TEKNOFEST

AEROSPACE AND TECHNOLOGY FESTIVAL

ARTIFICIAL INTELLIGENCE IN HEALTHCARE COMPETITION

(Category of Disease Detection for Abdominal area via Computer Vision)

PROJECT DETAIL REPORT

TEAM NAME:

FİZGET-CODEMEDICAL TEAM

TEAM ID:

419522

Table of Contents

Tables

Graphs

Figures

1. Project Current Statement Assessment

1.1. PPR Review

In this stage, the PPR Referee Evaluation was considered. To begin with the strength points, the literature research was improved and strengthened by including more details in our work. The teamwork was proceeded effectively in accordance with our project plan stated in the PPR. This report was prepared by following the "Writing Standards and Rules" specified in the specification in each step. The weak points were evaluated and discussed in detail by the team to get rid of the problems and complete the missing parts. In this context, first, the recommendation in the PPR Referee Evaluation of "**The algorithms planned to be used could be supported by newer architectures. In terms of specificity, a new hybrid algorithm could be proposed.**", literature research was performed and among various choices a new architecture was developed for our model to assist the used algorithm. Second, for the criticism "**Pulmonary embolism is not an intra-abdominal tomographic diagnosis**", this mistake was recognized and understood by the team, and more attention is being paid while doing any literature review. For the last recommendation, "**The method is not given in details. The authors do not provide any insights on the connection between the problem (Abdomen CT lesion detection) and their proposed solution. The report does not include the details of the originality of the proposed method. How and why part of the originality is missing.**", the used method is now given in more detail, the solution is provided by the team regarding problem as given in the PDR report, and the new method is proposed for satisfying the originality requirement. In addition, for the fastest and most efficient results, the programming software was modified in accordance with the project's methodology. Lastly, we would like to thank the referees for their valuable criticism, suggestions, and contributions to our project.

1.2. Current Model Review

The Convolutional Neural Network (CNN) algorithm model is planned to be used in this project for the capture via training the variables from datasets for abdominal inflammation detection. Various algorithms, such as Elman Neural Network (ENN), Back Propagation Artificial Neural Networks (BPANN), and Deep Learning (DL) based models, have provided solutions with some limitations based on the type and number of datasets. However, a portion of the territories where CNNs are broadly utilized are image recognition, image classification, image captioning and object detection. The key chore of the neural network is to make sure it processes all the layers, and hence detects all the underlying features, automatically. CNN is designed to function in much the same mechanism as the neurons in the human brain [1]. Also, CNN detects and analyzes images via numerical operations.

To approach the model, initially, the CNN algorithm used in our model contains components which typically has three layers: a convolutional layer, pooling layer, and fully connected layer as shown in Figure 1.

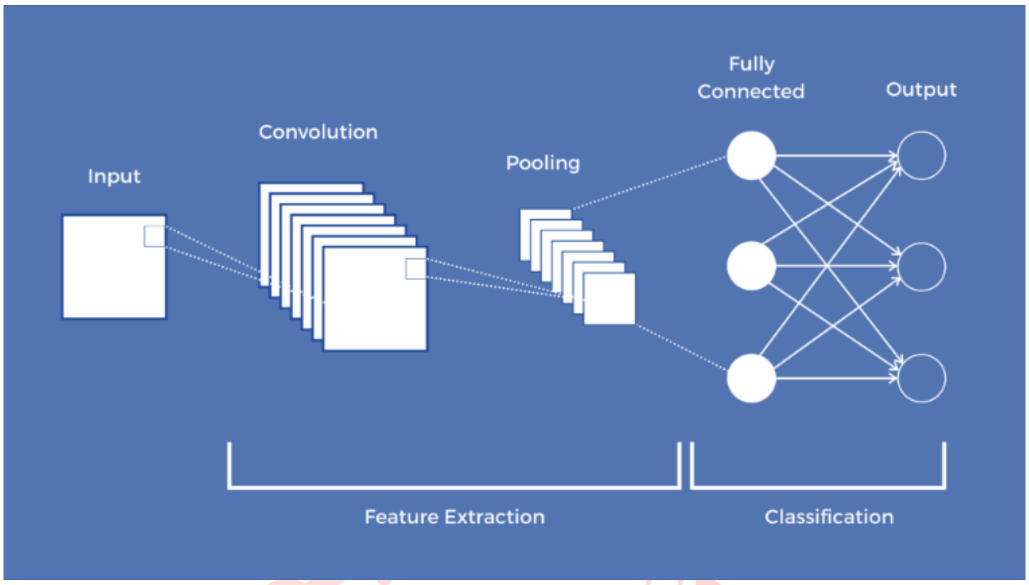


Figure 1. CNN Components: The first two layers (convolution and pooling) are responsible for feature extraction from the images, so both are also collectively referred to as feature extracting layers, while the last fully connected layer is responsible for classifying the image per the task at hand, so also called the classification layer.

In the Input Image the CNN layers are programmed to identify simpler patterns as lines and curves before progressing to more complex patterns as faces and objects. Hence, it is plausible to claim that using a CNN may provide vision to computers. The convolutional layer receives an image tensor as an input, applies a specific number of convolutional filters (Kernels) on the image tensor, adds a bias and applies a non-linear activation function (typically, ReLU) to the output. The objective of convolutional layers is to extract patterns and information from an image. The Convolutional filters/kernels at the starting of the network are responsible for capturing the low-level features such as color, gradient orientation, etc. The convolutional filters/kernels deeper down the network are responsible for capturing the high-level features such as edges in the image. The pooling layer is responsible for performing a series of pooling operations on an image. It receives an image tensor as an input and produces a tensor after applying the specified pooling operation. This helps in reducing the computational costs and make the network more generic. In Fully Connected Layer (FCL), neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular Fully Connected Neural Network (FCNN). Therefore, it can be computed as usual by a matrix multiplication followed by a bias effect [2].

1.3. Training Software and Environment

MATLAB Program is planned to be used in our model for the training and test processes in accordance with the programming experience that the team possess. MATLAB provides interactive tools that make it easy to perform a variety of machine learning tasks, including connecting to and importing data. Apps can generate MATLAB code, enabling us to automate tasks. Oftentimes, data has missing or incorrect values. Functions for finding, removing, and cleaning data enable us to get our data ready for analysis. In MATLAB, Computer Vision (Image Processing) and Neural Network Toolbox provide additional algorithms that guide us through the process of training and testing neural networks [3].

2. Originality

Various models such as DiceNet, AlexNet, VGG, Inception, and ResNet was examined on the architectures used with CNN for computer vision. Instead of using well-known architectures, a new single model classifier **FizNet** was developed for the detection of abdominal diseases (See Figure 2). Our developed model FizNet is based on the architecture of DiceNet [4]. Furthermore, to get better detection accuracy from the fed datasets, addition and ordering of CNN layers in DiceNet was performed. This alteration is the main originality of our model.

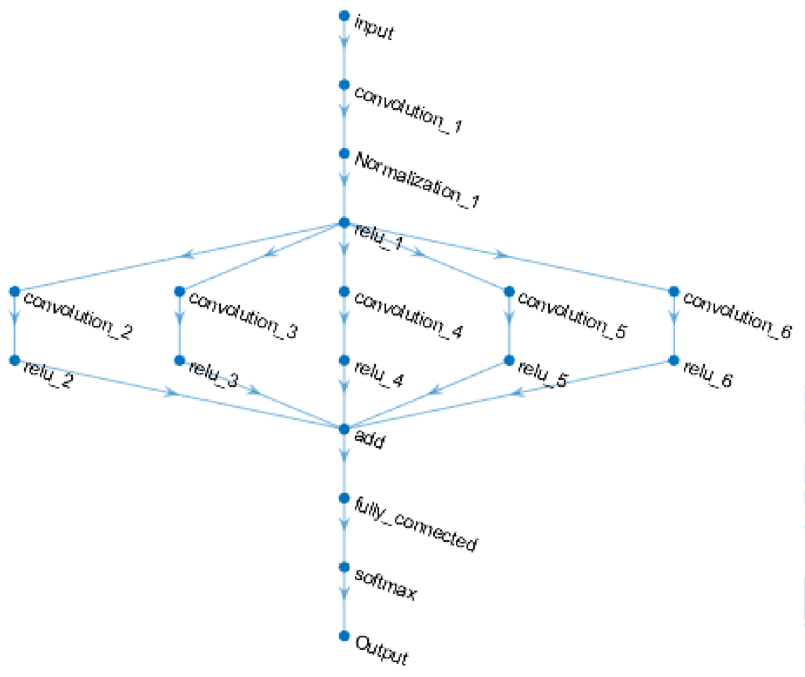


Figure 2. FizNet Architecture: FizNet contains the following layers: Image Input Layer, 6 Convolution 2D-Layer, Batch Normalization Layer, 6 Rectified Linear Units (ReLU) Layer, Addition Layer, Fully Connected Layer, SoftMax Layer, and Image Output Layer (Classification Layer).

- **Image Input Layer:** An image input layer inputs 2-D images to a network and applies data normalization [5]. It can be described as the following:

layer = imageInputLayer(inputSize,Name,Value)

This sets the optional properties using name-value pairs, then it can be easy to specify multiple name-value pairs and to enclose each property name in single quotes.

- **Convolution 2D-Layer:** A 2-D convolutional layer applies sliding convolutional filters to 2-D input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term [6].

layer = convolution2dLayer(filterSize,numFilters)

This creates a 2-D convolutional layer and sets the FilterSize and NumFilters properties.

- **Batch Normalization Layer:** A batch normalization layer normalizes a mini batch of data across all observations for each channel independently [7]. To speed up training of the convolutional neural network and reduce the sensitivity to network initialization, use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers. After normalization, the layer scales the input with a learnable scale factor γ and shifts it by a learnable offset β.

  layer = batchNormalizationLayer(Name,Value)

- **ReLU Layer:** A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero [8]. This operation is equivalent to

$$f(x) = \begin{cases} x\,, & x \geq 0 \\ 0\,, & x < 0 \end{cases}$$

  layer = reluLayer('Name',Name)

This creates a ReLU layer and sets the optional Name property using a name-value pair. For example, reluLayer('Name','relu1') creates a ReLU layer with the name 'relu1'.

- **Addition Layer:** An addition layer adds inputs from multiple neural network layers elementwise.[9]

  layer = additionLayer(numInputs,'Name',name)

This creates an addition layer that adds numInputs inputs elementwise. It also sets the Name property.

- **Fully Connected Layer:** A fully connected layer multiplies the input by a weight matrix and then adds a bias vector [10]. After it returns a fully connected layer and specifies the Output Size property.

  layer = fullyConnectedLayer(outputSize,Name,Value)

This sets the optional Parameters and Initialization, Learning Rate and Regularization, and Name properties using name-value pairs. For example,

- **Softmax Layer:** A softmax layer applies a softmax function to the input [11].

  layer = softmaxLayer('Name',Name)

This creates a softmax layer and sets the optional Name property using a name-value pair. For example, softmaxLayer('Name','sm1') creates a softmax layer with the name 'sm1'. Enclose the property name in single quotes.

- **Classification (Output) Layer:** A classification layer computes the cross-entropy loss for classification and weighted classification tasks with mutually exclusive classes [12]. The layer infers the number of classes from the output size of the previous layer. For example, to specify the number of classes K of the network, you can include a fully connected layer with output size K and a softmax layer before the classification layer.

```
layer = classificationLayer(Name,Value)
```

This sets the optional Name, ClassWeights, and Classes properties using one or more name-value pairs; e.g., classificationLayer('Name','output'). This creates a classification layer with the name 'output'.

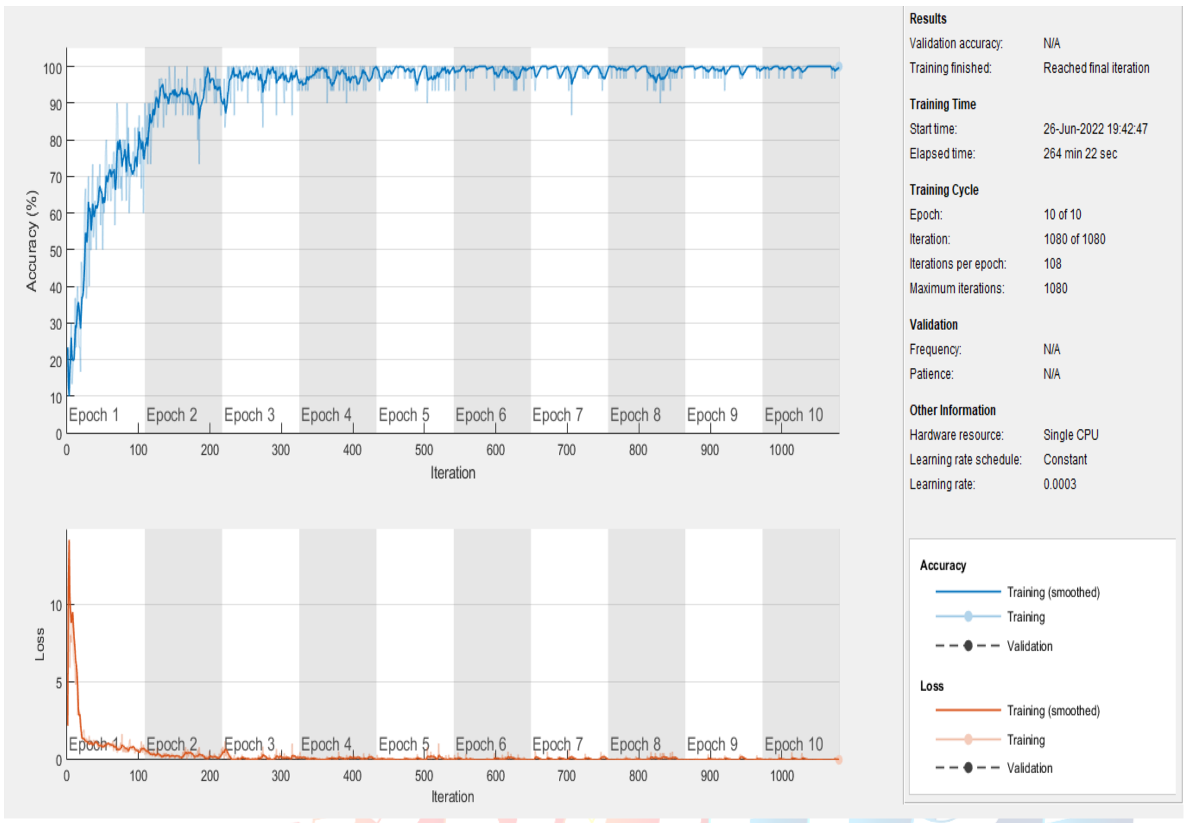
Accordingly, the tools in the used neural layers in FizNet are as the following:

```
inputSize = [512 512 1];
numClasses = 6;
layers = [
   imageInputLayer(inputSize,'Name','input')
   convolution2dLayer(5,15,'Name','convolution_1')
   batchNormalizationLayer('Name','Normalization_1')
   reluLayer('Name','relu_1')

     convolution2dLayer(1,15,'Name','convolution_2');
   reluLayer('Name','relu_2')
        convolution2dLayer(1,15,'Name','convolution_3');
   reluLayer('Name','relu_3')
     convolution2dLayer(1,15,'Name','convolution_4');
   reluLayer('Name','relu_4')
     convolution2dLayer(1,15,'Name','convolution_5');
   reluLayer('Name','relu_5')
     convolution2dLayer(1,15,'Name','convolution_6');
   reluLayer('Name','relu_6')

   additionLayer(5,'Name','add')

   fullyConnectedLayer(numClasses,'Name','fully_connected')
   softmaxLayer('Name','softmax')
   classificationLayer('Name','Output')];
```

Since the expected work is to detect diseases in the abdominal area via computer tomography (CT), a developed architecture is needed to produce better results and detection. In this context, the DiceNet model was examined and reviewed. The DiceNet model has an architecture including some layers that construct a linearly ordered architecture in which it contains DimConv 3x3 convolutional layers. Therefore, based on the DiceNet architecture, layer additions and enhancements have been made to the new model, FizNet. In addition, Convolutional layers, ReLU layers, Normalization layers, and Addition layers were inserted into the algorithm (See Figure 2). This insertion enhanced the algorithm's structure and helped yield an efficient result.

3. Results and Review

After the construction of the FizNet model and its architecture was completed, the selected datasets were classified: 90% of the datasets are used for training and 10% for testing (See Table 5). The training and testing progress yield expected results with good accuracy and other criteria through our model, Graph 1.



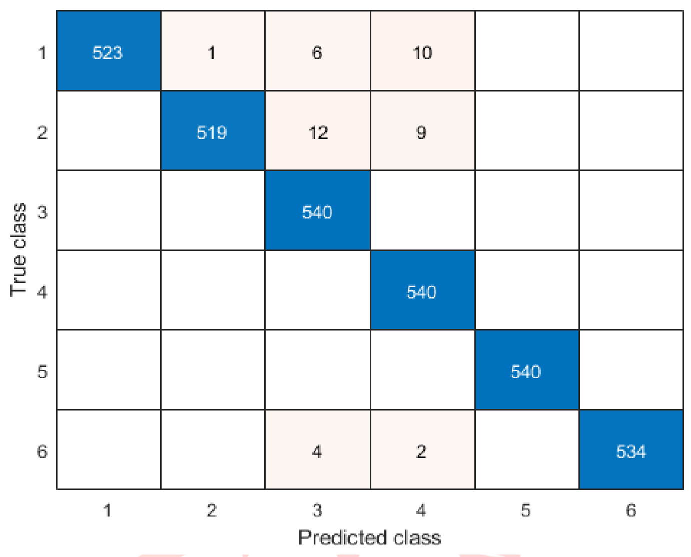
Results	
Validation accuracy:	N/A
Training finished:	Reached final iteration
Training Time	
Start time:	26-Jun-2022 19:42:47
Elapsed time:	264 min 22 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	1080 of 1080
Iterations per epoch:	108
Maximum iterations:	1080
Validation	
Frequency:	N/A
Patience:	N/A
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.0003

Graph 1. Training Progress

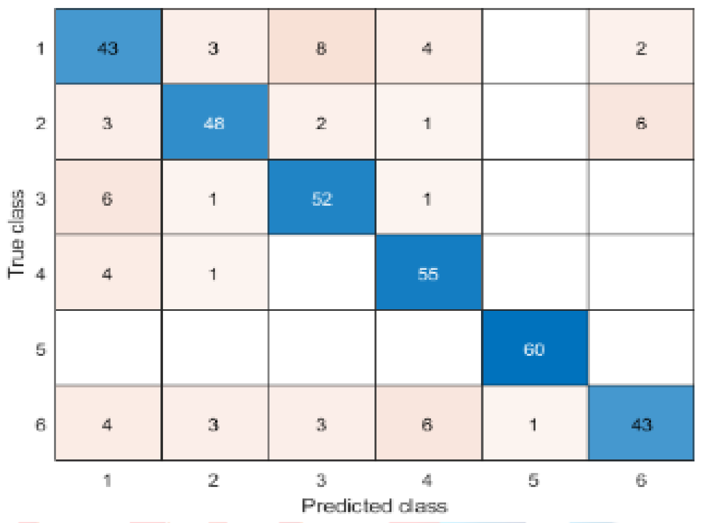
Furthermore, for training and testing, the Confusion Matrix (CM) was used to know the performance of the model classification. It gives a comparison between Actual and Predicted values. The CM is a N x N matrix, where N is the number of classes or outputs. It indicates a certain matrix such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) that are essential in calculating the accuracy and F1-score in our model using the following formulas [13]:

Accuracy = (TP+TN)/(TP+TN+FP+FN).
F1-score = (2TP)/(2TP+ FP+FN).

In our model, each class in CM indicates the tags of the output diseases in a 6 x 6 matrix form, as shown in Graph 2 and 3. In the training progress, the class 1 x 1, for instance, shows that out of 540 images, the model has read and analyzed 523 correctly (TP) and missed only 17 (FP), as shown in Graph 2. Moreover, the 4 x 4 and 5 x 5 classes give a 100% accurate result with 540 correct readings and analysis (TP). Correspondingly, the testing progress is demonstrated in the same manner as shown in Graph 3.



Graph 2. Training, True-Predicted Matrix



Graph 3. Testing, True-Predicted Matrix

The datasets provided to the model for training resulted in 98.64% accuracy, as shown in Table 1. The datasets which fed into testing the model provided two results. One, with an accuracy of 97%, is from trained datasets, and the other is from untrained datasets, with an accuracy of 83.61%. Since the model expectation is to function with untrained datasets for the test, the second result (83.61%) was considered, as shown in Table 2.

Table 1. Training Results	
Total datasets	3240
True	3196
False	44
F1 score	0.9864
IoU	0.9734
Accuracy	98.64 %

Table 2. Testing Results

Total datasets	360
True	301
False	59
F1 score	0.8348
IoU	0.7261
Accuracy	83.61 %

As a result, the applied methods of the original model are compatible with the attained outcomes. Indeed, some enhancement can be done in the model without changing its structure for better accuracy in results than the one obtained.

4. Datasets Used in The Experimental and Training Stages

The dataset used in the testing and training stages was received from the Ministry of Health (TUSEB) via TEKNOFEST, from which the required permission was obtained. The 1050 datasets in DICOM format have certain classifications, as given in the Data.xlsx file. The received folders contained the labeled datasets defined in the specification with an additional dataset for training, as presented in Table 3.

Table 3. Disarranged Datasets

The Label Name Given in Excel	Class	Slices Number
Abdominal aorta anevrizma (Abdominal aorta)	6	1426
Abdominal aorta diseksiyon (Abdominal aorta)	6	0
Abdominal aorta	-	77588
Akut apandisit ile uyumlu (Apandiks)	1	53388
Akut divertikülit ile uyumlu (kolon)	5	1076
Akut kolesistit ile uyumlu (Safra kesesi)	2	28683
Akut pankreatit ile uyumlu (Pankreas)	3	25034
Apandiks	-	0
Apendikolit	Ek	0
Böbrek taşı	4	15308
Böbrek- mesane	-	46206
Kalsifiye divertikül	Ek	572
Kolon	-	64413
Pankreas	-	12755
Safra kesesi	-	6878
Safra kesesi taşı	Ek	4093
Üreter taşı	4	16711

Since the datasets are in DICOM format and have 79 GB of memory, compression was needed. So, the format of the images was converted to PNG for better image handling. Also, this has reduced the memory of the data to approximately 26 GB. The datasets were reclassified and reordered by keeping the labeled diseases as donated in the specification. This classification yields a clearer and more readable output for our program and helps us get fast and accurate results for our model, as shown in Table 4.

Table 4. Arranged Datasets

The Label Name Given in Excel	Class	Slices Number
Akut apandisit ile uyumlu (Apandiks)	1	53388
Akut kolesistit ile uyumlu (Safra kesesi)	2	28683
Akut pankreatit ile uyumlu (Pankreas)	3	25034
Böbrek taşı	4	15308
Üreter taşı	4	16711
Akut divertikülit ile uyumlu (kolon)	5	1076
Abdominal aorta anevrizma (Abdominal aorta)	6	1426
Abdominal aorta diseksiyon (Abdominal aorta)	6	0
Abdominal aorta	6	77588

Among the datasets, a random sample of 3600 slices with their label name and class was chosen for training and testing (Table 5).

Table 5. Training and Test Datasets

The Label Name Given in Excel	Class	Slices Number
Akut apandisit ile uyumlu (Apandiks)	1	600
Akut kolesistit ile uyumlu (Safra kesesi)	2	600
Akut pankreatit ile uyumlu (Pankreas)	3	600
Böbrek taşı	4	300
Üreter taşı	4	300
Akut divertikülit ile uyumlu (kolon)	5	600
Abdominal aorta anevrizma (Abdominal aorta)	6	300
Abdominal aorta diseksiyon (Abdominal aorta)	6	0
Abdominal aorta	6	300
Total **Trained data = 90% = 3240** **Tested data = 10% = 360**		**3600**

5. References

[1] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., . . . Ghayvat, H. (2021, October 11). CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. electronics, 28. doi:https://doi.org/10.3390/electronics10202470

[2] Sharma, P. (2022). Basic Introduction to Convolutional Neural Network in Deep Learning. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/#h2_1

[3] MathWorks. (n.d.). Introducing Deep Learning with MATLAB. Retrieved from https://www.mathworks.com/campaigns/offers/next/deep-learning-ebook.html (Accessed June 09, 2022).

[4] S. Mehta, H. Hajishirzi and M. Rastegari, "DiCENet: Dimension-Wise Convolutions for Efficient Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 5, pp. 2416-2425, 1 May 2022, doi: 10.1109/TPAMI.2020.3041871.

[5] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html;jsessionid=59a722f8a79677af613a1f6ed695 (Accessed June 15, 2022).

[6] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html (Accessed June 15, 2022).

[7] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html, (Accessed June 15, 2022).

[8] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.additionlayer.html (Accessed June 15, 2022).

[9] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.relulayer.html, (Accessed June 15, 2022).

[10] https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html, (Accessed June 15, 2022).

[11]https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html (Accessed June 15, 2022).

[12] https://www.mathworks.com/help/deeplearning/ref/classificationlayer.html, (Accessed June 15, 2022).

[13] Wikipedia contributors, "Confusion matrix," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=1088688630 (Accessed June 21, 2022).