

CHES CONFIGURATION TECTION

Machine Learning Project
Semester - VI



Machine Learning Project Semester - VI

by

Student Name	Student Number
Madhumithaa S	185001087
Mahesh Bharadwaj K	185001089
Purushothaman M	185001308

Contents

1	Introduction	1
1.1	Forsyth-Edwards Notation (FEN)	1
2	The Dataset	2
2.1	Data Processing	3
2.2	Class Imbalance in Dataset	4
2.3	Undersampling Dataset	4
3	Model Architecture	5
3.1	Model Training	6
3.1.1	Classifier Performance Evaluation Metrics	6
3.2	Model Testing	6
3.2.1	Evaluating Predicted FEN	7
4	Conclusion	8

1

Introduction

This project presents a learning based model for identifying the configuration of a given chess board through image classification. Forsyth-Edwards Notation (FEN) is the standard notation to describe positions of a chess game. FEN makes it easy to translate any chess position into a single line of text. It allows players to restart games from any point they desire. FEN notation is accepted by most chess engines for performing analysis work on the games. We have used a Deep Learning model to identify the chess pieces. A three layer Convolutional Neural Network (CNN) model is used to identify the FEN configuration of a chessboard.

1.1. Forsyth-Edwards Notation (FEN)

The first field represents the placement of pieces. It starts describing the content of each square, beginning from the eighth rank and ending with the first. For each rank, squares begin from the first file and go to the eighth.

Lowercase letters describe the black pieces. Just like in PGN, "p" stands for pawn, "r" for rook, "n" for knight, "b" for bishop, "q" for queen, and "k" for king. The same letters are used for the white pieces, but they appear in uppercase. Empty squares are denoted by numbers from one to eight, depending on how many empty squares are between two pieces.



Figure 1.1: Sample FEN for a position

2

The Dataset

The given dataset contains images of chess board with pieces of both black and white. Each image is of size *254x254 pixels*. We have about **40,000 training images and 4,000 testing images**. There is also a CSV file which consists of the FEN of each image in the dataset. The dataset used can be found here: [AI Crowd Challenge](#) .

A sample image from the dataset and its FEN is given below:

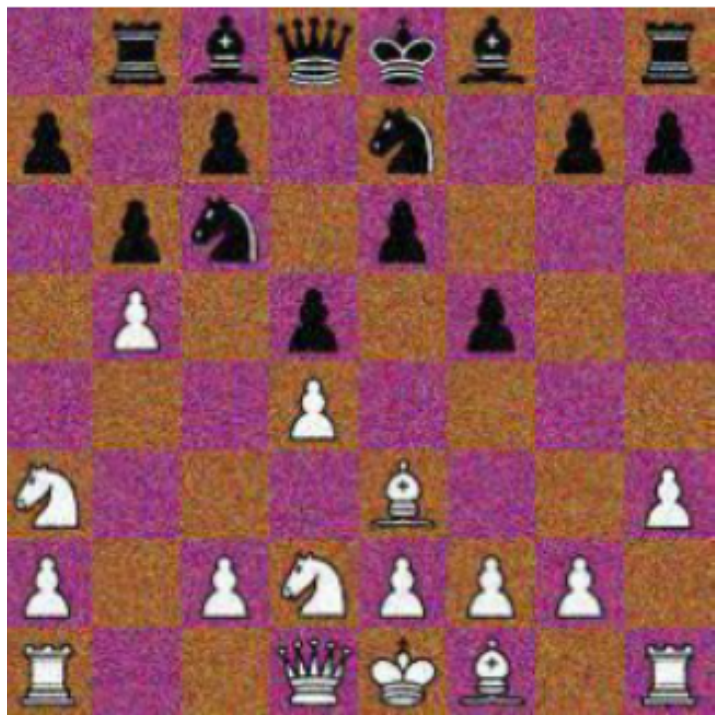


Figure 2.1: FEN: 1rbqkb1r/p1p1n1pp/1pn1p3/1P1p1p2/3P4/N3B2P/P1PNPPP1/R2QKB1R

2.1. Data Processing

- Each image is read in gray scale.
- We take the image and resize it to 256 x 256.
- Gaussian Blur is applied with 3x3 kernel size to reduce the noise in image
- This image is divided by 8 row wise and column wise to get 64 cells of a chess board, each of size 32 x 32.
- Each 32 x 32 image represents a square in the board which may or may not have a piece.
- Each cell is used as training data with the piece on that cell as the target.
- We convert the pieces into numbers for the model to understand as shown in table 2.1 below.








Piece	Class	Piece	Class
Empty Cell	0		7
	1		8
	2		9
	3		10
	4		11
	5		12
	6		

Table 2.1: Pieces and their numeric mapping

2.2. Class Imbalance in Dataset

Every chessboard will have a minimum of 32 empty squares at any given state, so there is an imbalance in the classes with empty cell having huge majority followed by black and white pawns. This will result in overfitting, i.e. the model will become overfitted to these classes. Figure 2.3 below shows the distribution of classes in the dataset:

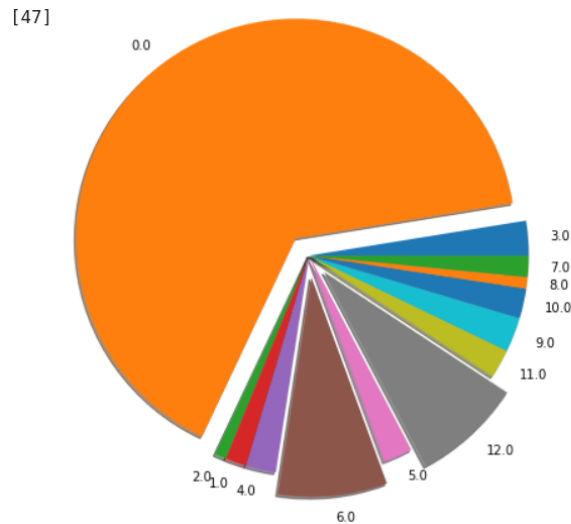


Figure 2.2: Distribution of Classes in dataset

2.3. Undersampling Dataset

To resolve this imbalance we are introducing the process of re-sampling. Re-sampling involves creating a new transformed version of the training data set in which the selected examples have a different class distribution. There are two approaches to re-sample data, oversampling and under sampling and here, we are going to use the under sampling approach since we have enough data even in minority class for training. The **RandomUnderSampler (RUS)** deletes records randomly from the majority class labels to match the minority class label so the imbalance in the data is removed. Figure ?? below shows the distribution of classes after under sampling in the data set:

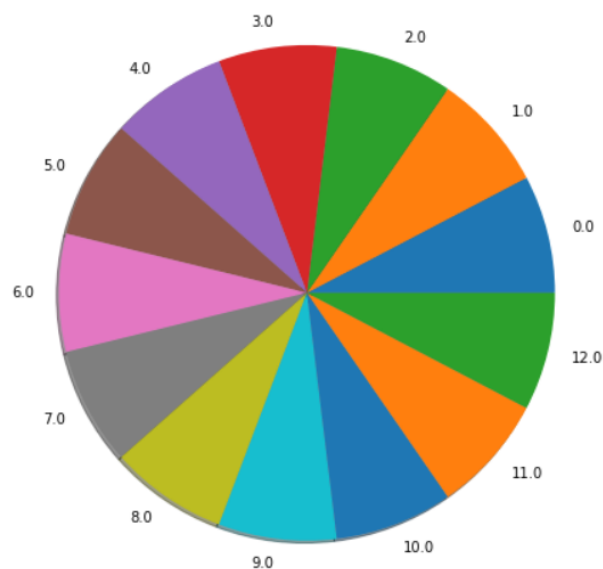


Figure 2.3: Distribution of Classes in data set after RUS

3

Model Architecture

A three layer Convolutional Neural Network is used as the model to predict the piece present in the cell and the visual representation of the same is given below in figure 3.1.

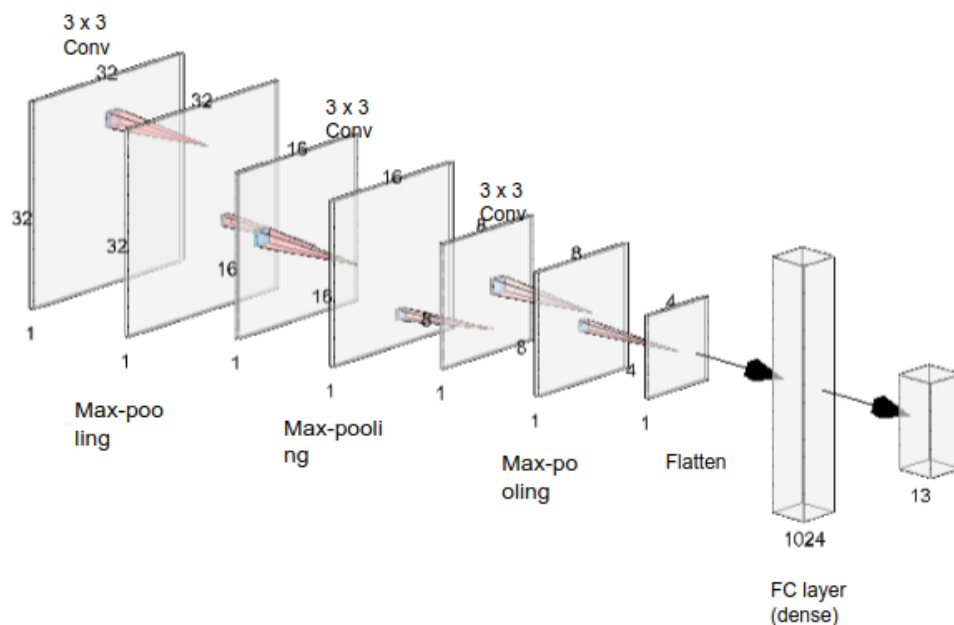


Figure 3.1: Our Proposed Architecture

The Convolutional layer is used to extract features from the image. The Convolution at the lower layers capture basic patterns like edges and curves and as we go deeper, patterns are combined to form higher level polygons (Ex: multiple lines combined to detect squares) and so on. The max-pooling layer reduces the size of the image without losing much information. A 32 * 32 image (one cell of the chess board) is fed into the model as input. The first convolutional layer extracts high level features from this image using a 3x3 kernel. The result is then fed into the max-pooling layer with 2x2 kernel, which reduces the size of the image to 16 * 16. The 16 * 16 image is then fed into another convolutional layer (3x3 kernel) followed by a max-pool (2x2) layer, where the size of the image is reduced to 8 * 8. This process is repeated again to obtain a 4x4 result. This result is then flattened into a one dimensional array and fed into the fully connected layer with *softmax activation function*. This layer produces the probabilities that the image belongs to each of the 13 classes. The class with the highest probability is chosen as the class for that image.

3.1. Model Training

The model is trained using **Adam Optimizer** with a **batch size of 1024** for **50 epochs** using GPU back-end in google colaboratory to train faster.

3.1.1. Classifier Performance Evaluation Metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results obtained on the test set after training are given below in figure 3.2.

Classification report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	167461
1	0.95	1.00	0.98	4000
2	0.98	1.00	0.99	2163
3	0.99	1.00	1.00	6530
4	1.00	1.00	1.00	5584
5	0.99	1.00	1.00	5371
6	1.00	1.00	1.00	20501
7	0.99	1.00	0.99	4000
8	0.96	1.00	0.98	2160
9	0.99	1.00	1.00	6614
10	1.00	1.00	1.00	5601
11	1.00	1.00	1.00	5429
12	0.99	1.00	0.99	20586
accuracy			1.00	256000
macro avg	0.99	1.00	0.99	256000
weighted avg	1.00	1.00	1.00	256000

Figure 3.2: Results Obtained

3.2. Model Testing

Steps to get FEN of unseen image

1. Read the image in grayscale and resize to 256x256.
2. Apply Gaussian Blur with (3x3) kernel to remove noise in images.
3. Divide the image into 8 rows and 8 columns and get 64 images of size 32x32.
4. Predict the class of these cells using the model trained above
5. Create an empty chess.Board object
6. For each cell, if it not 'empty cell', set the piece in the cell as the piece predicted by the model.
7. Display the FEN of the board using board.fen()

Given below in figure 3.3 of actual FEN vs Predicted FEN using the model:

ImageID		Label		PredictedFEN
0	0	7r/2k5/8/8/Pp2P3/RP5r/2R5/K2R1b2		7r/2k5/8/8/Pp2P3/RP5r/2R5/K2R1b2
1	1	r1bq1b2/2pppkpr/p6n/5p1p/P1pn1P2/4BNPP/R2KP3/1...		r1bq1b2/2pppkpr/p6n/5p1p/P1pn1P2/4BNPP/R2KP3/1...
2	2	r1b2br1/p1ppq2p/np1k1pp1/4p1B1/3P2PP/P4P2/1PP1...		r1b2br1/p1ppq2p/np1k1pp1/4p1B1/3P2PP/P4P2/1PP1...
3	3	r2r4/pp4bp/4BN1n/1P1pkb2/1n1p2Pp/P1N1Pp2/1BP5/...		r2r4/pp4bp/4BN1n/1P1pkb2/1n1p2Pp/P1N1Pp2/1BP5/...
4	4	2r3nr/1bppk3/5q1p/p1P1pPp1/1n3P2/1pPK3P/1P2PR2...		2r3nr/1bppk3/5q1p/p1P1pPp1/1n3P2/1pPK3P/1P2PR2...
...
3995	3995	8/1r3pBr/2Pk4/2p4p/5P1R/1P4P1/P4K2/5B2		8/1r3pBr/2Pk4/2p4p/5P1R/1P4P1/P4K2/5B2
3996	3996	4rkn1/ppp5/3q4/PN1pppPP/2PP1b2/N1B2r1P/2B3K1/Q...		3rrknp/ppp5/3q4/PN1pppPP/2PP1b2/N1B2r1P/2B3K1/...
3997	3997	nk6/5p1r/6Pp/1P1pNK1P/1P3bB1/4p3/3R4/7q		nk6/5p1r/6Pp/1P1pNK1P/1P3bB1/4p3/3R4/7q
3998	3998	rn3bn1/pp4p1/Pqp1bk2/3pp2r/2B2p2/4P2P/RPPP1PP1...		rn3bn1/pp4p1/Pqp1bk2/3pp2r/2B2p2/4P2P/RPPP1PP1...
3999	3999	rnbk2nr/1p4pp/p3pp2/2pp3P/P1P1PNPR/R5b1/1P1P1q...		rnbk2nr/1p4pp/p3pp2/2pp3P/P1P1PNPR/R5b1/1P1P1q...

4000 rows x 3 columns

Figure 3.3: Dataframe showing results obtained using model

3.2.1. Evaluating Predicted FEN

We have used Word Error Rate to evaluate the performance of the predicted FEN to the actual ground truth.

$$WER = \frac{S + I + D}{N}$$

- **Substitutions (s)** are anytime a word gets replaced (for example, “twinkle” is transcribed as “crinkle”)
- **Insertions (I)** are anytime a word gets added that wasn’t said (for example, “trailblazers” becomes “tray all blazers”)
- **Deletions (D)** are anytime a word is omitted from the transcript (for example, “get it done” becomes “get done”)
- **N** is the total words in the string/sentence.

Our model obtained a **WER of 0.014** when tested on the 4000 testing images provided in the dataset.

4

Conclusion

The dataset of 40,000 images was used to create a training corpus consisting of cells in the board and the piece present on the cell. The imbalance in the dataset was rectified using **Random Under Sampler**. A convolutional neural network having 3 layers was trained on this corpus for 50 epochs using **Adam optimizer** and achieved **99% accuracy** on the training data. The trained model was used to predict the FEN of the images in the test set and the Word Error Rate of the predictions was calculated and reported as 0.014.