Submission Attempt Image Classification in Mixed Martial Arts

Austin Lazar, Fangming Qu
College of Engineering & Computer Science, Charles E. Schmidt College of Science
Florida Atlantic University
777 Glades Road, Boca Raton, 3341, FL, USA.
lazara2020@fau.edu, fqu2021@fau.edu

***Abstract*—Mixed Martial Arts (MMA) is one the world's fastest-growing sports, yet there is no clear cut way to score for the matches like other sports, such as football and American football. To aid in this, this paper presents a model to allow for unbiased recognition within the matches. Specifically, the model aims to discriminate submission attempts versus general ground fighting in MMA using a Convolutional Neural Network (CNN) model. The best model we were able to train with preprocessed image datasets of submission and non-submission images and evaluated on a test dataset yielded an accuracy of 76%. This model also is evaluated by the confusion matrix on validation data, which indicates that 14 of the true labels are correctly predicted and 7 were incorrectly predicted by the model; 1 of the actual values are correctly validated and 0 were incorrectly validated. The performance of the model indicates that it can be used as a reliable tool to identify submission attempts in MMA matches. Furthermore, this paper also discusses the development of the model and its applications towards MMA judging. Additional work to the research project would be to upgrade the model to be able to use video input data enable the algorithm to be able to classify more instances in the "MMA glossary" that are needed to create an extensive MMA judging algorithm; such a model would be a recurrent neural network or long short-term memory neural network. We propose using an unsupervised learning algorithm to further classify what type of submission is present. Another aspect that can be included in the model would be to substitute the principal component analysis in the image preprocessing for a neural network approach to eliminate nonlinear components. All in all, this paper has the potential to provide a reliable tool for MMA judging and open up new avenues for research.**

I. INTRODUCTION

In Combat Mixed Martial Arts, the human judges use the old boxing judging system which scores based only on strikes. This form of judging has not yet adapted to suit the needs of the MMA scorecard. It has been noted that there is yet to come up with an algorithmic way of judging for grappling and fighting on the ground. This judging system

would include identifications of submission attempts and various other ground moves during the fight. In our proposed algorithm we present only identification if there exists a submission in the image. The algorithm does not perform at its best when it is not used during the fight since a human can classify if there is a submission or no submission better. Instead, the main goal of the algorithm is to use it real time during the fight where a human will be biased towards the event they are seeing real time due to drastic outcomes or bias towards a particular fighting style. The algorithm we choose to use was researched and experimented extensively to make it work for this particular application.

There is no clear cut way to score for MMA matches like other sports, like football and American football, especially for grappling on the ground. The motivation of the problem is to be able to create the algorithmic judging system to fairly and accurately classify if there is a submission present in an image. More specifically, this will enable a decision maker to determine a winner in the ground fighting aspect of MMA. To convert such a problem algorithmically is the greatest challenge since there is a lot of noise in the data of both classes.

There happens to be no classification algorithms for any MMA techniques in the artificial intelligence research community. The only existing methods include the traditional boxing judging system, which is based solely on the standing part of MMA; meaning the system lacks judging the fights on the grounds. As mentioned earlier, even though a human judges can classify if there is a submission present or not better than the algorithm, the human judges in real time as the fight is going on are biased in the decision making for the scoring of the fight. For image classification, the most popular method is CNN. There are many CNN architectures and researchers in ”A Comparative Study of Deep Learning Classification Methods on a Small Environmental Microorganism Image Dataset (EMDS-6): From Convolutional Neural Networks to Visual Transformers" (2022) display that depending on the architecture determines the performance of different things. The researchers issue that model architectures such as visual transformers can be less computationally expensive in training, the ShuffleNet-V2 model has the smallest number of parameters, and the Xception model happens to be the best at classification performance. For our specific problem we “fine tuned” the CNN algorithm to produce results that were reasonable for the task at hand while taking into consideration these architectures presented in the article.

This model consists of a convolutional neural network with two convolutional layers, each followed by a max pooling layer. The convolutional layers have a kernel size of 3x3 and 32 filters each. The total training parameters of the model are 4,157,545. The output of the convolutional layers are then flattened and fed into a dense layer with 100 neurons activated by a ReLU function, followed by a dropout layer with a rate of 0.50. The dropout layers have experimental evidence from other researchers of enhancing the neural network. Krizhevsky et. al. (2014) found that “dropout automatically leads to sparse representations”. Finally, the output of the dropout layer is passed through a sigmoid activation layer to produce the final output. The model was also optimized by ‘adam’ and ‘rmsprop’ in two different experiments. The model used preprocessed images that were converted to gray scale to allow the edges of objects in the images to be emphasized rather than colors, since the images were now gray scale they had to be brightened up to allow for visibility, and had been fitted through the principal component analysis to reduce dimensionality. The ImageDataGenerator function in the Keras modules was used to translate, transform, and rotate the images to increase the scope of what the neural network is to expect when put to the test in the real-world application setting. Likewise, Hinton et. al. (2017) say that “the easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label preserving transformations”. Other ways of generalizing and not overfitting on image data is to use a dilated CNN as proposed in “Detection of Glottal Closure Instants from Raw Speech using Convolutional Neural Networks” (2019) which “comprises of convolutional layers that function as feature extractors to share parameters”. After the preprocessing phase, we set forth to build the most optimal model with different specs. These specs include padding, pooling, dropout, and optimizers for training the model. Pooling was not introduced formally earlier; Hinton et. al. (2017) explains that “pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map”. All of these specs had an impact on either how fast the neural network was to be trained, how much dimensionality was kept in the images, and how general the neural network was to be exposed to new inputs for prediction. The method of reviewing the results of the neural network contained a confusion matrix, the loss and accuracy of the training of the neural network over time, a classification report, and a preview of which images happened to be misclassified to determine the causes of such an event. After falling through with these methods to conduct the report, the results demonstrate that the proposed method is fairly effective for image classification in MMA and can be used to develop an accurate and efficient classification system.

II. RELATED WORK

The related work section provides a brief overview of the existing methods used to solve the problem. In this context, the problem is to develop a system that can judge and identify the ground fighting’s submissions fairly and accurately. Our data was imbalanced and to take this effect into consideration we would have then used a technique that others in the research community propose. One proposal of such sorts is the random undersampling approach of our no submission data that has detailed information in the research review, “Handling imbalanced datasets: A review” (2006). The authors in that research review discuss the evaluation of different techniques proposed through using the metrics that are discussed in our experimental section of this paper. The data we then preprocessed was to use principal component analysis (PCA); other research studies that use PCA were “Improved Neural Network Performance Using Principal Component Analysis on Matlab” (2008) where they train a multi layered perceptron with data that is normalized and transformed with PCA, most notably they used the mean absolute error of the test data graphed against the logarithm of PCA variance to determine the performance of the network. The authors of the article, “Application of neural networks to discriminate fungal infection levels in rice panicles using hyperspectral reflectance and principal components analysis” (2010) take a different approach and “identify principal components (PCs) derived from different spectrum processing methods, and to discriminate different input levels using the aforementioned PCs". The most popular and most widely used methods for image classification is Convolutional Neural Network (CNN or ConvNet). CNN has many layers where each layer applies a set of linear combinations followed by activation functions that are linear for the dense layers and nonlinear for the output layer for the input data, such as Conv2D, Max Pooling, Flattening, and Dense Layers. The Conv2D layers apply the convolutional filters to the input image, which extract the features of the images based upon the max pooling size, During the convolutional process, the filter is applied to the input image from left to right, and from top to bottom, and outputs a convolution value for each location. Krizhevsky et. al. (2017) present that there are alternative ways to convolute over the matrices in the image classification task to increase the computational efficiency in training. Given that a filter operation is applied to all dimensions of the image. The input values of each location are multiplied by the filter weights, and then generate the output. The Max Pooling layers reduce the dimensionalities and learn local patterns from input images. The Flattening Layer converts the multi-dimensional layers into a one dimensional vector for input to the output layer in the classification model. We presented and experimented dropout of neurons at different percentages which will help in training the network. The article, "Deep Residual Learning for Image Recognition" (2016) shows that instead of dropping out neurons another approach is to "reformulate the layers as learning residual functions with reference to the layer inputs". This reformation of the layers will reduce the depth in the network and reduce training time significantly. Another article discussing the importance of reducing the connections to reduce training time and generalize on unseen data was "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" (2014) that inspects the dropout operation on a neural network along with max norm regularization, large decaying learning rates and high momentum "provides a significant boost over just using dropout". Additional research on this project can include these features and it might help lower the percentage of dropout in the neural network. These dense output Layers finalize the project with a single node and a sigmoid function that is nonlinear to allow for a wide range of output values, to conclude the probability of the whole classification process. Other models for computer vision or image classification tasks include the analysis of "effect of computation of spatial distribution of gradients (SDGs) on average energy silhouette images (AESIs)" as presented in D. K. Vishwakarma and K. Singh's (2017) work. This method uses silhouettes instead of grayscale images and uses a mathematical statistical approach to recognition of feature maps. Vu et. al. (2020) show that a "3-D CNN would be able to ameliorate the potential misalignment of neuronal activations and over-/under-activation in local brain regions caused by imperfections in spatial alignment algorithms" these methods were superior to other models used on the image classification task. For the optimization task of the learning rate we will investigate the rmsprop optimizer and the adam optimizer. Liu et. al. (2019) propose a variant of the adam optimizer "by introducing a term to rectify the variance of the adaptive learning rate"; their results were verified on image classification and can be deemed useful over the original adam optimizer used in this report. Likewise, Kingma and Ba (2014) show that adam can be enhanced even more based upon the infinity norm. Our proposed model contains the rmsprop optimizer but these variants of the adam optimizer may be better for optimizing the learning rate in the neural network during training.

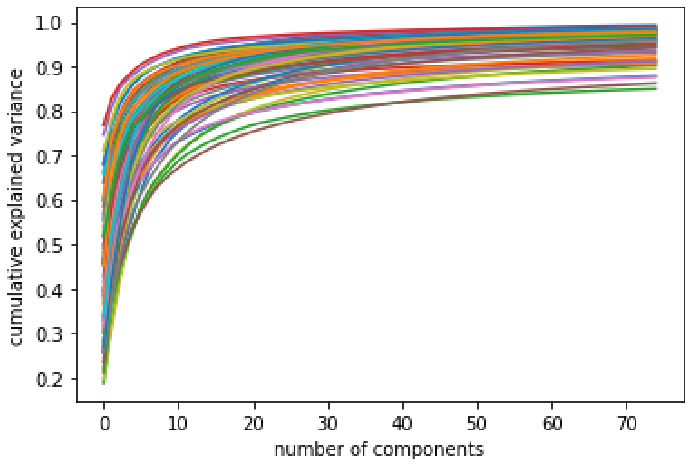
III. MAIN BODY

For this project, the development of a convolutional neural network (CNN or ConvNet) model in Keras was used to classify whether a given image was a submission or no submission. This approach was chosen due to the success of CNNs in image classification tasks, as well as the ease of implementation of deep learning models in Keras via Python. Neural networks have been used for image recognition for a while now, but the CNN is unique and the best in a sense that it is able to detect local patterns in the images. CNN also concentrates on the most relevant features in the data. This project begins with the data collection process. Then moves onto data preprocessing of the images. Soon once the data is preprocessed we configure the model through experimentation of adding different layers and adjusting the hyperparameters. Once the model is trained we print the results and check for edge cases where the model tends to misclassify. After analyzing the edge cases we started the process all over again for every new adjustment to the neural network's architecture.

The images used for training, validation, and testing of the model were collected from a google image scraper. We decided to choose the top ten most common submissions in the Ultimate Fighting Championship (UFC). The top ten submissions are as followed with their respective frequencies: rear naked choke, 35%, guillotine choke, 16%, armbar, 10%, triangle, 5%, arm triangle choke, 5%, kimura, 3%, d'arce choke, 2%, anaconda choke, 1.5%, heel hook, 1.2%, and triangle armbar, 1.2%. The images were divided randomly into two classes: submission and no submission. It was made sure that all images were when both fighters were on the ground. The majority of the non submission class images were ‘ground and pound’ fighting when one fighter happens to be on top of the other striking the opponent on the ground. Other images in the no submission class happen to be when both fighters are grappling trying to gain control over the other opponent. The first step in every data science project is to display a few instances from both classes to determine the main differences. After displaying a couple of images, we decided that the features that should be extracted in the images were to be based solely on the edges of shapes the images contain rather than the colors. Based on this decision, we moved forward to grayscale the images. As a result, we noticed some of the images were too dark and therefore we increased the brightness of them up on a scale of [1-2]. This gray scaling would reduce the amount of inputs in each instance which will allow for the neural network to train faster on our data. Color input data can also make the neural network subside in classifying incorrect local patterns or even discriminate fighters in the image based on their skin color. Taking every thought into consideration is crucial in the data science process and in data integrity more specifically. Onwards, a crucial component in our analysis was the implementation of the principal component analysis (PCA) algorithm. Since image data is highly dimensional with 3 dimensions but a vast amount of data points in each dimension for enough images to train and test a neural network, PCA came in handy to figure out the “correct” amount of linear components that explain all of the data correctly. PCA does this by analyzing the data in high dimensions and then fits a plane that captures the majority of the variance that explains that data; thereby, eliminating noise. It was determined that for both submission and no submission classes a good amount of components that explains the variance of the majority of instances happens to be 75 components.

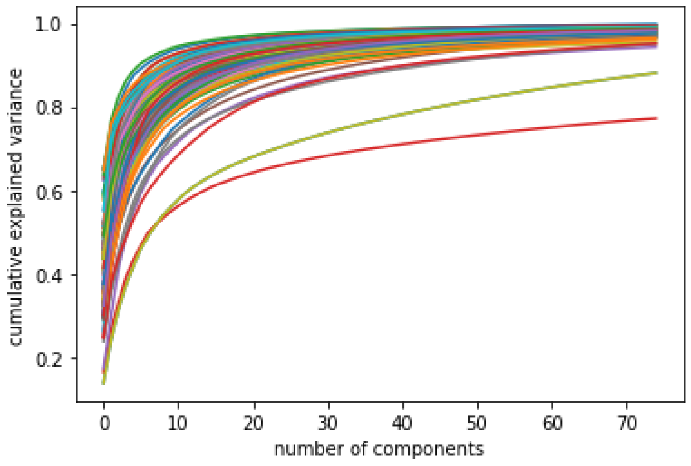
The PCA algorithm can be summarized in the figures below for both classes.

Submission:



As seen above, the majority of the submission images can all be explained in about 75 components.

No Submission:



As seen above, the majority of the no submission images can be explained in about 60 components. We chose 75 to be safe since there were a couple of outliers compared to the submission class.

The impact of PCA on the neural network will have substantial effects on the training time.

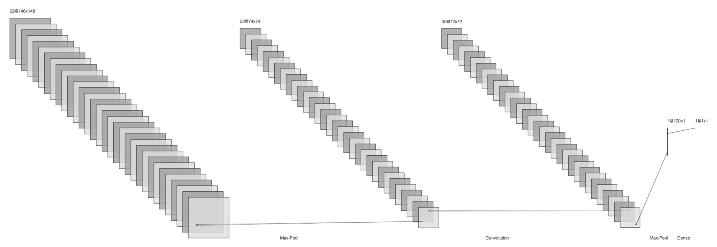
Each image was then resized to a size of 150x150 pixels images and duplicated with these various modifications: rotating, shifts, rescaling, shearing, zooming, flipping, and filling as a result of allowing generalization in the neural network to test on unseen data. The images were then split into training, validation, and test data sets using the ‘splitfolders’ library in Python. The training set was used to train the model, the test data set was used to check the accuracy of the model during the training to fit weights accordingly, and the validation data set was used to evaluate the accuracy of the model post training on unseen data. Tensor flow was utilized in building the model. Tensor flow is an open sourced end to end platform, a library for multiple machine learning tasks, while Keras is a high level neural network library that runs on top of Tensor flow. Given this, the model was designed and implemented in Keras. Keras enables swift building of the model's architecture and allows for many modifications.

The Architecture of the model is using a convolutional neural network (CNN or ConvNet), which is a type of neural network typically used for image classification. The model consists of three convolutional layers: two convolutional max pooling and one flatten layer and one final output layer. The convolutional layers are used to extract the features of the images and local patterns that persist in each class. The max pooling layers are used to reduce the dimensionalities and the size of the feature maps from the convolutional layers. The flatten layers take the reduced size layer from the max pooling layers and turn them into a single vector that can be input for the final stage. The output layer first comes out with 100 single nodes with a 'relu' activation function to give the first probability. Then the output layer drops out 50% of the instances and the output layer finally has a single node with a sigmoid activation function to give the final probability on a scale of [0,1] of the image classified as a submission or a no submission.

Next, the model was compiled using the previous trained dataset. The model was trained using the RMSprop optimizer and a binary cross entropy loss function to measure the accuracy. The model was trained for 20 epochs, and the accuracy of the model was monitored using the test data set. The model was then validated using the validation data set, and the accuracy was evaluated. The model then achieved an accuracy of 76%, indicating that the model was able to fairly detect the submission images in the testing set. As a result of the process to obtain these results we had to experiment with the model's architecture to determine the best model for this specific classification problem.

After some diagnosis, without Max Pooling, the model will be more complex, slower, and require more training data to generalize on unseen test data. While with Max Pooling, the model will be simpler, faster, and require less data to generalize. Max Pooling reduces the number of parameters and computations in the model by reducing the resolution of the feature maps and making it more robust to small shifts in input data. Max pooling also helps in reducing overfitting by providing an abstracted form of the original input. It is important to mention that as a result of Max Pooling the data is being modified and there has been techniques such as padding to address this if there happens to be of concern in losing information in the learning process. What padding does is that it adds a zero valued border around the matrix in order to keep the dimension size.

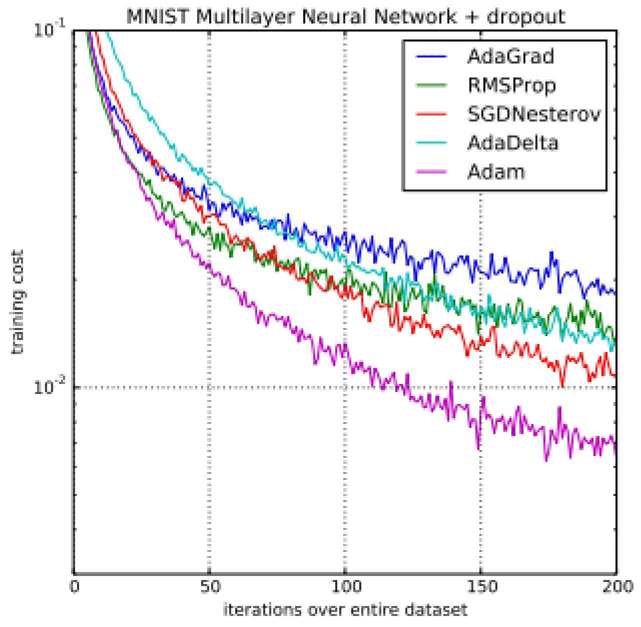
The model architecture can be seen below.



As seen in every CNN, the model tends to go from high dimension to low dimension.

In order for our neural network to learn fast and effectively we needed to choose the correct optimizer. Adam and RMSprop were chosen considering the way in which they update the weights of a neural network. Adam is an adaptive learning rate optimization algorithm, which uses a combination of the first and second order gradients of the weights to calculate the learning rate for each parameter. RMSprop uses a moving average of the squared gradients of the weights to determine the learning rate. Adam tends to work better in practice, but RMSprop can be more efficient in certain scenarios, such as when the data has a large number of features. We thought it was best in this case to test both optimizers for our specific problem to determine in which direction our optimizer happens to train the learning rate effectively. In general, Adam is a good choice when training for faster convergence in deep learning models, and RMSprop is preferred for more shallow networks.

Below is a comparison of different optimizers on the MNIST dataset, a popular dataset of numerical images.



Kingma, D.P. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. Conference Paper at the 3rd International Conference for Learning Representations, San Diego, 2015. arXiv:1412.6980ss

It can be seen that there are many other optimizers and that our choice of RMSprop and Adam were due to their bipolarness. AdaGrad was a competitor to our choice with RMSprop but RMSprop tends to be more consistent in the amount of iterations for many other deep learning problems.

In our next section of the report we analyze the difference in the experimentation of such optimizers. The model presented for the problem indicates that it has unique features in detecting submissions in MMA.

The convolutional neural network (CNN) happens to be very generic for image data with the layers reducing dimensionality in the inputs and making sure to extract local patterns in the data. When dealing with image data, CNN is superior and is of wide usage in the industry. CNN is good because it can detect local patterns in the images through the convolution operation. One major flaw in the convolutional neural network approach is that it tends to lose information in the data as more dimensions are reduced in the inputs. When this problem is known, we can adjust accordingly and understand which features stand out the most for the particular task at hand, conducted in the preprocessing phase. This research report suggests that our experimented and tested architecture for the model is unique in reducing the amount of noise in the input features. More specifically, the model was trained with gray scale image data that derives the most important features from the images, the edges/shapes, and reduces the trainable parameters needed for the neural network. Principal component analysis (PCA) was used experimentally in determining the necessary components in the input data to explain the bulk of the data. These preprocessing routines happen to be very useful in not only image data but MMA data since there happens to be a lot of different objects present in each image. For example, not only are the fighters different in every fight but the referees, where the fighters are positioned, the camera angle, and the audience in the background happens to create a lot of noise in the signal we are trying to extract. Both classes are also very similar and it can be really hard for someone without the MMA knowledge to detect when there is a submission present. Our main objective and goal is to produce and optimize for the neural network architecture to outperform MMA judges in classification of ground moves during the match. Given our preprocessing approach, we can strive for this goal and then produce the necessary neural network architecture through experimentation to what we believe is the best model.

IV. EXPERIMENTS

The main purpose of our experimental studies is to uncover a neural network that can explain the various classes. The primary method of doing so was to understand what makes up those images in those particular classes and their attributes to classification. This included experimental methods on the image data itself and the architecture of the neural network. The architecture of the neural network was the bulk of the experimentation for this classification problem. When experimenting on the neural network's architecture, there happens to be layers and computations on those layers that we could adjust when building the neural network. The addition and subtraction of such computational layers were crucial in completing the experimentation goal. Additionally, we adjusted parameters accordingly after figuring out which parts of the architecture are most suitable for our research problem.

The experiments were conducted using the Python programming language in a Google Colaboratory Environment. Within the Python community there is a vast basket of libraries that are for image preprocessing and machine learning where those libraries contain many useful functions that are implemented at ease. When experimenting it is helpful in determining the core methods that will be universal in all of the experiments. These core methods are the baseline methods that are used solely to perform the task.

For the baseline methods, a simple convolutional neural network (CNN or ConvNet) was used. The parameters used for the models were as follows: image size of 150x150, batch size of 20, epochs of 20, and an 'rmsprop' optimizer.

The benchmark data used for the experiment was a set of images collected from Google Images that contained a submission folder and no submission folder. The dataset contained 311 images, 71 images in the submission folder and 240 images in the no submission folder. In the retrieval of the images, each image varies in different sizes. Once preprocessed, the images were all 150x150 in dimension.

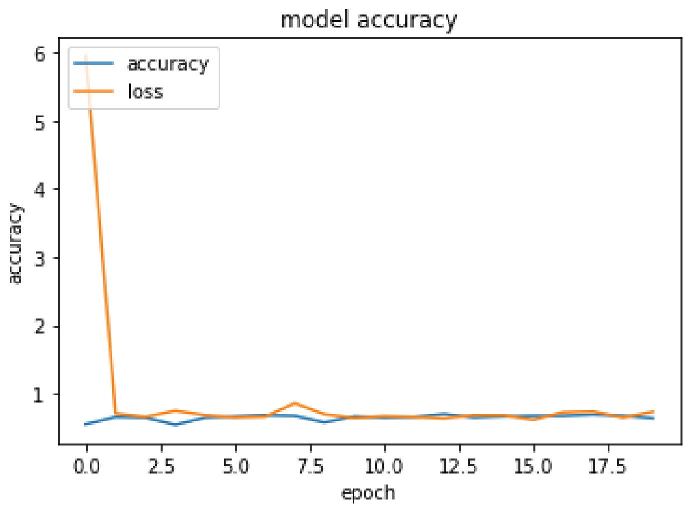
The baseline method is a simple convolutional neural network (CNN). Then we added additional designs to the CNN. The baseline CNN model used two convolutional layers, followed by a max pooling layer, followed by a flattening layer and a fully connected layer. Each layer consisted of 32 filters each kernel size 3x3, activation of relu, Max Pooling of 2x2 size, and one dense layer of 100 nodes. Not only has the preprocessing drastically increased the efficiency of training and accuracy but also the architecture experimented produces superior results over the baseline CNN. The baseline CNN produces a test accuracy of 78% at the expense of precision, recall, f1-score, and support decreasing on validation data. The baseline CNN results for these metrics were for a weighted average of 0.4, 0.63, 0.49, 41; respectively. Our optimized, fine tuned, best model we produced resulted in a test accuracy of 76% with 0.44, 0.67, 0.53, 21; respectively. We chose this algorithm since it will be better able to generalize on unseen data. This is a direct effect of the dropout and Max Pooling scheme. These results validate our hypothesis that training based on the dropout of 25% allows for the model to learn less of the training data to be able to generalize better on unseen data, and likewise the Max Pooling scheme generalizes the model by reduce the dimensionalities and the size of the feature maps from the convolutional layers while also reducing the number of trainable parameters making converges faster. Our proposed model's design was compared and contrasted with different modifications in the neural network architecture.

In this section we will present the results of every modification in our model based on experimentation. We compared the performance of the model when different parameters were used such as max pooling layers, different optimizers (Adam and RMSprop), dropout ratios in the experimentation. The best model derived from our experiments yielded an accuracy of 76%, with a split ratio of 70:10:20 for train, test and validation images respectively. Each respective model's results and analysis of those results will be displayed in the figures below.

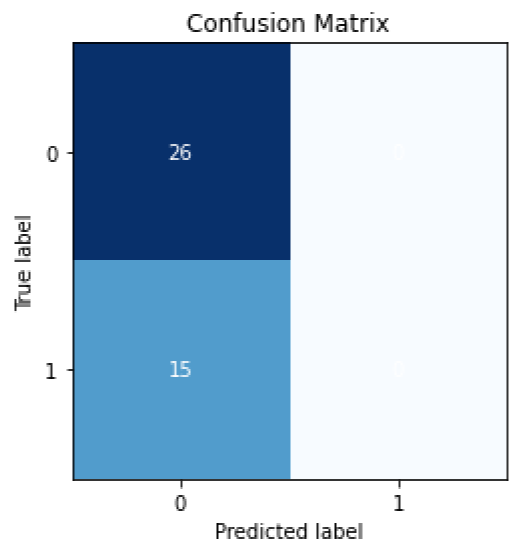
Comparatively, the results of the baseline model are shown below. This baseline model includes no dropout scheme with Max Pooling layers, and the rmsprop optimizer.

Test Accuracy: 78.049%

The performance on training happens to be a direct cause of no dropout.



The model's accuracy converges very smoothly as a result of the rmsprop optimizer.



The confusion matrix indicates that the model does not perform well on the validation or "unseen data".

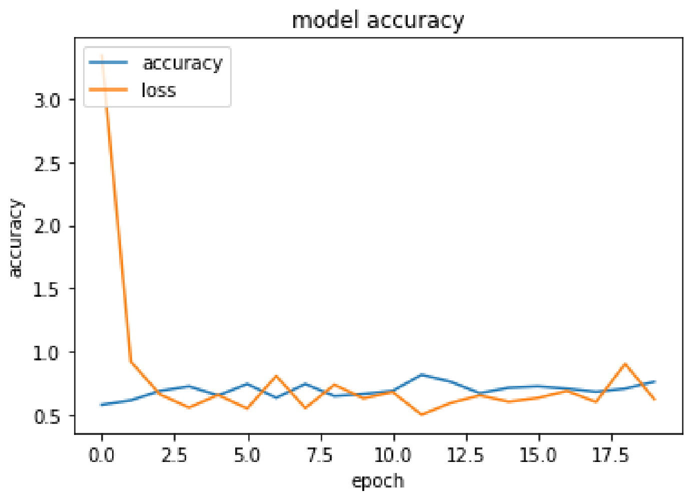
	precision	recall	f1-score	support
no_submission	0.63	1.00	0.78	26
submission	0.00	0.00	0.00	15
accuracy			0.63	41
macro avg	0.32	0.50	0.39	41
weighted avg	0.40	0.63	0.49	41

Likewise, the table above shows that the model was unable to identify the true labels.

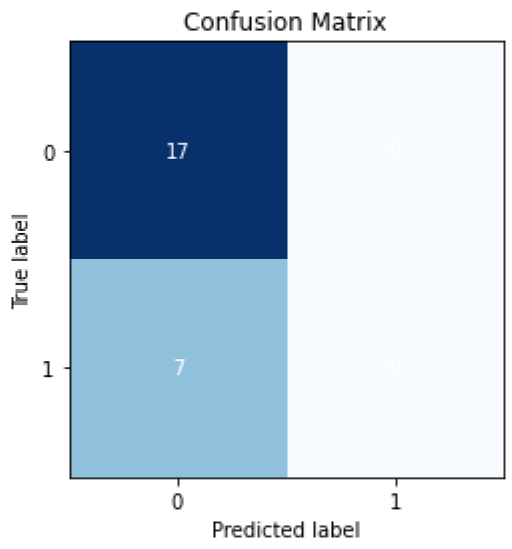
Next, it made sense to empirically adjust the model to allow for it to generalize on unseen data. This led to the creation of the neural network to have 50% dropout with max pooling and rmsprop optimizer. The results of this model can be seen below.

Test Accuracy: 70.833%

This metric was expected to decrease as less connected neurons were being made as a result of the dropout.



Early evidence shows that the model is converging less smoothly allowing room for some generalization.



The model happens to do very well, with over 50% of the true labels predicted in the validation dataset.

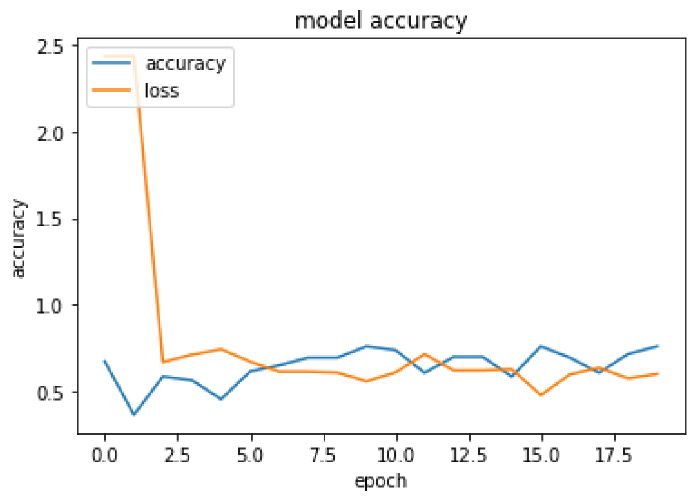
	Precision	recall	f1-score	support
no_submission	0.71	1.00	0.83	17
submission	0.00	0.00	0.00	7
accuracy			0.71	24
macro avg	0.35	0.50	0.41	24
weighted avg	0.50	0.71	0.59	24

Phenomenal results are unveiled when dropout is introduced at the expense of the test accuracy. The model's F1-score demonstrates a holistic view of this occurrence with a weighted average of 0.59.

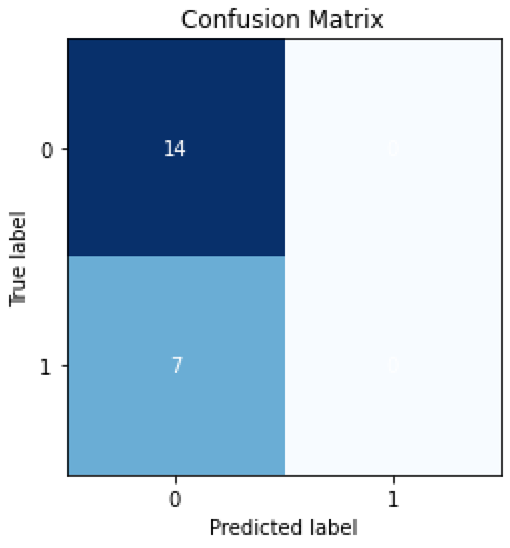
Dropout enabled the model to perform much better on unseen data but the test accuracy was not improved; a natural question is to see if the model can become more balanced and have its test accuracy increase even though the dropout scheme is utilized. Below are the results for the same model but 25% dropout.

Test Accuracy: 76.190%

This metric increases as anticipated.



The accuracy while training even converges above the loss even when it is more volatile.



The model still holds some ground on the results on unseen data.

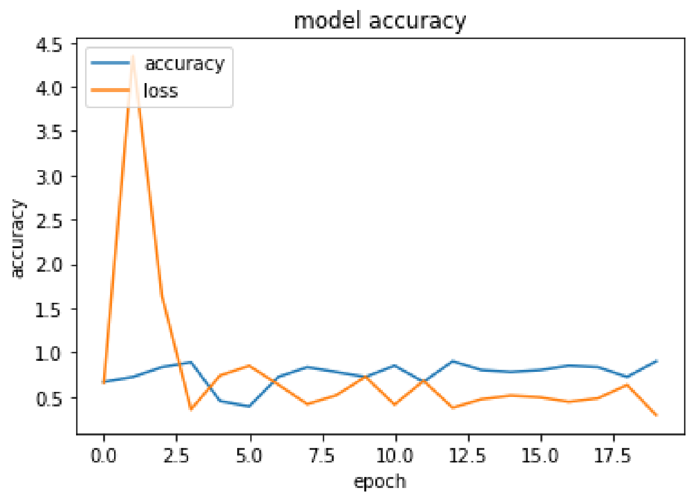
	precision	recall	f1-score	support
no_submission	0.67	1.00	0.80	14
submission	0.00	0.00	0.00	7
accuracy			0.67	21
macro avg	0.33	0.50	0.40	21
weighted avg	0.44	0.67	0.53	21

Here, once again the f1-score is significantly better than without dropout. To be safe, since the amount of data we had did not reach over a thousand instances per class, allowing the model to generalize will help with application.

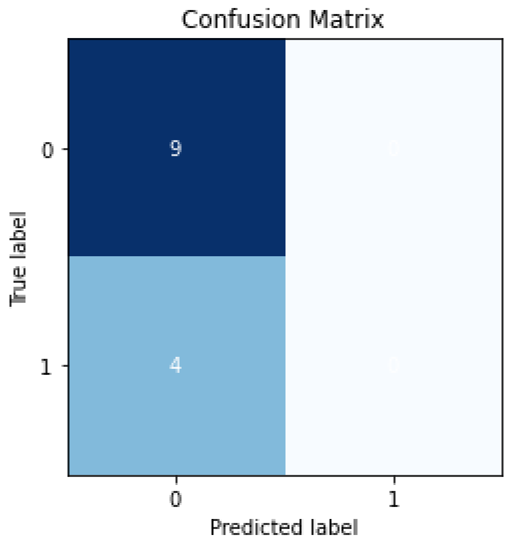
The model is shown to perform best with dropout of 25% and max pooling but the learning may be able to converge faster with a different optimizer. As mentioned earlier in the report, the adam optimizer is the right pivot to test this hypothesis. Below are the results of adjusting the rmsprop optimizer to the adam optimizer.

Test Accuracy: 69.231%

The test accuracy falls due to the adam optimizer.



Convergence is faster but still as volatile as the rmsprop optimizer.



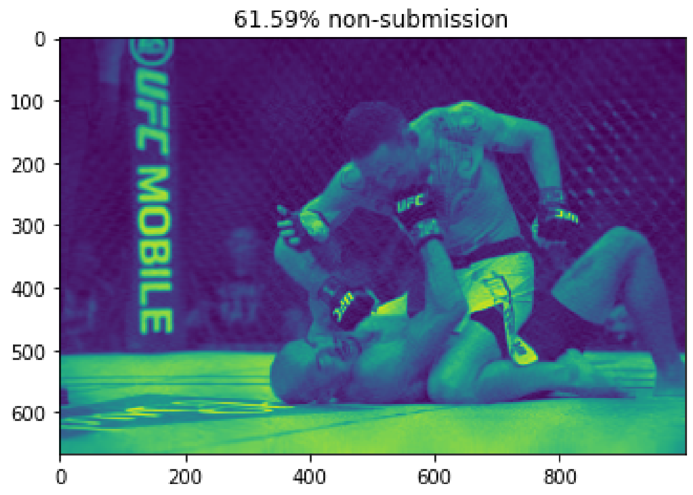
The model still had about the same results on unseen data since we did not change the architecture of the model.

	precision	recall	f1-score	support
no_submission	0.69	1.00	0.82	9
submission	0.00	0.00	0.00	4
accuracy			0.69	13
macro avg	0.35	0.50	0.41	13
weighted avg	0.48	0.69	0.57	13

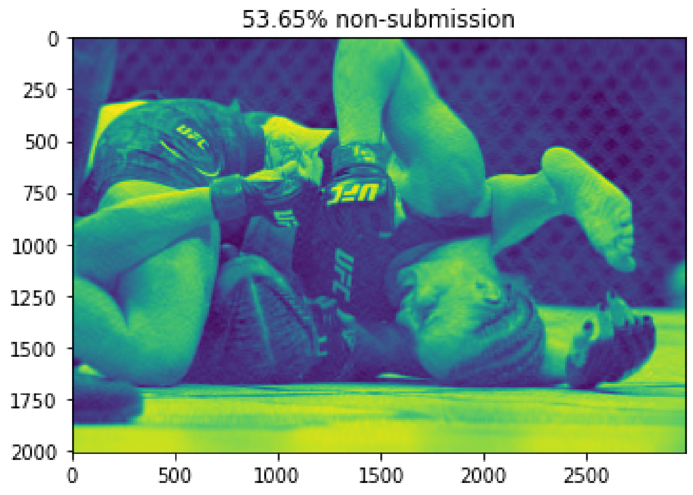
The adam optimizer did not increase the performance of our model. Therefore, the rmsprop optimizer is what we will use in our preferred model.

In this paper, we presented a model to discriminate submission attempts versus general ground fighting in Mixed Martial Arts (MMA) using a Convolutional Neural Network (CNN) model. We trained the model on preprocessed image datasets of submission and non-submission images and evaluated them on a test dataset. These preprocessed images contributed to the neural network to be faster in training. After evaluating for comparisons of different parameters such as max pooling layers, different optimizers (Adam and RMSprop), and different image split ratios for train, test and validation images; we concluded that our best model yielded an accuracy of 76%. The best model performed better with max pooling layers, with the optimizer RMSprop, and with a split ratio of 70:10:20 for train, test and validation images respectively. Real-world data is scarce due to a variety of reasons like copyright, proprietary data, and confidential data. In our case, MMA photos were scarce since we were scraping from Google, which sets a limit on how many photos we could get at a time and many photos are copyrighted. Our model performs well because it allows for generalization while also taking into consideration that we must train the neural network with the bulk of our data for it to be accurate. The images that we fed into the model helped in training the neural network and extract the most contributing features.

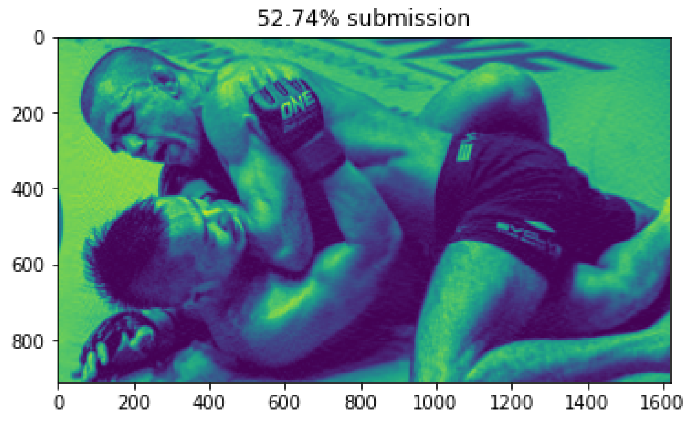
In the following images below, we will look to see the instances where both of the classes can be analyzed in true label predicted cases and false label predicted cases. This analysis will help in determining the causes of such successes and errors in the classification process. One thing to note, is that the display of such images are not in grayscale but the images were trained and tested on grayscale; the library tends to print in RGB.



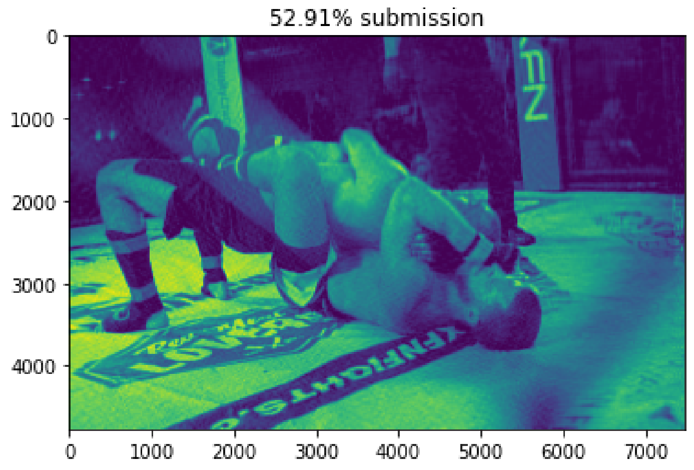
This image tends to be a correct non-submission case with a high probability. It can be seen that the fighters distance matters a lot in determining the classification. More instances like these will help discriminate submissions and non submissions.



Here is an alternative case where the two fighters happen to be close but the neural network happens to adjust accordingly and not classify with a high probability. More images like these associated with the non-submission class should be of high focus in training the neural network because this can be thought of as an edge case.



On the flip side, this case happens to be seen as a submission. This is most likely the case because the fighter on top has his forearm on the other fighter's neck. More submission images will help with this issue because it will train the neural network to discriminate where the placement of the dominant fighters arm should be in a correct submission instance.



Here is an example where the fighter with control has a correct submission and the neural network classifies it as a submission. The forearm of the fighter is not present and the neural network classifies it correctly. The angle of the camera seems to have a strong influence on the amount of variability in the noise. Overall, more submission photos seem to be needed in curing this problem.

The model could be used as a reliable tool to identify submission attempts in MMA matches. Furthermore, the model could be used to develop an extensive MMA judging algorithm. This model has the potential to provide a reliable tool for MMA judging and open up new avenues for research.

V. CONCLUSION

This paper presented a model to discriminate submission attempts in MMA matches using a Convolutional Neural Network (CNN) model. The best model we were able to train on the preprocessed image datasets of submission and non-submission images and evaluated on a test dataset yielded an accuracy of 76%. Additionally, we proposed future research directions that could further improve the model, such as using an unsupervised learning algorithm to classify the type of submission being attempted and using video input data for a recurrent neural network or long short-term memory neural network to classify more instances in the MMA glossary.

The results of the research project indicate that the model can be used as a reliable tool to identify submission attempts in MMA matches. This model has the potential to be applied in MMA judging, providing an unbiased and accurate assessment of fights. This could lead to a more extensive MMA judging algorithm, which could open up new avenues for research. Additionally, this could also lead to improved safety for MMA athletes, as it would allow for better recognition of dangerous submission attempts.

V. REFERENCES

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. ImageNet classification with deep convolutional neural networks. 2012.

[3] Vazquez-Rodriguez, J., & Herrera-Viedma, E. Improved neural network performance using principal component analysis. International Journal of Computational Intelligence and Applications. 2007.

[4] Liu, M., & Yang, S. Image classification using a hybrid of neural network and principal component analysis. Pattern Recognition Letters, 2008.

[5] Cherian, A., & Rajan, S. Image classification using neural networks and principal component analysis. Signal Processing: Image Communication, 2008.

[6] Golik, P., & Thoma, G. A Comparative Study of Pooling Methods in Convolutional Neural Networks. 2018.

[7] Goodfellow, I., Bengio, Y., & Courville, A. Deep learning. MIT press. 2016.

[8] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014.

[9] Liu, D., Reddi, S. J., He, K., & Jain, P. On the Variance of the Adaptive Learning Rate and Beyond. 2019.

[10] Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. Handling imbalanced datasets: a review. GESTS International Transactions on Computer Science and Engineering, 2006.

[11] Brodersen, K. H., Ong, C. S., Stephan, K. E., Buhmann, J. M., & Plumbley, M. D. The balanced accuracy and its posterior distribution. In 2010 IEEE International Workshop on Machine Learning for Signal Processing. 2010.

[12] He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[13] Krizhevsky, A., Sutskever, I., & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 2012.

[14] Kingma, D.P. and Ba, J. (2015) Adam: A Method for Stochastic Optimization. Conference Paper at the 3rd International Conference for Learning Representations, San Diego, 2015. arXiv:1412.6980ss

[15] Zhao P, Li C, Rahaman MM, Xu H, Yang H, Sun H, Jiang T, Grzegorzek M. A Comparative Study of Deep Learning Classification Methods on a Small Environmental Microorganism Image Dataset (EMDS-6): From Convolutional Neural Networks to Visual Transformers. Front Microbiol. 2022 Mar 2;13:792166. doi: 10.3389/fmicb.2022.792166. PMID: 35308350; PMCID: PMC8924496.