Chess Image Clasification

Babita Chaini
School of Computing
National College of Ireland
Dublin, Ireland
x21139211@student.ncirl.ie

Komal
School of Computing
National College of Ireland
Dublin, Ireland
x21148082@student.ncirl.ie

Nitish Sharma
School of Computing
National College of Ireland
Dublin, Ireland
x21154147@student.ncirl.ie

Rajbharath Jothimani
School of Computing
National College of Ireland
Dublin, Ireland
x21133000@student.ncirl.ie

Abstract—**A significant technological hurdle is the automatic digitization of chess games using computer vision. This topic is of great interest to tournament organizers as well as amateur or expert players who want to stream or use chess engines to evaluate their over-the-board (OTB) internet games.Although prior research has yielded encouraging results, there is still room for advancement in the recognition rate and latency of cutting-edge techniques in order to enable their widespread and practical use.Our features in addition to a faster detection technique for the chessboard.As a result, we examined various Convolutional neural networks were utilized to classify chess pieces, and we found the most effective technique to integrate them into our embedded platform. Notably, we have created a helpful framework that, with 81 percent accuracy, quickly digitizes a chess position from an image. classification accuracy for the pieces. This paper describes a method that takes a chess piece-containing image as input and outputs as classification of those pieces.The remainder of this study will first review earlier chess recognition work, then showcase an unique improvement to a part of the existing algorithm, explain the technique in technical terms, go over the experimental setup and findings, and finally give conclusions. Index Terms - over-the-board, convolutional, neural, accuracy, algorithm**

I. INTRODUCTION

The use of image classification techniques has advanced astonishingly thanks to recent developments in deep neural networks. For this objective, many Convolutional neural network (CNNs) have been used, including MobileNet, Xception,Resnet and VGG16. These CNNs have undergone ongoing research and development in order to perform effectively even for massive image categorization.

Nevertheless, an open computer vision problem is the accurate identification of chess pieces images. Its method will help seasoned players who like to use a physical chessboard but want to study and develop using chess algorithms and other specialist software programs. It will also be helpful for amateur athletes who want to post their OTB games with friends and for chess event organizers who want to stream OTB matches online.

1) Research Questions:

- Can we do multi class classification of the chessmen images into appropriate chess piece groups such as rook, king, queen, Bishop, Knight and Pawn ?
- Can we forecast the best neural network method based on accuracy to classify the images ?

II. RELATED WORK

Research on Chess image recognition started from the developments of chess robots who detects the human opponent moves with the help of cameras. As stated in the research report, these robots typically employ the three-way classification technique to detect each square's occupancy and (if inhabited) Figure colors. [1]. Additionally, as per [2] Several methods use the same method to record chess moves from video footage. Any such three-way classification method, however, has to know the previous board state in order to predict the final move (based on the expected occupancy and piece color of each square) and determine the current chess position. While a chess robot or move recording computer software can quickly access this information, a chess recognition system that only accepts a single still image as input cannot. Additionally, these methods fail to distinguish advanced pieces and are unable to recover once a single move was predicted incorrectly.

In the last five years, a number of methods have been developed to address the issue of chess recognition from a single image by categorizing each bit sort (pawn, knight, bishop, rook, queen, and king) and color. The input image may occasionally be shot at an oblique angle to the board because chess pieces are almost indistinguishable from a bird's-eye perspective. While, as noted in academic papers [3] For piece classification, it also depends on scale-invariant feature rework (SIFT) and bar graph of adjusted gradients (HOG) feature descriptors, moreover [4] and [5] asserts that they are insufficient due to the comparable texture of chess pieces and that one should instead apply guide matching to the outlines of the pieces. However, [4] Change the board's colors from black and white to red and green in order to fix the problem (similar changes are also being considered for other systems). ([6], [7]), However, any such change places unreasonably tight restrictions on conventional chess games.

CNNs are used in a variety of other recognition methods at various phases of the procedure. Comparison of CNNs and template matching was conducted by [8], however, they only discovered a slight increase in accuracy (although they used only 40 images per class). a localization algorithm for the chessboard designed by [9] produces heatmaps with an accuracy of 95 percentage that show how probable each pixel is to represent a piece of the chessboard. The corner locations identified by the heatmap are then further refined using a CNN, outperforming the outcomes. obtained by [10]. Then, they contrast an SVM-based solution with a CNN-based piece classification system proposed by [3] andfind no major amelioratio, but succeed in acquiring ameliorations by analyzing likely and legitimate chess positions. the well-liked AlexNet CNN architecture([11], [9] created an app for augmented reality. Their CNN was trained to discriminate between 13 classes using a dataset with 200 samples per class (six types of colored pieces, and the empty square). To the best of our knowledge, their end-to-end pipeline achieves a total accuracy of 93.45 percent (corresponding 6.55 percent per-square error rate), which represents the state-of-the-art. Any chess recognition system must be capable of detecting the location of the chessboard in order to locate the chessboard and each of the 64 squares.

The four corner points are defined, and then all that is needed is a straightforward perspective transformation to locate the squares in photographs taken from a bird's-eye viewpoint. Systems that require the user to interactively choose the four corner points can get around this issue.([2], [3], But in a perfect world, chess recognition software would be able to understand board positions on its own. a corner detector by Harris ([6], [12]) The most typical method for automatic chess grid detection is J, or a variation of J. Despite the usage of flood fill and template matching [7], According to study, different imaging approaches have been utilized to image 2021, 7, 943 of 16 line detectors based on the Hough transform. paper([13], [4]). Because they can recognize grid corners when they are obscured by pieces, line-based detection algorithms are preferred to corner-based ones. These algorithms frequently make use of the chessboard's geometric features to compute a perspective transformation of the grid lines that most nearly resembles the identified lines. ([5], [14]).

There are currently no suitable datasets for chess detection (particularly at the size required for deep learning), which is seen as an issue by many. ([9], [3]) . To this purpose, the synthesis of training data from 3D models appears to be a potential method to quickly produce big datasets without requiring manual annotation. [15] easily create point-cloud data for volumetric CNNs using 3D chess models, and [16] uses his 3D models' representations as input. Hou describes a non-collapsible chessboard identification system utilizing a rudimentary neural network that only reaches 72 percent accuracy, and it only functions when the chessboard is taken using a depth camera.

III. METHODOLOGY

In this study KDD methology is being used.The KDD method's main objective is to extract data from big databases. This is done by using data mining. A deliberate, exploratory exploration and modeling of large data sources is what is referred to as KDD. In large and complex data sets, valid, useful, and intelligible patterns are found systematically with KDD. Data mining, which uses inference of algorithms to evaluate the data, create the model, and identify previously unnoticed patterns, is the foundation of the KDD approach. KDD take information out of the data, analyze it, and make predictions.Project flow can be seen from the figure 1.

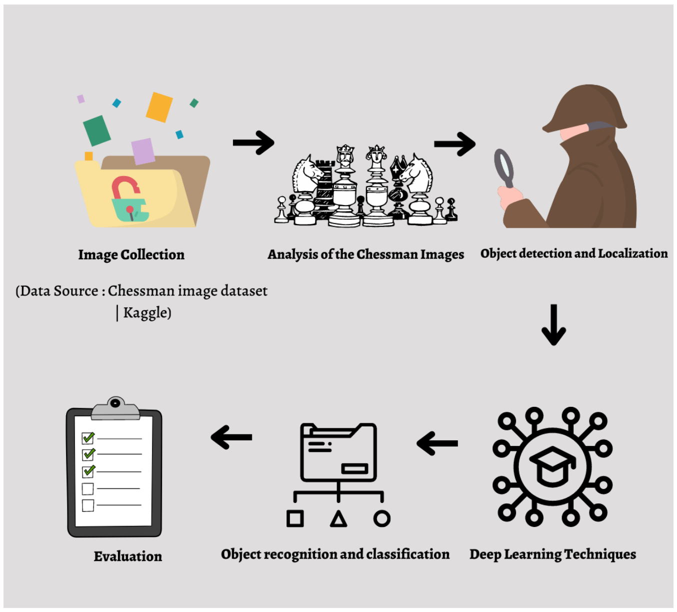


Image Collection
(Data Source : Chessman image dataset | Kaggle)
Analysis of the Chessman Images
Object detection and Localization
Deep Learning Techniques
Object recognition and classification
Evaluation

Fig. 1. Methology

A. Data Pre-Processing

This dataset Chessmen Image Classification is taken from Kaggle [1](https://www.kaggle.com/code/emericzhang/chessman) and it contains 552 images of chess pieces of 5 different types such as Queen,Pawn,Rook,Bishop, King and Knight encoded in JPG format.The photos were cropped to center producing images that ranged in size starting from 300 300 pixels.All Images are in different resolutions and colors.Red, green,and blue are the three hues that are used to represent every pixel in the image. Additionally, every pixel has a value range from 0 to 255 which needs to be normalized.In this study images are resized as 224 X 224 resolution and batch size used is 16 as it is a comparatively small dataset.Dataset is splitted into training dataset which contains 80% of the data and Validation dataset which contains 20% of the data. Samples of training and validation dataset is shown in figure 2 and 3 respectively.

B. deep learning techniques

In this report Keras is used with CNN (Convolutional Neural Network) to develop the model.Deep learning method known as Convolutional Neural Network(CNN) is primarily used for picture classification. The Keras package offers a simple way for creating CNN models. Typically, a CNN model has three primary types of layers:

Convolutional Layer : This layer applies various picture filters to the input dataset to low-level visual features, such

[1]https://www.kaggle.com/code/emericzhang/chessman

Fig. 2. Training Dataset Sample

Fig. 3. Validation Dataset Sample

as edges and curves. Three different filter sizes are available: 3x3, 5x5, and 7x7 , usually as odd numbers.The number of filters can be 16, 32, 64 and so on. The choice of the activation function is a crucial factor. It is possible to utilize a variety of activation functions, including Sigmoid, Tanh, Softmax, and ReLU. Since Relu is thought to be the best tool for feature extraction from photos, so it is used in the model as activation function.

Pooling Layer : In this layer lower - level features of pictures from the convolutional layer are used as input. Feature selection is performed via the pooling layer. Important image features are kept and carried over to subsequent levels, while less important features are suppressed at this layer. The pooling layer has two options maximum pool and mean pool. Max pool layers is selected for this study.

Dense Layer or Fully Connected Layer : The CNN model's final layers is dense layer. It also goes by the name "totally connected layer. CNN model should categorize the image as input into one of the 6 groups because there are six different sorts of chess pieces. This layer uses the data from the layer below to do classification and report the probabilities for each class. The class with the best chance will be chosen . In this layer Softmax is used in the model as activation function.

Other CNN architectures used in this report are also described in following sections:

1) Xception: The core principles of Inception are stretched to their absolute extent by Xception, which stands for "extreme inception." Contrary to Xception, 1x1 convolutions were used in Inception to compress the initial input. Rather, it applies the filtering to every depth map separately before compressing the input vector all at once with 1X1 convolution. This procedure is quite similar to a depth - wise separable convolution, which was initially used in 2014 to construct neural networks. Another difference amongst Inception and Xception is whether or not there is a non-linearity after the initial operation. While Xception doesn't introduce any non-linearity, the Inception model has a ReLU non-linearity that follows both processes.

2) VGG16: Convolutional neural network architecture known as VGG has been around for a while. It was derived from studies on increasing the density of these networks. In the network, there are tiny 3 x 3 filters. The network's simplicity, which includes pooling layers and a fully - connected layer as extra components, serves as its primary characteristic. In VGG16,16 denotes the fact that there are 16 layers with weights. Figure 4 shows the VGG16 architecture diagram.

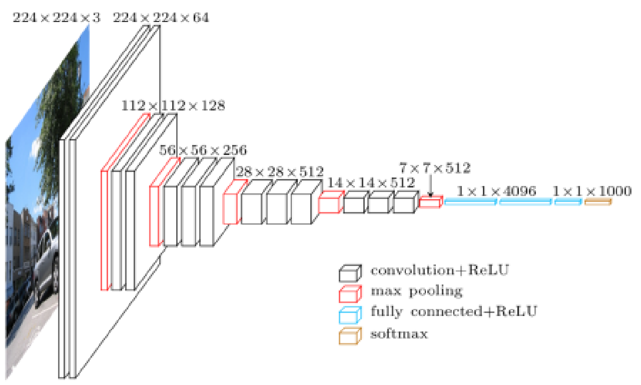


Fig. 4. VGG16 Architecture

3) Resnet50: ResNet, short for Residual Networks, is a generic neural network that forms the basis for a variety of visual identification applications. ResNet was a major accomplishment since it made it possible to train extraordinarily deep networks with more than 150 layers. Before ResNet, training deep learning models was difficult because of the problem of vanishing gradients.

4) MobileNet: The MobileNet model is TensorFlow's first mobile image recognition model and is designed for use in mobile applications, as its name suggests. MobileNet employs depth-wise separable convolutions. It significantly lowers the dimensionality when compared to a network with traditional convolutions of the same depth in the nets. The result of this model is compact deep neural networks. This provides us with a fantastic starting point for training our classifiers, which are quite compact and incredibly fast.MobileNet architecture can be seen from Figure 5.

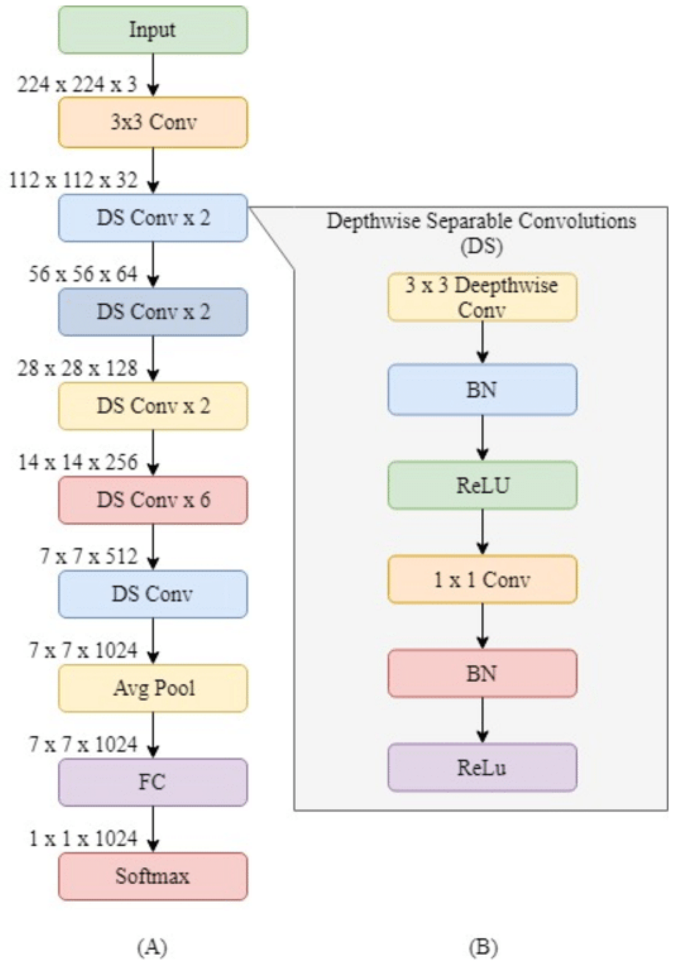
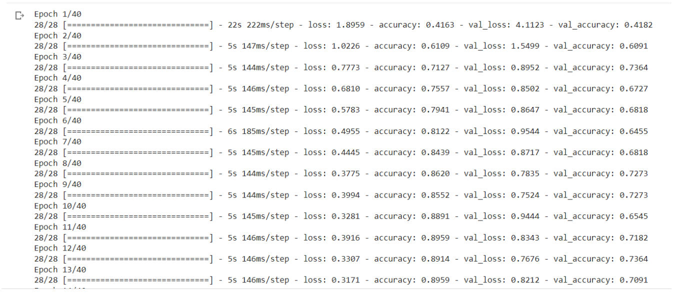


Fig. 5. MobileNet Architecture

IV. Evaluation and Results

The models chosen for the chessman classification problem are evaluated based on Accuracy and Loss. Validation accuracy and loss are considered for measuring the performance of the model. The measured values can be compared to choose the model with better Validation accuracy and loss.

1) Xception Model Result: The measures displayed in Fig 6, provides us with the accuracy and loss obtained using Xception model. categorical crossentropy loss function is used along with adam optimizer for determining the accuracy metrics. Forty epochs is being used to traverse the image both forward and backward. Model fitting is being done to calculate the accuracy for both the training and the validation dataset. As seen in Fig 6, the best validation accuracy was obtained as 0.7364 and the lowest value of loss was found to be 0.7135.



```
Epoch 1/40
28/28 [==============================] - 22s 222ms/step - loss: 1.8959 - accuracy: 0.4163 - val_loss: 4.1123 - val_accuracy: 0.4182
Epoch 2/40
28/28 [==============================] - 5s 147ms/step - loss: 1.0226 - accuracy: 0.6109 - val_loss: 1.5499 - val_accuracy: 0.6091
Epoch 3/40
28/28 [==============================] - 5s 144ms/step - loss: 0.7773 - accuracy: 0.7127 - val_loss: 0.8952 - val_accuracy: 0.7364
Epoch 4/40
28/28 [==============================] - 5s 146ms/step - loss: 0.6810 - accuracy: 0.7557 - val_loss: 0.8502 - val_accuracy: 0.6727
Epoch 5/40
28/28 [==============================] - 5s 145ms/step - loss: 0.5783 - accuracy: 0.7941 - val_loss: 0.8647 - val_accuracy: 0.6818
Epoch 6/40
28/28 [==============================] - 6s 185ms/step - loss: 0.4955 - accuracy: 0.8122 - val_loss: 0.9544 - val_accuracy: 0.6455
Epoch 7/40
28/28 [==============================] - 5s 145ms/step - loss: 0.4445 - accuracy: 0.8439 - val_loss: 0.8717 - val_accuracy: 0.6818
Epoch 8/40
28/28 [==============================] - 5s 144ms/step - loss: 0.3775 - accuracy: 0.8620 - val_loss: 0.7835 - val_accuracy: 0.7273
Epoch 9/40
28/28 [==============================] - 5s 144ms/step - loss: 0.3994 - accuracy: 0.8552 - val_loss: 0.7524 - val_accuracy: 0.7273
Epoch 10/40
28/28 [==============================] - 5s 145ms/step - loss: 0.3281 - accuracy: 0.8891 - val_loss: 0.9444 - val_accuracy: 0.6545
Epoch 11/40
28/28 [==============================] - 5s 146ms/step - loss: 0.3916 - accuracy: 0.8959 - val_loss: 0.8343 - val_accuracy: 0.7182
Epoch 12/40
28/28 [==============================] - 5s 146ms/step - loss: 0.3307 - accuracy: 0.8914 - val_loss: 0.7676 - val_accuracy: 0.7364
Epoch 13/40
28/28 [==============================] - 5s 146ms/step - loss: 0.3171 - accuracy: 0.8959 - val_loss: 0.8212 - val_accuracy: 0.7091
```

Fig. 6. Xception Accuracy and Loss

2) Accuracy Plot of Xception: The accuracy of both training and validation data is plotted for all the 40 epochs as seen in Fig 7. From the fig it is observed that the accuracy for the various epochs obtained was between 0.6 and 0.8 with the maximum accuracy as 0.7364.

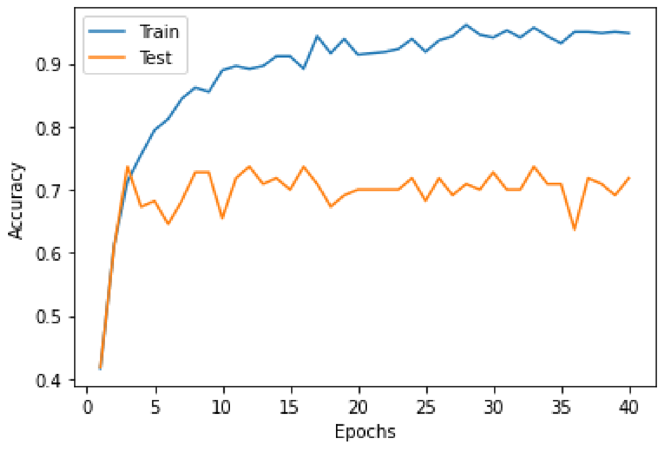


Fig. 7. Xception Accuracy Plot

3) Loss Plot of Xception: The loss value of both training and validation data is plotted for all the 40 epochs as seen in Fig 8.From the fig it is observed that the loss value for the various epochs obtained was between 0.7 and 1.0 with the minimum loss value as 0.7135.

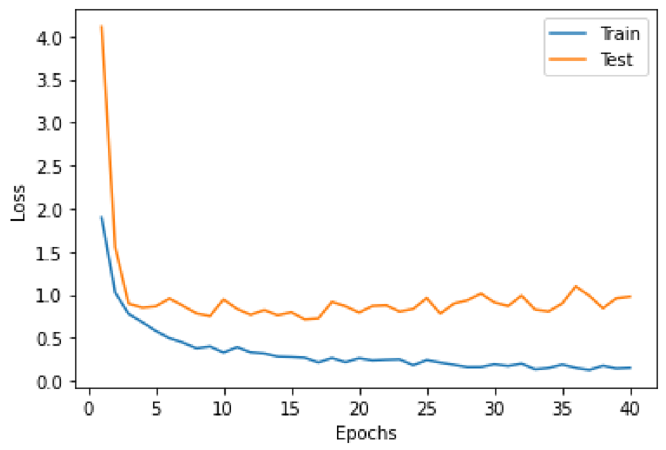


Fig. 8. Xception Loss Plot

4) Confusion Matrix of Xception: From the confusion matrix being obtained as seen in Fig 9, it is clear that higher proportion of correct prediction happened with classes Rook and Knight followed by king, pawn , bishop and queen.

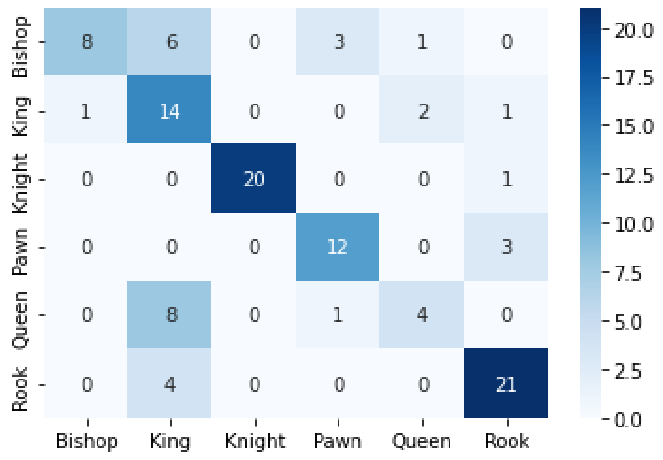


Fig. 9. Xception Confusion Matrix

5) VGG16 Model Result: The data in Fig. 10 shows us the accuracy and loss as determined by the VGG16 model. This model employs the activation functions of Relu and Softmax. Adam optimizer and categorical crossentropy loss function are utilized to calculate the accuracy metrics. The image is traversed both forward and backward across 40 epochs. The accuracy of the model is being calculated for both the training dataset and the validation dataset. As can be seen in Fig. 10, the lowest value of loss was determined to be 0.5912 and the best validation accuracy was obtained as 0.8455.

6) Accuracy Plot of VGG16: As demonstrated in Fig. 11, the accuracy of the training and validation data is plotted for each of the 40 epochs. The accuracy for the several epochs obtained ranged between 0.4 and 0.9, with the maximum accuracy being 0.8455

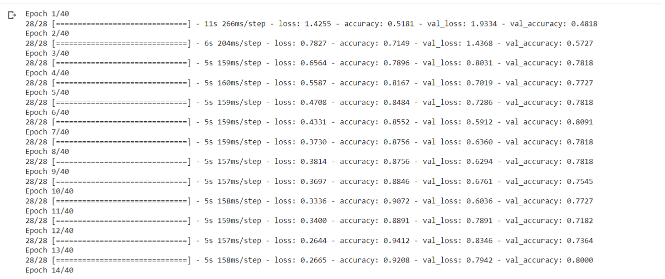

```
Epoch 1/40
28/28 [==============================] - 11s 266ms/step - loss: 1.4255 - accuracy: 0.5181 - val_loss: 1.9334 - val_accuracy: 0.4818
Epoch 2/40
28/28 [==============================] - 6s 204ms/step - loss: 0.7827 - accuracy: 0.7149 - val_loss: 1.4368 - val_accuracy: 0.5727
Epoch 3/40
28/28 [==============================] - 5s 159ms/step - loss: 0.6564 - accuracy: 0.7896 - val_loss: 0.8031 - val_accuracy: 0.7818
Epoch 4/40
28/28 [==============================] - 5s 160ms/step - loss: 0.5587 - accuracy: 0.8167 - val_loss: 0.7019 - val_accuracy: 0.7727
Epoch 5/40
28/28 [==============================] - 5s 159ms/step - loss: 0.4708 - accuracy: 0.8484 - val_loss: 0.7286 - val_accuracy: 0.7818
Epoch 6/40
28/28 [==============================] - 5s 159ms/step - loss: 0.4331 - accuracy: 0.8552 - val_loss: 0.5912 - val_accuracy: 0.8091
Epoch 7/40
28/28 [==============================] - 5s 159ms/step - loss: 0.3730 - accuracy: 0.8756 - val_loss: 0.6360 - val_accuracy: 0.7818
Epoch 8/40
28/28 [==============================] - 5s 157ms/step - loss: 0.3814 - accuracy: 0.8756 - val_loss: 0.6294 - val_accuracy: 0.7818
Epoch 9/40
28/28 [==============================] - 5s 157ms/step - loss: 0.3697 - accuracy: 0.8846 - val_loss: 0.6761 - val_accuracy: 0.7545
Epoch 10/40
28/28 [==============================] - 5s 158ms/step - loss: 0.3336 - accuracy: 0.9072 - val_loss: 0.6036 - val_accuracy: 0.7727
Epoch 11/40
28/28 [==============================] - 5s 159ms/step - loss: 0.3400 - accuracy: 0.8891 - val_loss: 0.7891 - val_accuracy: 0.7182
Epoch 12/40
28/28 [==============================] - 5s 157ms/step - loss: 0.2644 - accuracy: 0.9412 - val_loss: 0.8346 - val_accuracy: 0.7364
Epoch 13/40
28/28 [==============================] - 5s 158ms/step - loss: 0.2665 - accuracy: 0.9208 - val_loss: 0.7942 - val_accuracy: 0.8000
Epoch 14/40
```

Fig. 10. VGG16 Accuracy and Loss

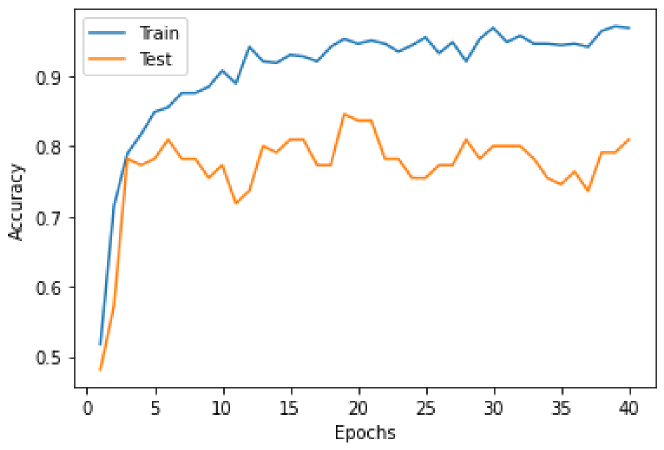


Fig. 11. VGG16 Accuracy Plot

7) Loss Plot of VGG16: As observed in Fig. 12, the loss value of the training and validation data is presented for each epoch from 1 to 40. The chart shows that the loss value for the different epochs was between 0.5 and 2.0, with 0.5912 being the smallest loss value.

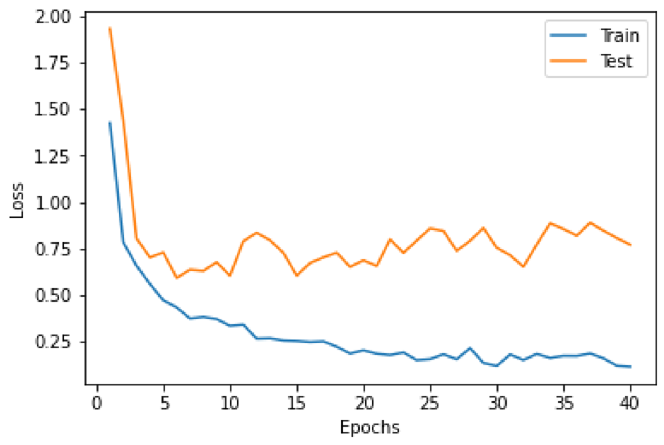


Fig. 12. VGG16 Loss Plot

8) Confusion Matrix of VGG16: The confusion matrix, which can be seen in Fig. 13, shows that classes Rook and Knight had the highest percentage of right predictions, followed by bishop, king, pawn, and queen.

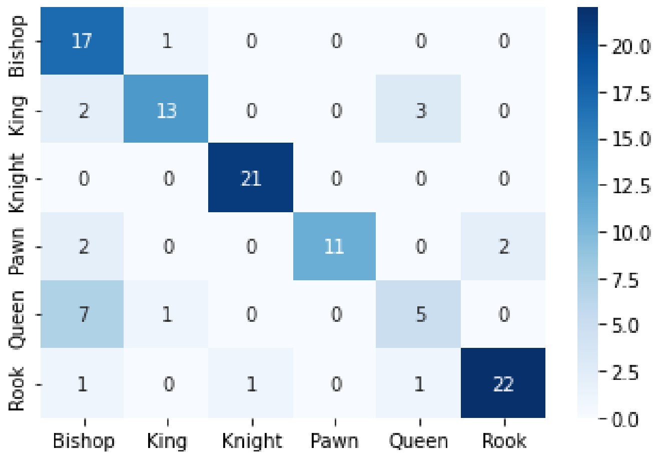
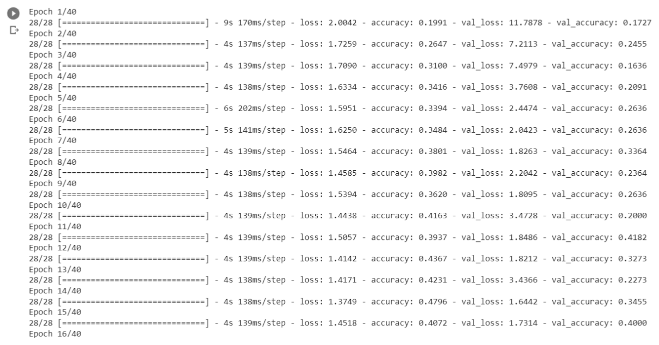


Fig. 13. VGG16 Confusion Matrix

9) Resnet50 Model Result: The accuracy and loss a calculated by the Resnet50 model using the data are shown in Fig. 14. Relu and Softmax's activation functions are used in this model. The accuracy metrics are computed using the categorical crossentropy loss function and the Adam optimizer. Over 40 epochs, the image is traveled both forward and backward. Both the training dataset and the validation dataset are used to determine the model's accuracy. As demonstrated in Fig. 14, the best validation accuracy was determined to be 0.4273 and the lowest amount of loss was determined to be 1.4923.



```
Epoch 1/40
28/28 [==============================] - 9s 170ms/step - loss: 2.0042 - accuracy: 0.1991 - val_loss: 11.7878 - val_accuracy: 0.1727
Epoch 2/40
28/28 [==============================] - 4s 137ms/step - loss: 1.7259 - accuracy: 0.2647 - val_loss: 7.2113 - val_accuracy: 0.2455
Epoch 3/40
28/28 [==============================] - 4s 139ms/step - loss: 1.7090 - accuracy: 0.3100 - val_loss: 7.4979 - val_accuracy: 0.1636
Epoch 4/40
28/28 [==============================] - 4s 138ms/step - loss: 1.6334 - accuracy: 0.3416 - val_loss: 3.7608 - val_accuracy: 0.2091
Epoch 5/40
28/28 [==============================] - 6s 202ms/step - loss: 1.5951 - accuracy: 0.3394 - val_loss: 2.4474 - val_accuracy: 0.2636
Epoch 6/40
28/28 [==============================] - 5s 141ms/step - loss: 1.6250 - accuracy: 0.3484 - val_loss: 2.0423 - val_accuracy: 0.2636
Epoch 7/40
28/28 [==============================] - 4s 139ms/step - loss: 1.5464 - accuracy: 0.3801 - val_loss: 1.8263 - val_accuracy: 0.3364
Epoch 8/40
28/28 [==============================] - 4s 138ms/step - loss: 1.4585 - accuracy: 0.3982 - val_loss: 2.2042 - val_accuracy: 0.2364
Epoch 9/40
28/28 [==============================] - 4s 138ms/step - loss: 1.5394 - accuracy: 0.3620 - val_loss: 1.8095 - val_accuracy: 0.2636
Epoch 10/40
28/28 [==============================] - 4s 139ms/step - loss: 1.4438 - accuracy: 0.4163 - val_loss: 3.4728 - val_accuracy: 0.2000
Epoch 11/40
28/28 [==============================] - 4s 139ms/step - loss: 1.5057 - accuracy: 0.3937 - val_loss: 1.8486 - val_accuracy: 0.4182
Epoch 12/40
28/28 [==============================] - 4s 139ms/step - loss: 1.4142 - accuracy: 0.4367 - val_loss: 1.8212 - val_accuracy: 0.3273
Epoch 13/40
28/28 [==============================] - 4s 138ms/step - loss: 1.4171 - accuracy: 0.4231 - val_loss: 3.4366 - val_accuracy: 0.2273
Epoch 14/40
28/28 [==============================] - 4s 138ms/step - loss: 1.3749 - accuracy: 0.4796 - val_loss: 1.6442 - val_accuracy: 0.3455
Epoch 15/40
28/28 [==============================] - 4s 139ms/step - loss: 1.4518 - accuracy: 0.4072 - val_loss: 1.7314 - val_accuracy: 0.4000
Epoch 16/40
```

Fig. 14. Resnet50 Accuracy and Loss

10) Accuracy Plot of Resnet50: As demonstrated in Fig. 15, the accuracy of the training and validation data is plotted for each of the 40 epochs. The accuracy for the several epochs obtained ranged between 0.1 and 0.5, with the maximum accuracy being 0.4273

11) Loss Plot of Resnet50: The loss value of the training and validation data is shown for each epoch from 1 to 40, as can be seen in Fig. 16. According to the graph, the loss value for each epoch ranged from 1.0 to 12.0, with 1.4923 being the smallest loss value.

12) Confusion Matrix of Resnet50: Classes Rook and Knight had the highest percentage of accurate predictions,

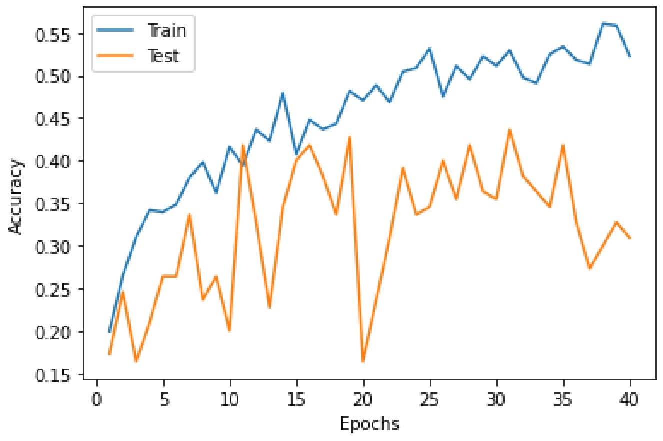


Fig. 15. Resnet50 Accuracy Plot

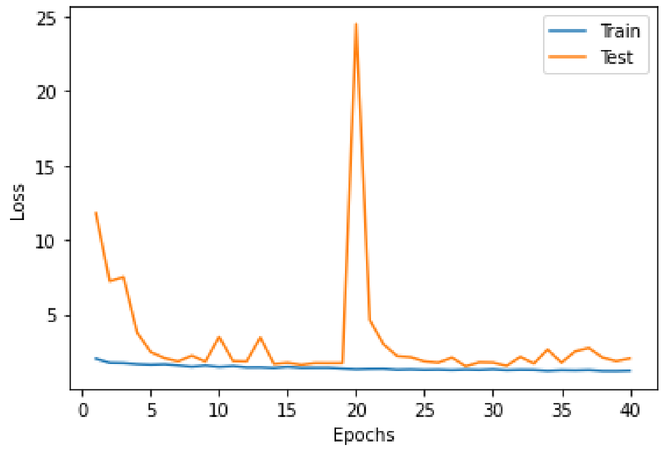


Fig. 16. Resent50 Loss Plot

followed by bishop, king, pawn, and queen, according to the confusion matrix, which is depicted in Fig. 17.

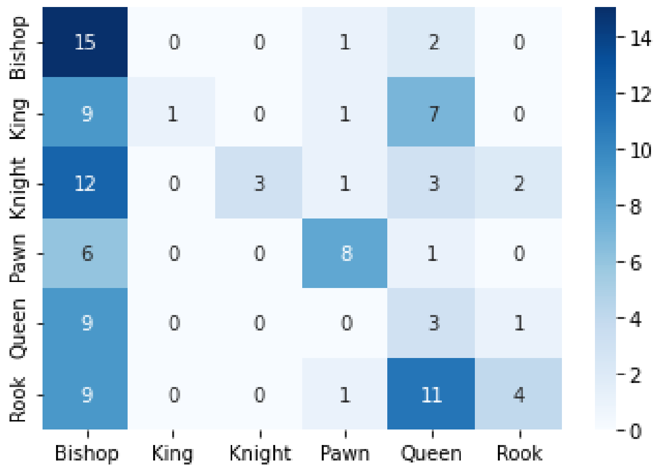
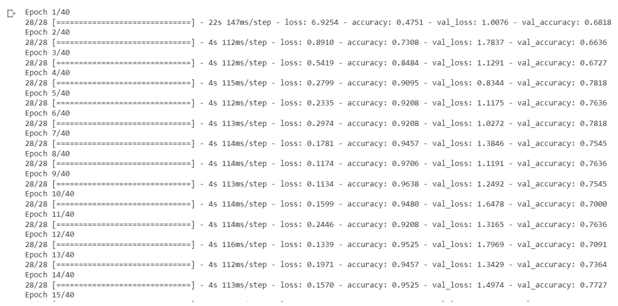


Fig. 17. Resnet50 Confusion Matrix

A. MobileNet Model Result

Using the data, the MobileNet model determined the accuracy and loss, which are displayed in Fig. 18. The activation functions of Relu and Softmax are employed in this model. The categorical crossentropy loss function and the Adam optimizer are used to calculate the accuracy metrics. The image travels both forward and backward during a period of 40 epochs. To assess the model's precision, both the training dataset and the validation dataset are employed. The best validation accuracy was found to be 0.8273, and the least amount of loss was found to be 0.8344, as shown in Fig. 18.



```
Epoch 1/40
28/28 [==============================] - 22s 147ms/step - loss: 6.9254 - accuracy: 0.4751 - val_loss: 1.0076 - val_accuracy: 0.6818
Epoch 2/40
28/28 [==============================] - 4s 112ms/step - loss: 0.8910 - accuracy: 0.7308 - val_loss: 1.7837 - val_accuracy: 0.6636
Epoch 3/40
28/28 [==============================] - 4s 112ms/step - loss: 0.5419 - accuracy: 0.8484 - val_loss: 1.1291 - val_accuracy: 0.6727
Epoch 4/40
28/28 [==============================] - 4s 115ms/step - loss: 0.2799 - accuracy: 0.9095 - val_loss: 0.8344 - val_accuracy: 0.7818
Epoch 5/40
28/28 [==============================] - 4s 112ms/step - loss: 0.2335 - accuracy: 0.9208 - val_loss: 1.1175 - val_accuracy: 0.7636
Epoch 6/40
28/28 [==============================] - 4s 113ms/step - loss: 0.2974 - accuracy: 0.9208 - val_loss: 1.0272 - val_accuracy: 0.7818
Epoch 7/40
28/28 [==============================] - 4s 114ms/step - loss: 0.1781 - accuracy: 0.9457 - val_loss: 1.3846 - val_accuracy: 0.7545
Epoch 8/40
28/28 [==============================] - 4s 114ms/step - loss: 0.1174 - accuracy: 0.9706 - val_loss: 1.1191 - val_accuracy: 0.7636
Epoch 9/40
28/28 [==============================] - 4s 113ms/step - loss: 0.1134 - accuracy: 0.9638 - val_loss: 1.2492 - val_accuracy: 0.7545
Epoch 10/40
28/28 [==============================] - 4s 114ms/step - loss: 0.1599 - accuracy: 0.9480 - val_loss: 1.6478 - val_accuracy: 0.7000
Epoch 11/40
28/28 [==============================] - 4s 114ms/step - loss: 0.2446 - accuracy: 0.9208 - val_loss: 1.3165 - val_accuracy: 0.7636
Epoch 12/40
28/28 [==============================] - 4s 116ms/step - loss: 0.1339 - accuracy: 0.9525 - val_loss: 1.7969 - val_accuracy: 0.7091
Epoch 13/40
28/28 [==============================] - 4s 112ms/step - loss: 0.1971 - accuracy: 0.9457 - val_loss: 1.3429 - val_accuracy: 0.7364
Epoch 14/40
28/28 [==============================] - 4s 113ms/step - loss: 0.1570 - accuracy: 0.9525 - val_loss: 1.4974 - val_accuracy: 0.7727
Epoch 15/40
```

Fig. 18. MobileNet Accuracy and Loss

1) Accuracy Plot of MobileNet: The accuracy of the training and validation data is plotted for each of the 40 epochs, as seen in Fig. 19. The accuracy for the different epochs was 0.6 to 0.9, with 0.8273 being the highest accuracy.

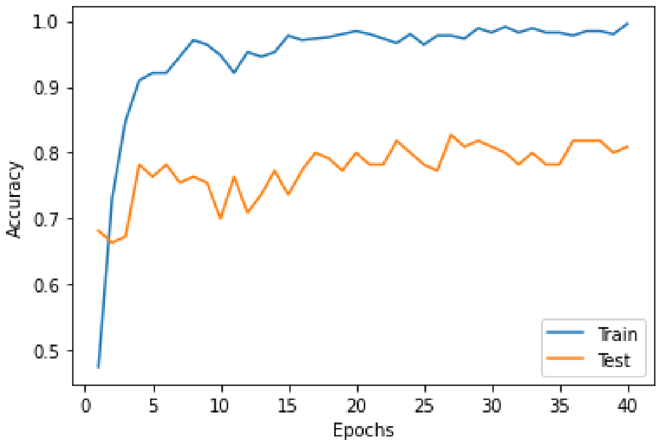


Fig. 19. MobileNet Accuracy Plot

2) Loss Plot of MobileNet: As observed in Fig. 20, the loss value of the training and validation data is displayed for each epoch from 1 to 40. The graph shows that each epoch's loss value varied from 0.0 to 2.0, with 0.8344 representing the smallest loss value.

3) Confusion Matrix of MobileNet: Classes Rook and Knight had the highest percentage of accurate predictions, followed by bishop, king, pawn, and queen, according to the confusion matrix, which is depicted in Fig. 21.

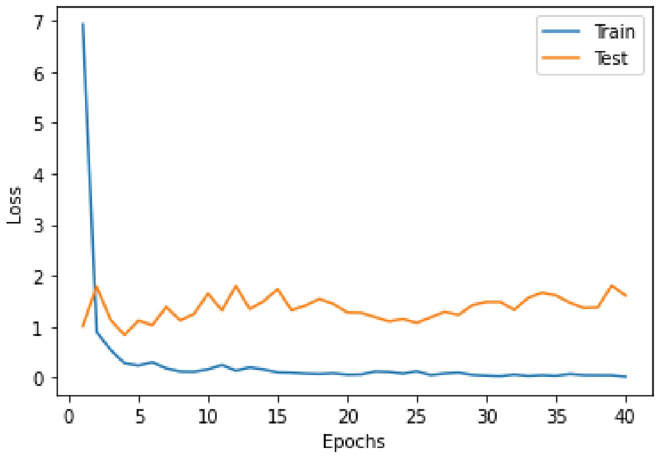


Fig. 20. MobileNet Loss Plot

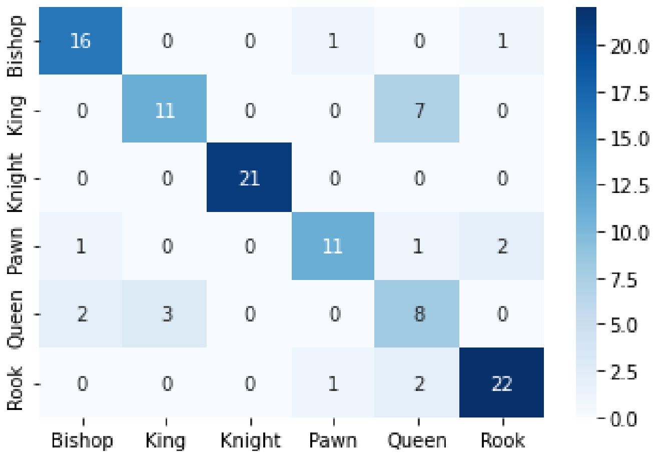


Fig. 21. Mobile Net Confusion Matrix

V. CONCLUSION AND FUTURE WORK

CNN architectures such as Xception, VGG16, MobileNet and Resnet50 are being used in the classification of images to identify the model with good classification accuracy on the validation data. From the results, it is clearly evident that VGG16 has the highest accuracy value of 0.8455 followed by MobileNet in the 2nd position with a classification accuracy of 0.8273. This research can be further extended by classifying the chess pieces by color. Thus it can help to determine whether a particular chess piece belongs to the player or the opponent. Hence it is successfully proved that we can do multi class classification of the chessmen images into appropriate chess piece groups such as rook, king, queen, Bishop, Knight and Pawn . Also, it is proved that we can chose the best neural network method based on accuracy to classify the images. In our case VGG16 is chosen since it has the highest accuracy generated when compared with other models

References

[1] D. Urting and Y. Berbers, “Marineblue: A low-cost chess robot.” in *Robotics and Applications*, 2003, pp. 76–81.

[2] E. Sokic and M. Ahic-Djokic, “Simple computer vision system for chess playing robot manipulator as a project-based learning example,” in *2008 IEEE International Symposium on Signal Processing and Information Technology*. IEEE, 2008, pp. 75–79.

[3] J. Ding, “Chessvision: Chess board and piece recognition,” Technical Report. Stanford University. URL: https://web. stanford. edu/class . . . , Tech. Rep., 2016.

[4] C. Danner and M. Kafafy, “Visual chess recognition,” *URL: http://web. stanford. edu/class/ee368/Project_Spring_1415/Reports/Danner_Kafafy. pdf*, 2015.

[5] Y. Xie, G. Tang, and W. Hoff, “Chess piece recognition using oriented chamfer matching with a comparison to cnn,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 2001–2009.

[6] N. Banerjee, D. Saha, A. Singh, and G. Sanyal, “A simple autonomous robotic manipulator for playing chess against any opponent in real time,” in *Proceedings of the International Conference on Computational Vision and Robotics*, 2011.

[7] V. Wang and R. Green, “Chess move tracking using overhead rgb webcam,” in *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*. IEEE, 2013, pp. 299–304.

[8] M. A. Czyzewski, A. Laskowski, and S. Wasik, “Chessboard and chess piece recognition with the support of neural networks,” *arXiv preprint arXiv:1708.03898*, 2017.

[9] A. Mehta, “Augmented reality chess analyzer (archessanalyzer): In-device inference of physical chess game positions through board segmentation and piece recognition using convolutional neural network,” *arXiv preprint arXiv:2009.01649*, 2020.

[10] J. Gonçalves, J. Lima, and P. Leitao, “Chess robot system: A multi-disciplinary experience in automation,” in *9th Spanish Portuguese Congress On Electrical Engineering*, 2005.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.

[12] J. Hack and P. Ramakrishnan, “Cvchess: Computer vision chess analytics,” 2014.

[13] O. Arandjelović, “Determining chess game state from an image,” *Journal of Imaging*, vol. 7, no. 6, p. 94, 2021.

[14] K. Y. Tam, J. A. Lay, and D. Levy, “Automatic grid segmentation of populated chessboard taken at a lower angle view,” in *2008 Digital Image Computing: Techniques and Applications*. IEEE, 2008, pp. 294–299.

[15] G. Wölflein and O. Arandjelović, “Determining chess game state from an image,” *Journal of Imaging*, vol. 7, no. 6, p. 94, 2021.

[16] J. Hou, “Chessman position recognition using artificial neural networks.”