in [9] proposed a multi-FPGA edge-device solution as an alternative to cloud-based high-performance PCs. The investigation reported in [10] combines both an FPGA and an ASIC (Application Specific Integrated Circuit) for training and classification, respectively. The flexibility of the FPGA hardware is used to create a dynamic model generation system based on the dataset using different softcore processors. Further evidence of FPGA-based deep learning acceleration is reported in [11–15]. As stated in [11], the authors used an FPGA to increase the speed of stochastic gradient descent in matrix factorization operations. The FPGA-based solution offered a  $15.3\times$  speed-up over the GPU implementation with a  $60\times$  less data dependency reduction. The work reported in [12–15] achieved greater performance in object detection inference [15] and a reduction in overall MAC operations per layer [14]. Alternatively, work performed in [12] uses an FPGA to handle the parallel context of echo data from received laser signals at high speed by using deep learning.

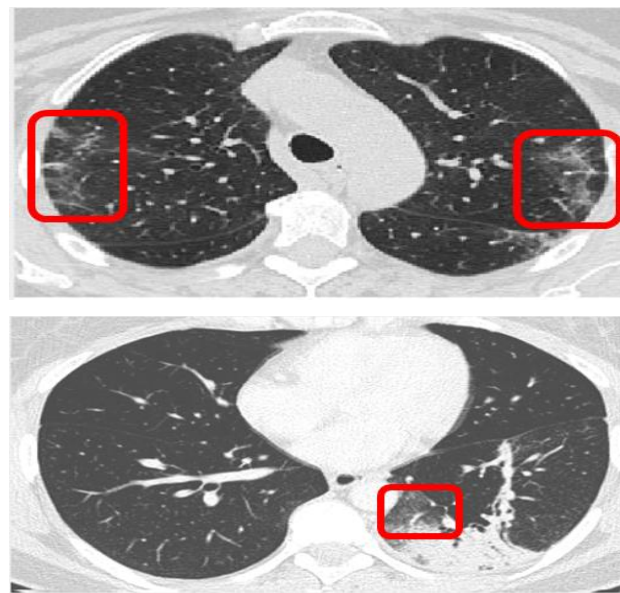
The investigation performed in [16] utilizes a data-driven hybrid model of LSTM (Long Short Term Memory) and a CNN network to predict the cases of COVID-19 per

region. While it makes good use of LSTM to mitigate the vanishing gradient issue of RNNs, a CNN is used in combination, but is not suited to long sequences of timescale-based data. Efforts have been made to predict the nonlinearity pattern of reported cases in specific regions with good accuracy.

This case study reports the use of hardware accelerators such as FPGAs to accelerate the diagnosis of COVID-19 CT scan images by utilizing a machine learning algorithm. It was demonstrated that an increased rate in detection could lead to performing the necessary measures more promptly than RT-PCR testing. Taking the combined concepts of self-supervised learning [2], convolutional image processing techniques, and classification methods [17], a solution could be realized to meet the speed and accuracy requirements needed for medical diagnostics based on computer vision. This case study investigated a CNN-based classification method that has been optimized for the classification of COVID-19 CT scans. The authors reviewed and implemented different approaches to develop a machine learning-based method and map it on reconfigurable hardware to improve the overall classification speed. This case study is organized as follows: Section 2 provides data pre-processing methods. Section 3 elaborates on the process behind the design of the proposed CNN and its hardware implementation. Section 4 concludes this case study.

## 2. Materials and Methods

Previous research on CT scan-based diagnostics has related many visual indications to the presence of COVID-19 symptoms. As shown in Figure 1, the red rectangles indicate the existence of ground-glass opacities (GGO), whereas the further distributions of aberrant lesions are the after-effects of the blockages generated [18,19]. GGOs reflect the mucus inflammation present, and the additional distributions of abnormal lesions are the after-effects of the obstructions caused. All cases of GGOs, pleural effusion, and consolidation are classified as COVID-19 indicators in this investigation and are used as factors in the CNN model presented in this case study to classify CT scans.



**Figure 1.** Examples of COVID-19 signatures in CT scans (red rectangles indicate GGO).

To generate the most effective training results, the input data was pre-processed with various methods [20]. The approach involved converting grayscale input images into binary images using fuzzy logic to isolate ground-glass opacity sections. Grayscale conversion to binary images presented a variation of results. To extract meaningful features from the training set, fuzzy logic edge detection was used on a binary image that was

created from a grayscale image filter using adaptive thresholding. Figure 2 shows a COVID scan in binary format. The aim was to analyze the GGO and consolidation areas to form a distinct differentiation between COVID and non-COVID scans. The consensus is that COVID scans display patterns of white pixels (1's) in binary form, whereas non-COVID scans will display minimal white pixels or none. Figure 3 shows inverted versions of the source image where the black pixels are the isolated consolidation sections [21]. Figure 4 is an example of a COVID image that is filtered to display the most important parts of the image. The white patches on the edges are the detections of GGO presence. By using the grayscale thresholding function, the values were calculated within the range 'i' as shown by the expression ( $95 < i < 145$ ).



Figure 2. Region analysis of thresholding image as a binary file (positive COVID scan).

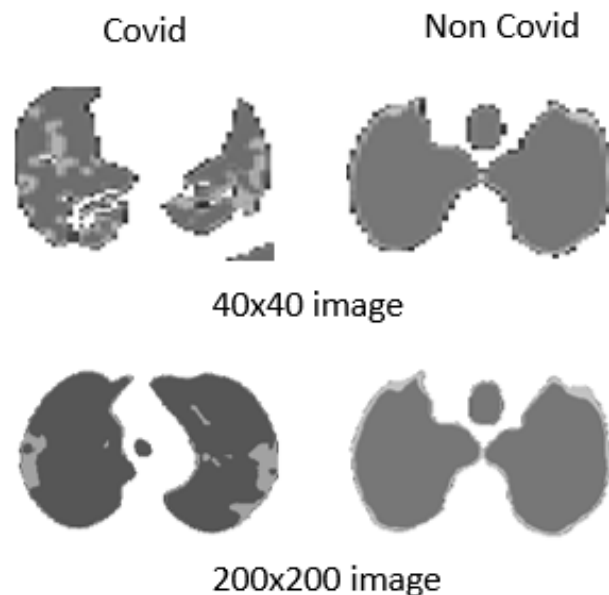


Figure 3. Different resolutions of the pre-processed images by using multi-level thresholding.



**Figure 4.** Binary image after using a median filter kernel size  $10 \times 10$ .

A median filter was used to remove the noise which isolated the ground-glass opacities and consolidation. A multiple-level grey-level threshold function was performed. The multi-level thresholding technique takes the regions of the image with the most standard deviation as separate gradients. It then uses this data to create indices for each region of the image that has the greatest deviation in grey levels to isolate specific elements. The image is then re-sized to a lower resolution and applied to various classifiers tested in this case study.

In Figure 3, the series of images being shown are the different output resolutions of the feature map to be used by the neural network. The top images are COVID scans and the bottom images are non-COVID scans. The approach used was to model a neural network classifier as the final layer of the CNN. The accuracies of the classifications were used as performance metrics for the most input image resolution as shown in Table 1, where both SVM (Support Vector Machine) and KNN (K-nearest neighbors) classifiers were used. To improve the classification accuracy, different feature extraction methods were used and evaluated. The feature extractors used were SURF (Speeded-Up-Robust-Features), MSER (Maximally Stable Extremal Regions), Bag of features and Haar Wavelet Transform.

**Table 1.** Classification performance impact at different resolutions and pre-processing methods.

Image Resolution	Classifier Type	Accuracy	Pre-Processing Method
$200 \times 200$	SVM	87.5%	Adaptive Thresholding
$50 \times 50$	SVM	90.0%	Wavelet Transform
$100 \times 100$	SVM	84.2%	Adaptive Thresholding
$200 \times 200$	KNN	81.0%	Wavelet Transform

Recognition and classification become challenging, because of the variety in shape, size, and other associated variables within the CT scan images. As a result, approaches for extracting local features, such as SURF, are particularly beneficial in enhancing classification accuracy. MSER, on the other hand, is a method for detecting blobs in images. Bag of features is a method for describing and computing visual similarities. All these methods were investigated to pre-process images to improve the feature vector for classification by CNN.

For each of the feature extractors, the multi-level thresholding method was used as the primary pre-processing method. By using the MSER, it acted as an incremental pixel intensity categorizer. Depending on the threshold delta parameter, the features extracted were measured by how many occurrences of pixels in a specific range compared to the others. The next stage of the investigation used the bag of features functionality. This method collected a combination of MSER and SURF features to generate a vocabulary of words. The vocabulary bank was used as a reference to the extracted features to determine which class an image sample belonged to with accuracies shown in Table 2. In Table 3, the classification accuracies are given for each classifier type where the K nearest neighbor was found to be the most effective with the bag of features method.

**Table 2.** Results of a vocabulary bank of 500 words.

Type	Accuracy
Fine Tree	74.5%
Medium Tree	75.9%
Fine KNN	92.9%
Coarse KNN	80.5%
Cubic KNN	87.5%

**Table 3.** Adjusted parameters of vocabulary bank results (100 Words).

Adjusted Parameter (Fine KNN) Distance Weight	Accuracy
Squared Inverse	93.3%
Inverse	93.3%
<b>Equal</b>	<b>93.4%</b>
Chebyshev	78.7%

In order to improve the accuracy, the number of features in the bag of words was reduced to 100. This investigation helped in selecting the most relevant feature vectors as a form of PCA (Principal Component Analysis), where the weaker features were omitted, thereby reducing predictor overlap, which increases classification accuracy. With the changes to the number of words in the vocabulary bank, the accuracy increases by 0.5% when comparing Tables 2 and 3. The distance calculated for the KNN was also modified to only consider equal distances for feature quantization. For this specific case study, it could be established that the bag of features method could be a viable choice for pre-processing; however, the time taken to extract useful features from a relatively large dataset (2482 images) is much longer than the previously mentioned functions. It takes approximately 8.2 s per iteration, which equates to 14 min to process 2482 images.

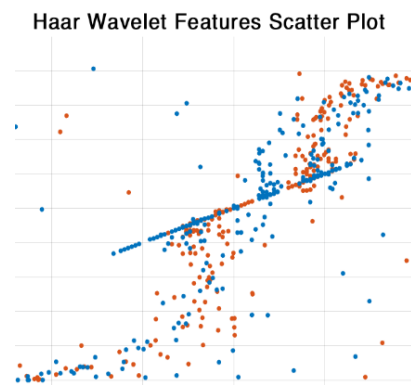
Figure 5 shows the distribution of features for each class. Here the bag of features function generates features with noticeable separation around the bottom left-hand side of the graph, whereas the top right-hand side produces more overlapping features. For optimal classification, the discrete wavelet transformation (DWT) approach was also investigated, where the Haar Wavelet variant was tested. This type includes finding the approximate coefficients of sudden changes in pixel intensity [22] as shown in (1)

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

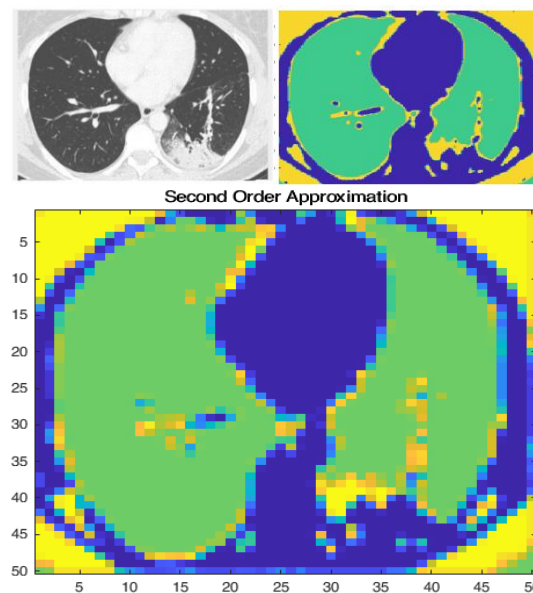
where  $\psi(t)$  is the wavelet transformation function and the outputs depend on the result of applying the wavelet kernel to the image, whereas the output could be between 1 to  $-1$  [22]. To illustrate this process and the effect it produces, the indices were separated as RGB values. DWT, in this case, reduces the dimensionality of the overall image, thus creating a feature map that is more suitable for the FPGA implementation of the neural network.

The RGB index image in Figure 6 shows the previously mentioned multiple thresholding technique, which is separated into three parts and the areas of consolidation are marked as the yellow sections. The primary objective for the Haar Wavelet transformation was to reduce the overall dimensions whilst retaining the most important data. By implementing the first and second-order derivation, Figure 6 (bottom plot) shows the difference in the features retained. The second-order derivation was used as a benchmark for its efficacy as a feature map.





**Figure 5.** Scatter plot of COVID and non-COVID features (orange dots show COVID and blue dots show non-COVID features).



**Figure 6.** Original and Filtered Image (top) and second-order approximation of wavelet transform (bottom).

Compared to the MSER and the SURF features, the predictors have much larger ranges in value, which lead to higher classification scores, especially for the SVM classifier, as shown in Table 4.

**Table 4.** KNN and SVM classification accuracy results.

Type	Accuracy
Coarse KNN	61.2%
Cubic KNN	75%
Linear SVM	80.1%
Quadratic SVM	86.5%
<b>Cubic SVM</b>	<b>90.0%</b>

Out of all the investigated feature extraction methods, the Haar wavelet transforms produced a feature map which made it easier for the classifier to distinguish different classes.

The COVID-19 dataset chosen for this study is reported in [19]. The dataset contains COVID-19 images collected from hospitals in Sao Paulo, Brazil. The dataset included 1252 CT scans that were positive for SARS-CoV-2 infection and 1230 scans for patients who were non-infected by SARS-CoV-2. In total, 2482 CT scans were available. This dataset is particularly favorable as the negative COVID-19 scans do not contain signatures of other lung diseases. The data contained images that were anonymous and cannot be traced back to the patients. The images are in JPEG format and have resolutions of  $202 \times 256$ . Tests for GoogLeNet splits the data set in a 70:30 ratio (1736 training images, 745 test images). The key difference in the dataset provided by [19] is that there are significantly more samples to work with (2482) and the non-COVID samples are of healthy patients with no other conditions. The CNN was used where simulations were performed by using GoogLeNet. An epoch is a full pass through the entire dataset. During training, the accuracy of the network reached a plateau, which made it clear that the accuracy was no longer improving. The training was stopped, and once completed, the trained network was returned. The training accuracy is shown in Figure 7 and the loss rate is shown in Figure 8. Results obtained by using CT scan images are shown in Table 5.

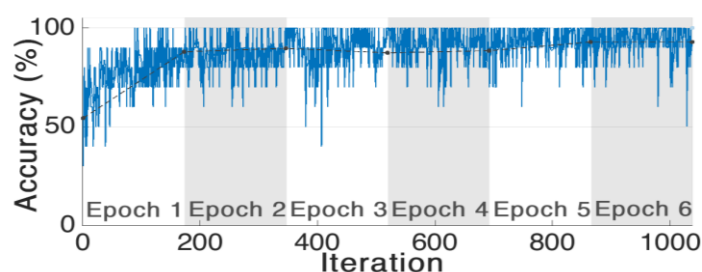


Figure 7. GoogLeNet training accuracy.

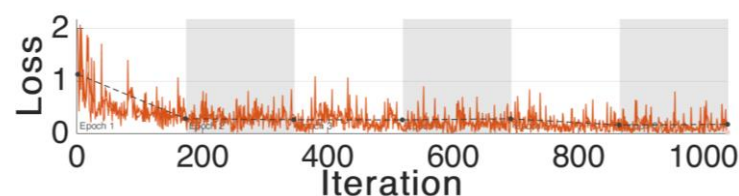


Figure 8. The loss rate.

Table 5. Training Accuracy GoogLeNet CNN with the CT scan Dataset.

Parameters	Outputs
Accuracy	95.36%
Elapsed time	7 min 57 s
GPU	GTX 960M

In order to speed up the overall processing time, it was important to reduce the dimensions of the dataset through the PCA technique stated earlier. The hardware accelerator investigated and reported in the following section required that the input data to the FPGA be minimized, thereby reducing the internal logic space required for the hardware accelerator.

### 3. Custom CNN Design and Testing for Hardware Implementation

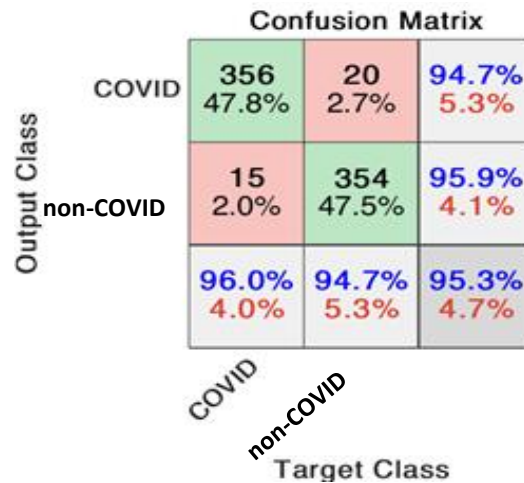
To realize a customized CNN implementation for this specific dataset, an optimized CNN model was designed where it was shown that a significant reduction in the number of layers can be achieved without a significant reduction in the overall accuracy. The structure of the investigated CNN model is shown in Table 6. Two key deep learning paradigms



were investigated: transfer learning and self-supervised learning, respectively. The most promising results were obtained from supervised learning. One of the main considerations for this case study was to investigate and choose a network that does not cause overfitting. The test results achieved through CNN-based classification are shown in Figure 9 where the column on the far right of the plot shows the percentages of all the examples predicted that belong to each class that are correctly and incorrectly classified. Convolutional feature mapping, Rectifying Linear Unit (ReLU), cross-channel normalization, and max-pooling were performed on the input image ( $40 \times 40$ ) and required features were filtered for the classification layers, as shown in Figure 10. The training parameters for the custom CNN included a maximum of 30 epochs, with a shuffle after each epoch, 48 mini-batch sizes, 70:30 split for training and testing with a validation frequency of every 60 images. An Adam optimizer was used with a learning rate of 0.001.

**Table 6.** Structure of the custom CNN.

Layer Name	Activations	Learnable Parameters
Image Input	$40 \times 40 \times 1$	N/A
Conv ( $3 \times 3$ )	$40 \times 40 \times 2$	Weights: $3 \times 3 \times 1 \times 2$ Bias: $1 \times 1 \times 2$
Batch Normal	$40 \times 40 \times 2$	Offset: $1 \times 1 \times 2$ Scale: $1 \times 1 \times 2$
ReLU	$40 \times 40 \times 2$	-
FC	$1 \times 1 \times 2$	Weights: $2 \times 3200$ Bias: $2 \times 1$
SoftMax	$1 \times 1 \times 2$	-
Class Output	-	-



**Figure 9.** Results of the  $40 \times 40$  input image based on the CNN model.

The reduction of layers in the FPGA CNN implementation, as shown in Figure 11, was performed to reduce the circuit logic needed to implement the model. Layers such as batch normalization and the softmax are used primarily for the feed-forward weight adjustment. Hence, the software model needs these layers to generate the hyperparameter values during the training stage. To optimize the classification pipeline for FPGA implementation, the layers were combined retrospectively.

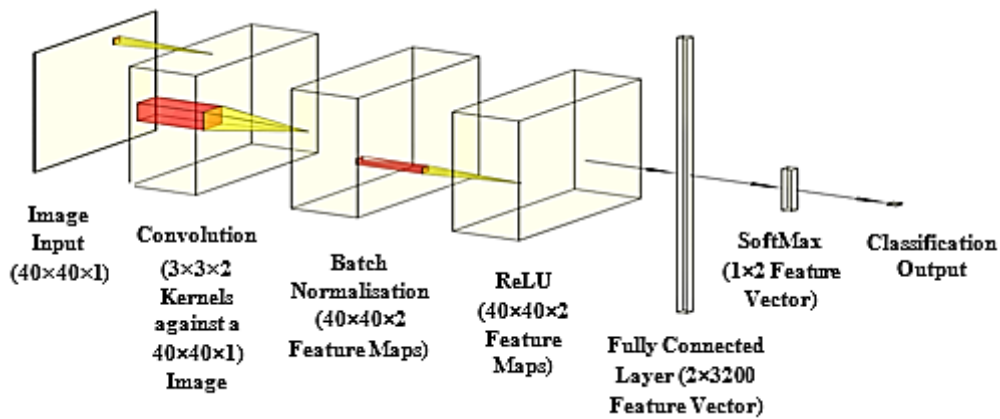


Figure 10. Software architecture of the CNN.

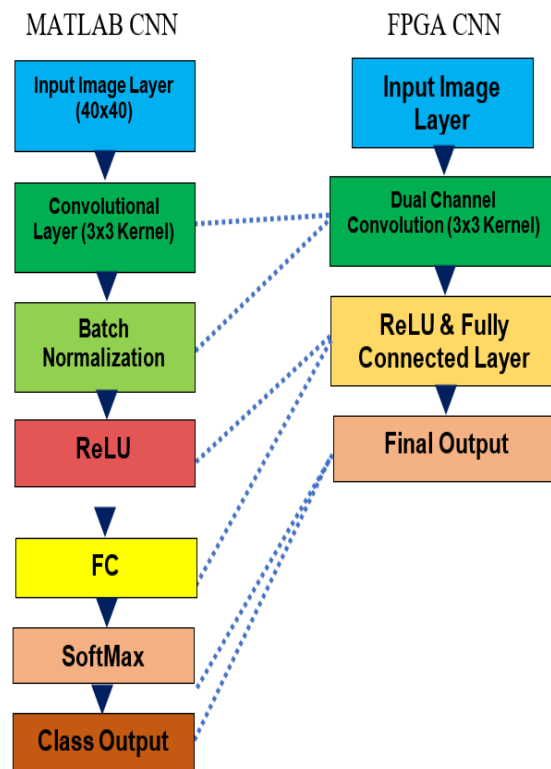


Figure 11. Customized CNN hardware implementation.

A workflow of the proposed software and hardware implementation of custom-designed CNN is shown in Figure 12, whereas Figure 13 gives an overview of the custom CNN operation. It starts by taking the fixed-point converted image data as a  $40 \times 40$  image. Then two  $3 \times 3$  filters are used as the weighted kernels. These kernels are the same as used in the FPGA implementation. The convolution process involves performing dot-matrix multiplication with both kernels. This creates two channels of feature maps for the cross-entropy classifier to determine the final prediction.

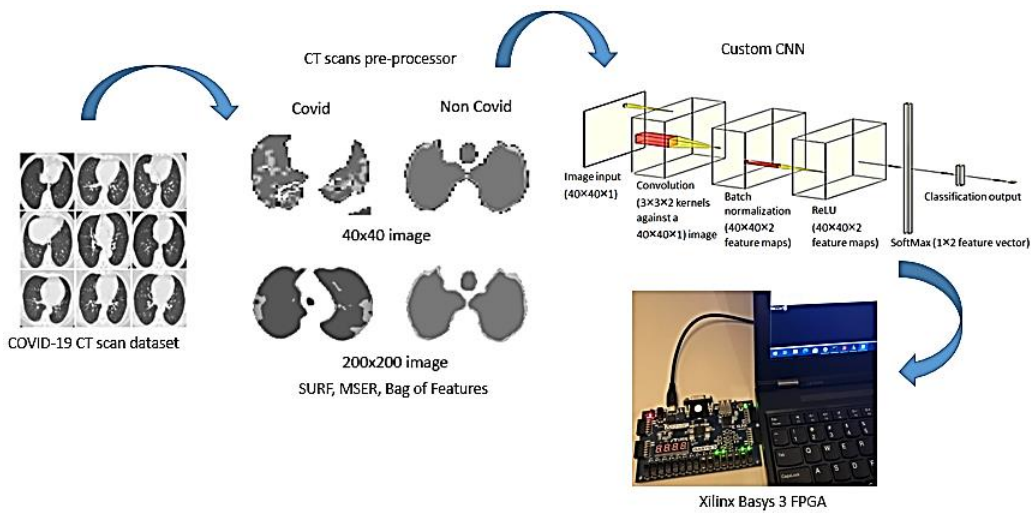


Figure 12. Workflow of the proposed CNN software and hardware implementation.

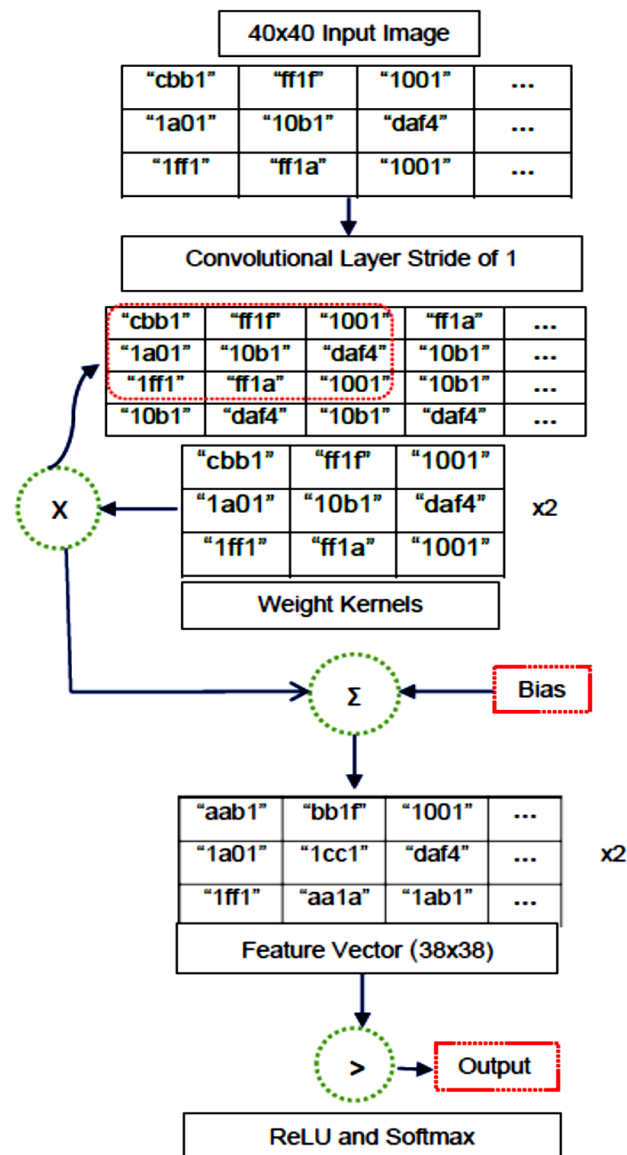


Figure 13. Layer structure and processes of the custom CNN.

A mainstream Xilinx FPGA evaluation board Basys3 was used to synthesize hardware architecture that performs similarly to the CNN designed in the software environment. Many considerations were made into hardware architecture deployment on FPGA, such as the limitations of the data that can be processed. The Basys3 FPGA board contains a 32-bit IP core with 1800 Kbits of block memory (225 KB of block RAM), 106 I/O ports, and a 100 MHz clock. These specifications indicate the data format in which the input images are converted to reduce on-chip complexity [23].

Each value in the input image was converted to a double from 8-bit integers. The precision loss was calculated from (2). Minimal data was lost with the fixed-point representation where  $n$  is the number of pixels in the image,  $x$  is the 64-bit value of the pixel,  $fp$  is the 16-bit fixed-point value of  $x$ , and  $i$  and  $j$  are indices for the position of the pixel in the array.

$$Loss = 1 - \sum_0^{n-1} \frac{x(i, j)}{fp(x)} \tag{2}$$

The image data was read as a text file and fed into the convolutional layer block. The weights were collected from a separate ROM block as well as the bias values. The feature map was produced after the matrix calculations were performed with the weights, and the summation of the bias as shown in (3) and (4).

$$a(i, j) = \sum_{n=0}^n w(i, j)x(i, j) + b \tag{3}$$

$$A \begin{cases} 1 & a(i, j) > thr, \\ 0 & otherwise, \end{cases} \tag{4}$$

These summations are then fed to the ReLU layer. Finally, the classifier block determined the class of the image based on the activation level threshold as shown in (5).

$$Class \begin{cases} 1 & A > ReLU \text{ threshold}, \\ 0 & otherwise, \end{cases} \tag{5}$$

In order to speed up classification time in retrospect to the software simulations, a hardware CNN classification pipeline was designed. A customized hardware solution was investigated which contained the Basys3 Xilinx FPGA Micro Blaze IP core to control the data being fed into the classifier. Figure 14 shows an overall system design where the class output presents the result of the classification process. The output port outputs 1 when both images are read and classified. The proposed hardware implementation includes an IP core that takes the inputs loaded into the local memory of the Micro Blaze softcore which helped in reducing the time taken for data to be loaded to the classifier.

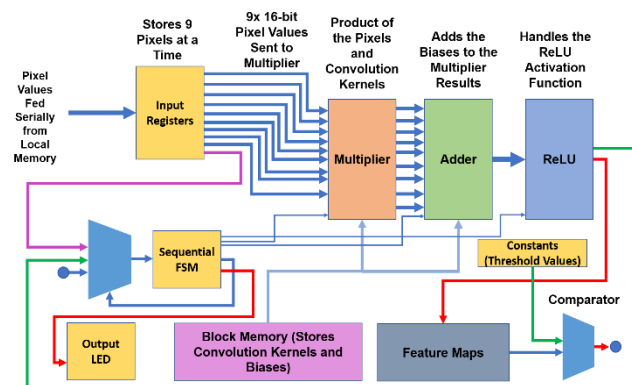


Figure 14. Overall system pipeline.

Figure 15 shows the simulated waveforms where the first two blue markers show the time taken to read the positive sample. This shows the desired output of 1 at the 'cl' signal, indicating a positive sample is present. The second to the third marker shows the waveform, which is calculating the negative sample, which is read into the classifier. The third to the last blue marker indicates the total time taken for the second sample to be classified. An output of 0 at the 'cl' signal indicates a negative sample has been detected. The simulations were performed with a 100 MHz clock frequency. The accuracy of the proposed hardware accelerator investigated in this case study was 95.3% with 0.952 precision, 94.7% sensitivity, 0.95 specificity, and 0.949 F1 score.

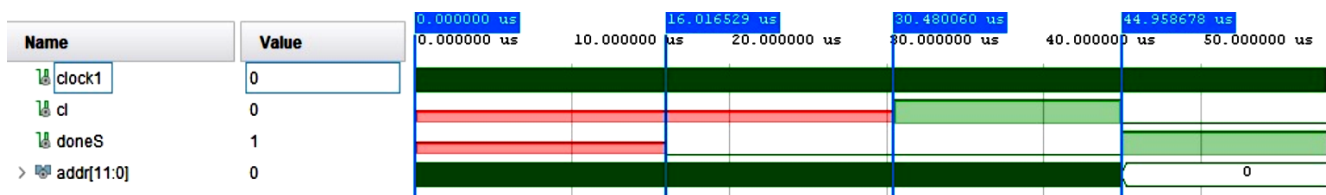


Figure 15. Simulation of the FPGA based CNN implementation.

To calculate the computational efficiency, a single image processing time on a CPU required 0.485 s, whereas implementation on an FPGA device required only 0.046 ms. Processing test images (745 images) on a CPU took 25.094 s and FPGA-based processing required 3.427 ms. This demonstrated a significant computational efficiency to process test images in real-time.

In this case study, fewer convolutional layers were used, which is an important criterion to accelerate the recognition of samples in real-time. The hardware pipeline investigated and prototyped with the COVID-19 dataset helped to optimize the area on the hardware device and reduced the thermal footprint. An overall logic utilization for the investigated FPGA-based CNN model took 32 BRAMs with 14.5 us latency and 235 milliwatts power consumption as the sum of the static and dynamic power. Some of the similar work reported in the literature included the implementation of a CNN on hardware accelerators, albeit for different applications and target devices. Hence, there was no one-to-one comparison. For example, the work reported in [23] utilized 242 BRAMs and 2.15 W power consumption, and the work reported in [24] utilized 355 BRAMs with 313 us latency. In comparison, the work reported in [25] took 162 BRAMs with 5.59 W of power consumption.

To summarize, the findings associated with the CNN involved analysis of the input data for pre-processing as outlined in Section 2 of this paper. The impact of various pre-processing methods is shown in Table 1. Simulations performed with GoogleNet are demonstrated in Figures 7 and 8, where overall accuracy was shown to be above 95% with a comparatively shallow network. Previous studies reported in medical imaging used raw images for CNN-based classification, which is a major bottleneck for real-time applications. In this paper, the authors looked at an optimized CNN model for this specific application, where a significant reduction in the number of layers was achieved without compromising on the overall classification accuracy. The software architecture of CNN included an input image convolved with a kernel followed by batch normalization and ReLU layer. A fully connected output layer was used for the classification. A unique perspective of this study involved the mapping of this customized CNN model on a reconfigurable embedded device. The results demonstrate the viability of implementing such networks on both software and hardware platforms.

As reported in [3], multiple resolutions of the same image were used and cascaded to obtain the isolated sections of GGOs in a 3D plane. The method included an object separation process for two so-called RU-networks. However, this process is resource-intensive and suffers from excessive speed reduction. The multiple resolution processes use a series of  $256 \times 256$ ,  $128 \times 128$  images, which would not be suitable for hardware implementation.

The work reported in [23] involved hardware implementation of the deep CNN with different image resolutions. With the reduced resolution of an image, the number of BRAMs synthesized on embedded reconfigurable devices decreases and, as such,  $50 \times 50$  image resolution shows greater throughput in comparison to the  $100 \times 100$  image. The FPGA-based approach proposed in this work offers a good balance between accuracy, efficient utilization of hardware resources, and real-time processing.

#### 4. Conclusions

In this paper, the authors demonstrated a viable solution for the rapid diagnosis of Covid-19 CT scan images using a custom-designed CNN. This study provides a wider range of design options and demonstrates the potential of reducing total slice count and power dissipation. The proposed work offers a scope and has great potential in optimizing the CNN pipeline by using the approximations of feature selection. A significant reduction in classification time was achieved by developing a hardware accelerator. The thermal footprint of the proposed architecture is much smaller with a power consumption of 235 milliwatts at 100 MHz. The investigated CNN has a much simpler structure that could be utilized for similar tasks in computer vision-based applications. The reconfigurability of the proposed hardware (FPGA) solution allows for changes in its functionality to be made in a short space of time. By increasing the layers of the CNN, the architecture could be adopted for a deeper CNN pipeline and applied for other applications including imaging-based health-related diagnoses. As most conventional medical-ML solutions heavily depend on cloud computing as their backend infrastructure to run their CNN model. However, many hospitals do not have access to such infrastructure due to financial limitations or lack of human expertise. This paper present it is possible to use FPGA as an edge device to execute the CNN model.

**Author Contributions:** A.G. (conceptualization, formal analysis, funding acquisition, supervision, validation, and writing). A.A. (investigation, methodology, software, and writing). C.H.S. (formal analysis, writing). H.Y. (formal analysis, writing). S.K. (formal analysis, writing). All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the American University of Ras Al Khaimah via research grant number ENGR/007/22, United Arab Emirates.

**Data Availability Statement:** SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification [19].

**Acknowledgments:** We acknowledge the hardware/software support provided by the American University of Ras al Khaimah, UAE. Authors also acknowledge the British Council “2019 UK-China-BRI Countries Partnership Initiative” programme, with project titled “Adapting to Industry 4.0 oriented International Education and Research Collaboration.

**Conflicts of Interest:** The authors declare no conflict of interest.

#### References

1. Waheed, A.; Goyal, M.; Gupta, D.; Khanna, A.; Al-Turjman, F.; Pinheiro, P. CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection. *IEEE Access* **2020**, *8*, 91916–91923. [[CrossRef](#)]
2. He, X. Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans. *IEEE Trans. Med. Imaging* **2020**. [[CrossRef](#)]
3. Xie, W.; Jacobs, C.; Charbonnier, J.; van Ginneken, B. Relational Modeling for Robust and Efficient Pulmonary Lobe Segmentation in CT Scans. *IEEE Trans. Med. Imaging* **2020**, *39*, 2664–2675. [[CrossRef](#)] [[PubMed](#)]
4. Dinelli, G.; Meoni, G.; Rapuano, E.; Benelli, G.; Fanucci, L. An FPGA-Based Hardware Accelerator for CNNs Using On-Chip Memories Only: Design and Benchmarking with Intel Movidius Neural Compute Stick. *Int. J. Reconfigurable Comput.* **2019**, *2019*, 7218758. [[CrossRef](#)]
5. Zhao, R.; Luk, W.; Niu, X.; Shi, H.; Wang, H. Hardware Acceleration for Machine Learning. In Proceedings of the 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Bochum, Germany, 3–5 July 2017; pp. 645–650. [[CrossRef](#)]
6. Ghani, A.; See, C.H.; Sudhakaran, V.; Ahmad, J.; Abd-Alhameed, R. Accelerating Retinal Fundus Image Classification Using Artificial Neural Networks (ANNs) and Reconfigurable Hardware (FPGA). *Electronics* **2019**, *8*, 1522. [[CrossRef](#)]
7. Ghani, A. Healthcare electronics—A step closer to future smart cities. *ICT Express* **2018**, *5*, 256–260. [[CrossRef](#)]



8. Khan, S.Q.; Ghani, A.; Khurram, M. Population coding for Neuromorphic hardware. *Neurocomputing* **2017**, *239*, 153–164. [[CrossRef](#)]
9. Dass, J.; Narawane, Y.; Mahapatra, R.; Sarin, V. Distributed Training of Support Vector Machine on a Multiple-FPGA System. *IEEE Trans. Comput.* **2020**, *69*, 1015–1026. [[CrossRef](#)]
10. Sayed, R.; Azmi, H.; Nassar, A.; Shawkey, H. Design Automation and Implementation of Machine Learning Classifier Chips. *IEEE Access* **2020**, *8*, 192155–192164. [[CrossRef](#)]
11. Zhou, S.; Kannan, R.; Prasanna, V. Accelerating Stochastic Gradient Descent Based Matrix Factorization on FPGA. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1897–1911. [[CrossRef](#)]
12. Xu, X.; Chen, Y.; Zhu, K.; Yang, J.; Tan, Z.; Luo, M. Research on FPGA Pulse Laser Ranging Method Based on Deep Learning. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–11. [[CrossRef](#)]
13. Azghadi, M.; Lammie, C.; Eshraghian, J.K.; Payvand, M.; Donati, E.; Linares-Barranco, B.; Indiveri, G. Hardware Implementation of Deep Network Accelerators Towards Healthcare and Biomedical Applications. *IEEE Trans. Biomed. Circuits Syst.* **2020**, *14*, 1138–1159. [[CrossRef](#)] [[PubMed](#)]
14. Jain, A.; Goel, P.; Aggarwal, S.; Fell, A.; Anand, S. Symmetric  $k$ -Means for Deep Neural Network Compression and Hardware Acceleration on FPGAs. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 737–749. [[CrossRef](#)]
15. Li, S.; Luo, Y.; Sun, K.; Yadav, N.; Choi, K. A Novel FPGA Accelerator Design for Real-Time and Ultra-Low Power Deep Convolutional Neural Networks Compared With Titan X GPU. *IEEE Access* **2020**, *8*, 105455–105471. [[CrossRef](#)]
16. Khan, S.D.; Alarabi, L.; Basalamah, S. Toward Smart Lockdown: A Novel Approach for COVID-19 Hotspots Prediction Using a Deep Hybrid Neural Network. *Computers* **2020**, *9*, 99. [[CrossRef](#)]
17. Ai, T.; Yang, Z.; Hou, H.; Zhan, C.; Chen, C.; Lv, W.; Tao, Q.; Sun, Z.; Xia, L. Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases. *Radiology* **2020**, *296*, E32–E40. [[CrossRef](#)]
18. Jiang, X.; Yin, Z.; Wang, T.; Zhai, N.; Lu, F.; Zhan, C.; Han, Q.; Feng, C. COVID-19 Dynamic Computed Tomography (CT) Performance and Observation of Some Laboratory Indicators. *Med. Sci. Monit.* **2020**, *26*, e924403. [[CrossRef](#)]
19. Soares, E.; Angelov, P.; Biaso, S.; Froes, M.H.; Abe, D.K. SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification. *MedRxiv* **2020**. [[CrossRef](#)]
20. Ker, J.; Singh, S.P.; Bai, Y.; Rao, J.; Lim, T.; Wang, L. Image Thresholding Improves 3-Dimensional Convolutional Neural Network Diagnosis of Different Acute Brain Hemorrhages on Computed Tomography Scans. *Sensors* **2019**, *19*, 2167. [[CrossRef](#)]
21. Uk.mathworks.com. 2020. Get Started with Transfer Learning-MATLAB & Simulink-Mathworks United Kingdom. Available online: <https://uk.mathworks.com/help/deeplearning/gs/get-started-with-transfer-learning.html> (accessed on 25 May 2020).
22. Oh, Y.; Park, S.; Ye, J. Deep Learning COVID-19 Features on CXR Using Limited Training Data Sets. *IEEE Trans. Med. Imaging* **2020**, *39*, 2688–2700. [[CrossRef](#)]
23. Geng, T.; Wang, T.; Sanauallah, A.; Yang, C.; Patel, R.; Herbordt, M. A Framework for Acceleration of CNN Training on Deeply-Pipelined FPGA Clusters with Work and Weight Load Balancing. In Proceedings of the 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, Ireland, 27–31 August 2018; p. 394. [[CrossRef](#)]
24. Shah, N.; Chaudhari, P.; Varghese, K. Runtime Programmable and Memory Bandwidth Optimized FPGA-Based Coprocessor for Deep Convolutional Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 5922–5934. [[CrossRef](#)] [[PubMed](#)]
25. Li, S.; Sun, K.; Luo, Y.; Yadav, N.; Choi, K. Novel CNN-Based AP2D-Net Accelerator: An Area and Power Efficient Solution for Real-Time Applications on Mobile FPGA. *Electronics* **2020**, *9*, 832. [[CrossRef](#)]