Rock, Paper, and Scissor game using Convolutional Neural Networks

Abhinav Umat

Department of Computer Science, Wright State University

Email:- umat.2@wright.edu

Abstract

Convolutional Neural Networks (CNNs) is a great deep learning technique which we can use for analyzing visual imagery. CNNs are a regularized version of multilayer perceptron by that I mean they take advantage of the hidden patterns in a given data as opposed to a multilayer perceptron. We can use CNNs for various kinds of Image classification and Image recognition tasks. In this project I have built and trained a CNN model to classify between three different types of images, Rock, Paper and Scissor to create a Rock, Paper, Scissor game.

Keywords: Convolutional Neural Networks (CNNs), Rock, Paper, Scissor, Image Classification.

1. Introduction

Convolutional Neural Networks is a really powerful tool which can be used for various Image analysis related tasks. One of the areas where CNNs along with other machine learning techniques are extensively used is video games. CNNs ability to learn patterns which are not dependent on location has made it a great tool to be used in video games. Some of the games where we use these techniques are, Chess, Go, AlphaGo etc.

For this project I have built and trained a CNN model to play a Rock, Paper, Scissor Game. I have trained the model on a training set which includes images from all these three classes and tested it on the test set. I then used a Validation dataset with images which were not included in the training and the testing set. I also took some pictures of the hand gestures for all these three classes (Rock, paper and Scissor) from my own mobile camera and used those images as well to play the game to see how well the model performed on them.

The goal of this project is to see how well does a CNN perform to classify images and how we can use this technique to play these games and not only on images which it was trained and tested on but also on the images which were not included in the training or testing process. Images which the model has not seen before.

2. Related Research

The motivation for this project is to figure out how we can use CNNs in video games. There are plenty of research which is going on in the area of video games using CNNs. Some of those researches, I have listed below:

- **Training Deep Convolutional Neural network to play Go by Christopher Clark, and Amos Storkey.**

  In this paper they have used pattern recognition to train a deep convolutional neural network to play Go by training them to predict the moves made by expert Go players. They were able to achieve move prediction accuracy of 41.1% and 44.4% on two different Go datasets. Previous Go algorithms use thousands of simulations to make predictions of the next move, but human Go experts use pattern recognition to make the best next move. The authors

have used the same technique of pattern recognition using CNNs to close the gap between humans and computers in playing this game [1].

- **Predicting Resource Locations in Game Maps Using Deep Convolutional Neural Networks by Scott Lee, Aaron Isaksen, Christoffer Holmgard, and Julian Togelius.**

  In this paper they have used convolutional neural networks to predict the placements of the resources in StarCraft II maps. They trained the model on the existing maps taken from the databases of maps used in online competitions and tested on the unseen maps with resources removed to see how well the model predicts where a particular resource is placed. They believe these predictions can be used as suggestions, where the designer can be presented with suggestions for placing various map features based on incomplete designs at all stages of the map creation process [2].

- **Predicting Moves in Chess using Convolutional Neural Networks by Barak Oshri, and Nishith Khandwala.**

  In this paper they have designed and trained a convolutional neural network to make predictions which chess pieces to move and where to move them. They designed two models, one CNN was trained for selecting a piece and one CNN was trained to produce scores for where it should be moved. They trained the model using 20,000 games with 245,000 moves made by player with ELO rating higher than 2000 [3].

3. Exploratory data analysis

For this project, I have about 2188 images corresponding to the 'Rock' (726 images), 'Paper' (710 images) and 'Scissors' (752 images) of hand gestures of the Rock-Paper-Scissors game. All images are in RGB images of 300 pixels wide by 200 pixels height in .PNG format [4].

Figure 1 show the number of images I have picked for the training dataset. I have about 1620 training images and for all the three classes Rock, Paper and Scissors have 540 images each. I wanted no class imbalance for the training set so I picked the same number of images for all the three hand gestures.

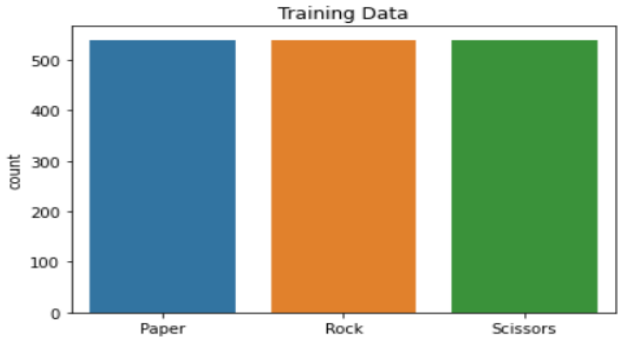

Fig. 1. Training Dataset

Figure 2 show the number of images I have picked for the testing dataset. I have about 562 test images corresponding to the ‘Rock’ (188 images), ‘Paper’ (186 images) and ‘Scissors’ (188 images). Test set is also pretty fairly balanced

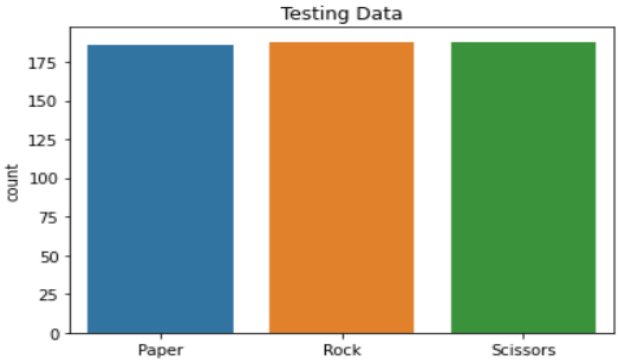

Fig. 2. Testing Dataset

There are some images which are in both training and testing dataset but most of the images are different. Regardless, I have picked 10 images from each of the three classes of the Rock, paper and Scissors for the validation dataset. It has a total of 30 images. These images are not included in either of the training or testing dataset. I have used these images to play the game.

I also have 30 images, 10 for each of the three hand gestures, which I have taken from my own mobile camera. These images are not part of training, testing or validation dataset. I have also used these images to play the game to see how well does the model perform on completely unseen images.

4. Preparing the Data

Before building and training the model I had to prepare the training and testing dataset. I created four different folders, one folder for the training dataset for all the three hand gestures, one folder for the testing dataset for all the three hand gestures, one folder for the validation dataset for all the three hand gestures with which I will play the game, and one folder for the images that I took with my own mobile camera for all the three hand gestures with which I will also play the game.

Preprocessing the data

I first created a training and testing dataset by importing the respective images. I converted those images into arrays, for normalization I divided all the images by 255.0 and then I labeled the images on the training and testing dataset respectively with their labels as, 0 for Paper, 1 for Rock and 2 for Scissors. I have also used one hot encoding for my labels for the training and testing dataset.

Once I finished preparing my data, I went on to build and train my model.

5. Building the Model

Figure 3 show the final architecture for my Model. For my model I have used 2 Convolutional layers with 32 filters and 64 filters respectively and (7,7) kernel size for both convolutional layers. I have not used any padding. I used 2 Maxpooling layers with a pool size of (2,2) after both convolutional layers. Then I flattened my parameters and added 2 hidden layers, with 100, 50 hidden nodes respectively. And 1 Output layer with 3 nodes, since there are 3 classes. For activation I have used relu for the convolutional layer and the hidden layers and softmax for the output layer. I have used SGD (Stochastic gradient descent) optimizer. I used Categorical Cross entropy for classifying multiple classes. I have set my learning rate to be 0.01. I have trained my model for 9 epochs. This model gave me test accuracy of 93.2%.

Experiments

I played around with the structure like first I used 1 convolutional layer with 1 maxpooling layer, test accuracy with this model was in mid-80's. I think the results with 1 convolutional layer were not that good because these images are really large with (300,200) size and also they are in color so 2 convolutional layers are required at least to achieve accuracy more than 90%. I also checked 3 convolutional layers with 3 maxpooling layers but the results were pretty much the same as with my final model but time taken to train was almost double. So, I picked the model with 2 convolutional layers as my final model.

I also played around with learning rates, I decayed the learning rate also, I also used Adam optimizer, but the results were really bad or good but not as good as this model.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 294, 194, 32)	4736
max_pooling2d (MaxPooling2D)	(None, 147, 97, 32)	0
conv2d_1 (Conv2D)	(None, 141, 91, 64)	100416
max_pooling2d_1 (MaxPooling2D)	(None, 70, 45, 64)	0
flatten (Flatten)	(None, 201600)	0
dense (Dense)	(None, 100)	20160100
dense_1 (Dense)	(None, 50)	5050
dense_2 (Dense)	(None, 3)	153

Total params: 20,270,455
Trainable params: 20,270,455
Non-trainable params: 0

Fig. 3. Final architecture of the Model

6. Results

Figure 4 and 5 show the training and testing accuracy and training and test loss respectively for my Model. As you can see my model does really well on both the datasets. Expect for the initial dip in the test accuracy it starts to increase over epochs just as the training accuracy. Training and testing loss also keep decreasing over epochs. I only trained this model for 9 epochs that's because after 9 epochs training gets really noisy, even though training accuracy keeps on increasing, the test accuracy starts to decrease, the best it could achieve was in mid-90's. I think it does that because all the images look quite similar. Even though there are slight differences in angles and hand placements but for the most part images look

pretty similar and after 9 epochs it starts to overtrain and model starts to overfit. So, I think for this dataset 9 epochs are enough to achieve the best results.

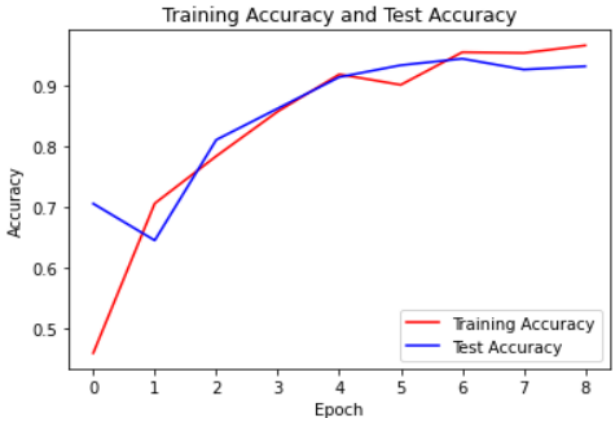

Fig. 4. Training and Test accuracy of the Model

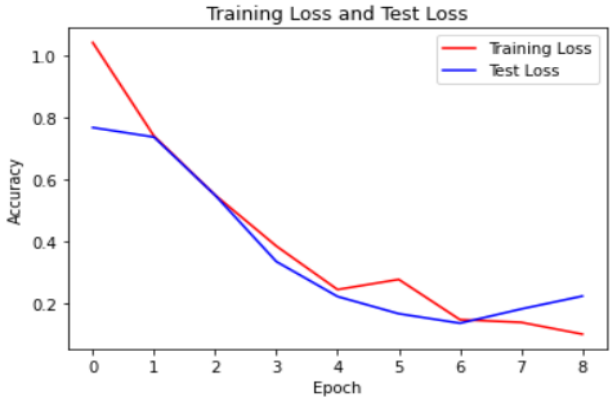

Fig. 5. Training and Test Loss of the Model

Figure 6 and 7 show the Confusion Matrix and Classification report for my Model. Test accuracy is really good with 93.2% and according to the classification report precision with 0.96 for paper, 0.90 for rock, and 0.94 for scissors is really good for all the three classes. Recall for paper with 0.88 is slightly low than rock with 0.98 which is really good and for scissors with 0.94 which is also good. F1 score for all the three classes with 0.92 for paper, 0.94 for both rock and scissors are also pretty good.

Test accuracy for all the three classes for my Model are as follows:

Test Accuracy for Class Paper = 87.6 %

Test Accuracy for Class Rock = 98.4 %

Test Accuracy for Class Scissors = 93.6 %

According to this information I can say my model does really well on all the three classes but test accuracy for class 'paper' is slightly lower than other two classes but I don't think it's that significant. It still does really well overall.

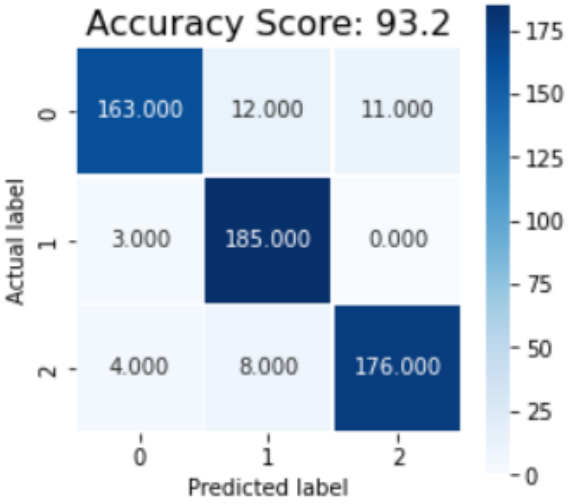

Fig. 6. Confusion Matrix of the Model

	precision	recall	f1-score	support
0	0.96	0.88	0.92	186
1	0.90	0.98	0.94	188
2	0.94	0.94	0.94	188
accuracy			0.93	562
macro avg	0.93	0.93	0.93	562
weighted avg	0.93	0.93	0.93	562

Fig. 7. Classification report of the Model

So overall, I would say my model performed really well on all the three classes and I think it's good enough to be used to play the game of rock, paper and scissors on the validation images, images which were not used in either training or testing of the model. I will also use my own images which I took with my own mobile camera to check how my model does on those images.

7. Playing the Game

After I finished building and training my model, I went ahead and played the game. The program asks the user to input 2 images and it predicts who wins player 1 or player 2 based on the images or if images are same then it says game is a draw. I played the game on my validation set 5 times and also played the game on my own images 5 times. On the validation set my model gave 4 out of 5

results correctly whereas on my own images my model gave 1 out of 5 results correctly.

Figure 8 show the execution of one of the games on my validation dataset and figure 9 show the execution of one of the games on my own images.

Fig. 8. Execution of the game on validation images

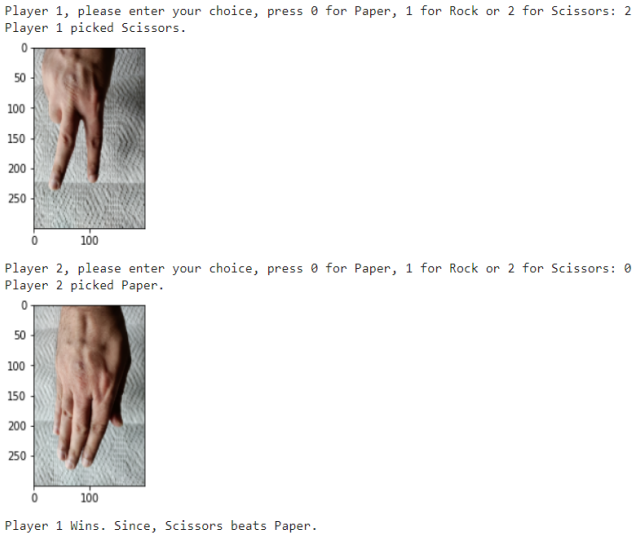

Fig. 9. Execution of the game on my own images

Overall, I would say my model performed really well on the validation images. Even though they were not part of training and testing set it was still able to predict the correct outcomes for the games most of the times. But for my own images it did not do so well, I think the reason it might have not done so well on these images is because they look quite different from the dataset I have for my original images. The background is different, original dataset has green background and my images has white background. Lighting is also quite different. Original images seem to have used artificial lights, whereas my images were taken in front of natural light. I wanted to keep this difference so I could test how well the model CNN model performs on images which might have similar structure but different lighting and background. Even for the results that it predicted incorrectly it was only because it predicted one of the images wrong but it predicted the other image correctly for the most part. But overall, I can say with this project that CNNs are a really powerful tool which we can use for various aspects and one of them being using them in video games.

8. Conclusion/Future work

To conclude, I would say my CNN model did really well overall. For the validation images it gave correct results for the most part. For my own images it wasn't able to perform that well and this is something I would like to work in the future, to see how can I improve my model to be able to predict images successfully that are a lot different from the ones used for training and testing. To see if from shapes and structures alone can CNN do just as well as it does on similar looking images. I would also like to work with CNNs for other games, such as chess and see how I can make use of CNNs and Neural Networks in general for these games.

9. References

[1] Christopher Clark, Amos Storkey, "Training Deep Convolutional Neural network to play Go", 2015. http://proceedings.mlr.press/v37/clark15.pdf

[2] Scott Lee, Aaron Isaksen, Christoffer Holmgard, Julian Togelius, "Predicting Resource Locations in Game Maps Using Deep Convolutional Neural Networks", 2021. https://www.aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14062/13611

[3] Barak Oshri, Nishith Khandwala, "Predicting Moves in Chess using Convolutional Neural Networks", 2015.

http://cs231n.stanford.edu/reports/2015/pdfs/ConvChess.pdf

[4] Link to the dataset:

https://www.kaggle.com/drgfreeman/rockpaperscissors