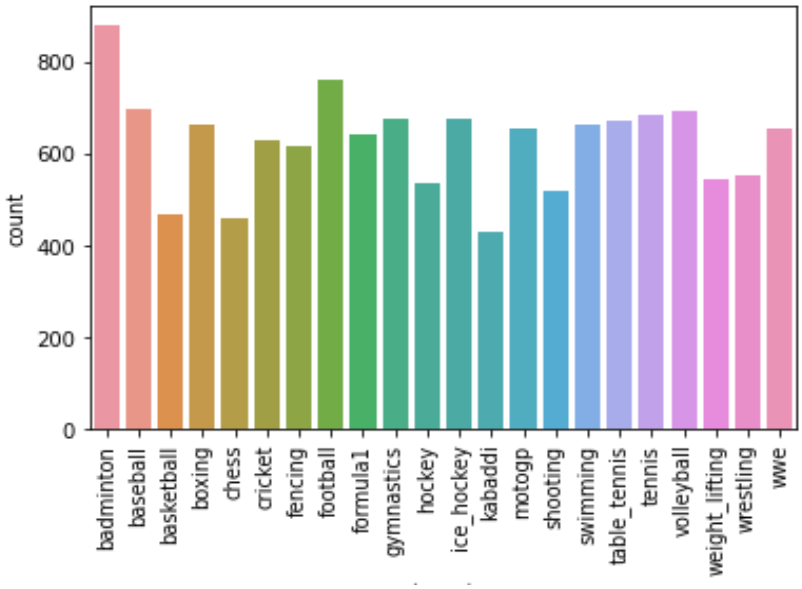
Sports Image Classification

Leticia Garcia & Danielle Strejc

Introduction

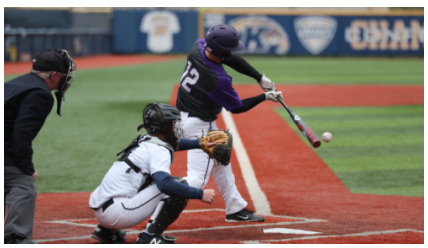
Sports have the ability to bring people together in a way that many forms of entertainment cannot. As two sports fans, we wanted to tie together our skills with image classification and deep learning with something we enjoy watching- sports. The purpose of this project is to see how accurately our model can distinguish between similar sports. We will compare between: cricket and baseball; table tennis, tennis, and badminton; boxing and WWE; and swimming versus basketball as a control. Additionally, we will look at the results for the overall dataset with its 22 labels to better understand how well the model is working.

To the right is a chart with the label distributions of the overall dataset. It is pretty well balanced, and we made sure to choose comparisons for the most part that were close to a balanced distribution.



A little background on our dataset: the size is 701MB and there are 13,800 images representing 22 unique sport categories. The data is in the form of images and a csv file comes separately to help us label our sports with the corresponding path to each image. We found this dataset on Kaggle.

The images can be of players actively engaging in the sport, equipment, the field or venue of the sport, or even promotional material from games. The variety of images makes us curious about how accurate the models will be predicting the specific sports.



In the remaining sections we will go over background research and how that influenced what we tried, set up and models used, evaluation and results, and a conclusion with what we learned, struggled with, and what we would like to see done in the future.

Background Research

Before diving into the modeling, we wanted to get a clear idea of what other work has been done on sport image classification.

From Russo et. al. (2018)[1], we realized that we should try an ReLU activation layer in addition to trying out RMSprop as an optimizer and utilizing an 80/10/10 split for training, testing, and validation. They were able to get accuracy in the high 90's for image classification from video broadcasts which gave us a target for our own models.

Kesorn and Poslad (2009)[2] attained accuracies in the high-80's and also gave us ideas for image-preprocessing such as rotation and cropping. Their process looks at images from a video broadcast and categorizes them by sport, which is similar to what we are doing, except our dataset is already all labeled for us.

Podgorelec et. al. (2020)[3] experiment with different CNN models with various hyperparameters which is something that we incorporated in our model. The research gives accuracy in the mid-70's to high-80's for variations of the models that they incorporated. In addition to doing CNN models with hyperparameters, they also try fine tuning their models afterwards and as well cross validation was used within their models.

Zhang (2021)[4] experiments with different sizes of images as well as the rotation of the image. He also tries different sampling sizes which is something that we incorporate into our project. In doing this we wanted to figure out if giving the CNN model fewer images in the training set would make a difference in accuracy. This research paper also gave us a better understanding of how the model works and how many layers we wanted to apply.

Models and Evaluation

To process the images, we apply a random cropping to the images of the input size that we wanted to try out for that given step- we do 64x64, 128x128, and 256x256 to experiment with what gives the best results. Once we have our images ready, we create the training, testing, and

---

[1] Russo, M. A, Filonenko, A. and K. Jo, "Sports Classification in Sequential Frames Using CNN and RNN," 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT), Busan, Korea (South), 2018, pp. 1-3, doi: 10.1109/ICT-ROBOT.2018.8549884.

[2] Kesorn, Kraisak, and Stefan Poslad. "Enhanced Sports Image Annotation and Retrieval Based Upon Semantic Analysis of Multimodal Cues." *Advances in Image and Video Technology*, Springer Berlin Heidelberg, pp. 817–28, doi:10.1007/978-3-540-92957-4_71.

[3] Podgorelec, V., Pečnik, P., & Vrbančič, G. (2020). Classification of Similar Sports Images Using Convolutional Neural Network with Hyper-Parameter Optimization. *Applied Sciences*, *10*(23), 8494. https://doi.org/10.3390/app10238494

[4] Zhang, X. (2021). Application of Convolution Network Model Based on Deep Learning in Sports Image Information Detection. *E3S Web of Conferences*, *233*, 02024. https://doi.org/10.1051/e3sconf/202123302024

validation batches and check that those load properly. We set our batch size to 32, and define our classification model

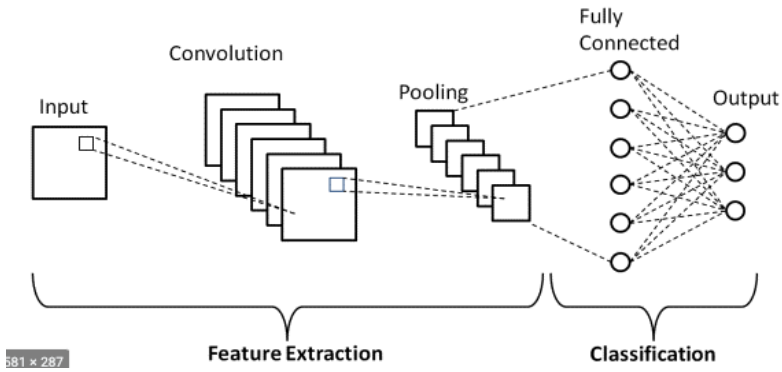
We started with a simple feedforward neural network with the chosen image size groups that we decided on. Before creating the neural network, we split the dataset into 80/10/10 and 70/20/10. We provide iterable over the DataLoader() function we created and set a batch size. The image below shows an example of our training batch when we look at the entire dataframe.

```
images.shape: torch.Size([64, 3, 64, 64])
```

We created an ImageClassificationBase class and built on top of a nn.module, with a training set and a validation method. We extended the ImageClassicationBase with a constructor and a forward() method with a 5-layer architecture.

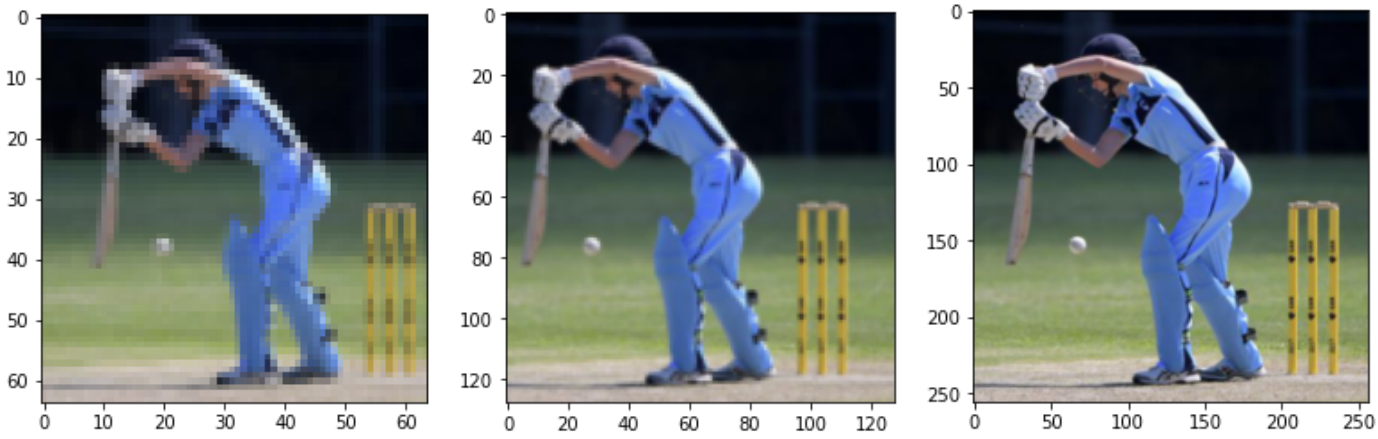
```
class SportsImageLinearModel(ImageClassificationBase):
    def __init__(self, input_size, output_size):
        super().__init__()
        self.linear1 = nn.Linear(input_size, 1024)
        self.linear2 = nn.Linear(1024, 512)
        self.linear3 = nn.Linear(512, 256)
        self.linear4 = nn.Linear(256, 32)
        self.linear5 = nn.Linear(32, output_size)

    def forward(self, xb):
        # Flatten images into vectors
        out = xb.view(xb.size(0), -1)
        out = self.linear1(out)
        out = F.relu(out)
        out = self.linear2(out)
        out = F.relu(out)
        out = self.linear3(out)
        out = F.relu(out)
        out = self.linear4(out)
        out = F.relu(out)
        out = self.linear5(out)
        return out
```



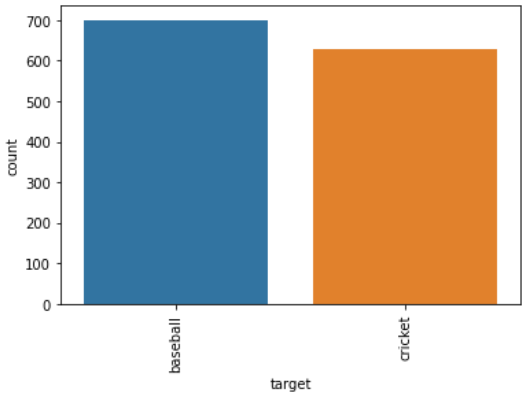
Once we had the model ready, we set up the optimization function. We wanted to try SGD, Adam, and RMSprop since those were three that we are familiar with and saw that some of our articles mentioned. Then, we were ready for training. The hyperparameters further tuned

were epochs and learning rate. The following section will detail more about what we changed to improve our results. Below is an example of how the images differ based on pixel size.



Results

Cricket and Baseball



The first classification task that we look into is cricket versus baseball. The two sports are similar in images and label split, so we wanted to see how well a model can pick out the differences. Here we compare between learning rates, input image size, and optimizer type. We found that no matter what we tried, 50 epochs was the best so that is all that is reflected in the table below. For a complete view of what we tested, see **Tables 1 and 2** in the appendix.

Sometimes we got better results with a smaller image size or learning rate, and other times the largest and clearest images gave us the best accuracy. We could count on the highest accuracy being in the high-70's

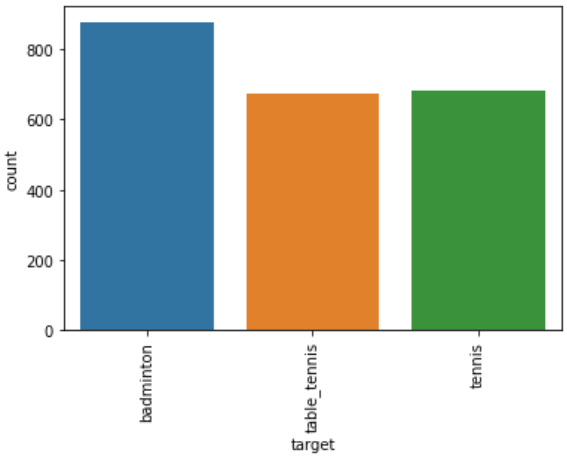
Accuracies when Classifying Cricket vs Baseball- 80/10/10 Split				
	Learning Rate (.001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.6990
Model 2	.0001	128x128	SGD	0.7146
Model 3	.0001	64x64	SGD	0.7826
Model 4	.0001	256x256	SGD	0.4260
Model 5	.0001	256 x 256	Rmsprop	0.7198
Model 6	.0001	64x64	Rmsprop	0.7615
Model 7	.0001	128x128	Rmsprop	0.7198
Model 8	.0001	64x64	Adam	0.7719
Model 9	.0001	128x128	Adam	0.7302
Model 10	.0001	256 x 256	Adam	0.7250

Accuracies when Classifying Cricket vs Baseball- 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.7188
Model 2	.0001	128x128	SGD	0.6625
Model 3	.001	256 x 256	SGD	0.7667
Model 4	.001	64 x 64	SGD	0.6323
Model 5	.001	64 x 64	Rmsprop	0.6375
Model 6	.001	256 x 256	Rmsprop	0.7875
Model 7	.001	128x128	Rmsprop	0.7198
Model 8	.001	64x64	Adam	0.6885
Model 9	.001	128x128	Adam	0.7094
Model 10	.001	256 x 256	Adam	0.8385

and even into the low 80's. It appeared that having the 70/20/10 split improved accuracy by several percentage points. We found these results to be relatively good, since the two sports are similar in how they are played and some of the equipment used. We also took into account the fact that this dataset we had about 1330 pictures, which we will compare with the other dataset sizes. In the following section we will compare three racquet sports.

Table Tennis, Tennis, and Badmintон

We next wanted to compare between three racquet sports: table tennis, tennis, and badminton. The labels are only somewhat imbalanced with badminton having over 800 images and table tennis and tennis each having just over 600. We were thinking that this would be even more difficult for our model to distinguish between. Our hypothesis was correct, and we were only able to achieve accuracies in the high 60's. For this dataset, the smaller images performed better than the most clear 256x256 ones. Once again, we could not determine an optimal learning rate, since with each training split, a different one was the best performing. Despite the lower than desired accuracy, we are pleased with the results and next move onto a comparison with combat sports. **Tables 3 and 4** feature more details of our trials and results from this data split.



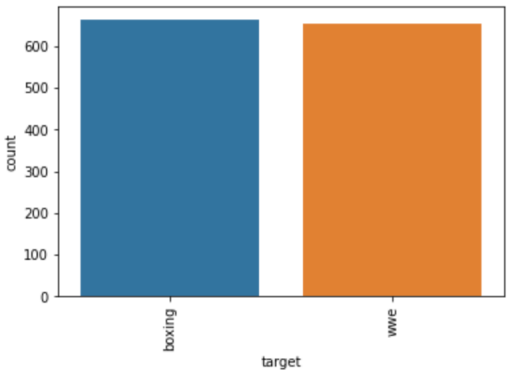
Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 80/10/10 Split				
	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	.001	128x128	SGD	0.6391
Model 2	.0001	128x128	SGD	0.6115
Model 3	.001	64x64	SGD	0.5207
Model 4	.001	256x256	SGD	0.4931
Model 5	.001	256 x 256	Rmsprop	0.6227
Model 6	.001	64x64	Rmsprop	0.6967
Model 7	.001	128x128	Rmsprop	0.6975
Model 8	.001	64x64	Adam	0.6620
Model 9	.001	128x128	Adam	0.6542
Model 10	.001	256 x 256	Adam	0.6789

Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	.0001	128x128	SGD	0.6578
Model 2	.001	256 x 256	SGD	0.4945
Model 3	.0001	64 x 64	SGD	0.6713
Model 4	.001	64 x 64	SGD	0.6355
Model 5	.0001	64 x 64	Rmsprop	0.6422
Model 6	.0001	256 x 256	Rmsprop	0.6421
Model 7	.0001	128x128	Rmsprop	0.6398
Model 8	.0001	64x64	Adam	0.6399
Model 9	.0001	128x128	Adam	0.6199
Model 10	.0001	256 x 256	Adam	0.6555

Boxing and WWE

Compared to the other two datasets, this one is a bit different when comparing each other. We expected the accuracy would be low due to both sports involving personal physical

movement and the lack of equipment compared to the other sports. The results ended up being surprisingly good given that some of the images are just people standing or their faces, and has nothing to do with the actual sports. We were getting accuracies with a wide range (60s- low 80s) which tells us just how much influence a good model and optimizer has on the results of an image classification task. We see in the above results that split 80/10/10 had better results compared to 70/20/10 and that amount of images in the training set. We see that the optimizer set as Adam and 256 x 256 input sizes did best in terms of accuracy. The image on our left is the distribution of images between the two sports. We see that the distribution between the two is evenly spread out and there are about 1,318 pictures within this dataset. Below are the top ten highest accuracies and top two highlighted for each split. The full results are shown below in **Tables 5 and 6.** In the next set, we wanted to do a control of sorts by contrasting two sports that are nothing alike to see what type of accuracies we could get.

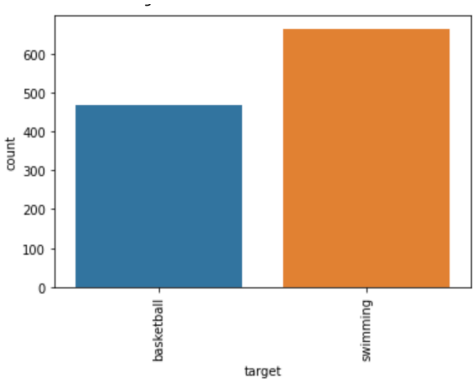


Accuracies when Classifying WWE vs. Boxing - 80/10/10 Split				
	Learning Rate (.001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.6597
Model 2	.001	128x128	SGD	0.6708
Model 3	.001	128x128	SGD	0.6962
Model 4	.0001	128x128	SGD	0.6858
Model 5	.001	64x64	SGD	0.7240
Model 6	.001	256x256	SGD	0.7378
Model 7	.001	256 x 256	Rmsprop	0.7552
Model 8	.001	128x128	Rmsprop	0.6285
Model 9	.001	64x64	Adam	0.7865
Model 10	.001	256 x 256	Adam	0.8281

Accuracies when Classifying WWE vs. Boxing - 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.4897
Model 2	.0001	128x128	SGD	0.6183
Model 3	.001	256 x 256	SGD	0.6138
Model 4	.001	64 x 64	SGD	0.5804
Model 5	.001	64 x 64	Rmsprop	0.6460
Model 6	.001	256 x 256	Rmsprop	0.7397
Model 7	.001	128x128	Rmsprop	0.6143
Model 8	.0001	64x64	Adam	0.6205
Model 9	.0001	128x128	Adam	0.6607
Model 10	.001	256 x 256	Adam	0.6741

Swimming and Basketball

On our right here is the distribution between the two sports, and there are about 1,133 images in this set. We see that swimming has more images which may cause some problems when doing training models. As expected, our best results are comparing two completely different sports- basketball and swimming as the images have very little in common. The model sees accuracy in the high 90's which lets us know that our model is strong, which is encouraging for the task of comparing images of sports that are very similar. This result can help us understand also how the model processes the images, since both



sports are very different from each other most likely the model observes the position within the image and the model might detect images with or without basketball for this particular set. In this particular dataset we see that all three image sizes give high accuracy, so size of the image was not a main factor. What did make a difference in accuracy was the optimizer type- RMSprop and Adam gave the best result from both tables of results. Below are the top ten highest accuracies and top two highlighted for each split. The full results are shown below in **Tables 7 and 8.**

Accuracies when Classifying Basketball vs. Swimming - 80/10/10 Split				
	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	.001	128x128	SGD	0.8066
Model 2	.0001	128x128	SGD	0.7580
Model 3	.001	64x64	SGD	0.7760
Model 4	.001	256 x 256	SGD	0.7886
Model 5	.001	64 x 64	Rmsprop	0.9334
Model 6	.001	256 x 256	Rmsprop	0.9718
Model 7	.001	128x128	Rmsprop	0.9483
Model 8	.001	64x64	Adam	0.9664
Model 9	.001	128x128	Adam	0.9232
Model 10	.001	256 x 256	Adam	0.9640

Accuracies when Classifying Basketball vs. Swimming - 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 3	.001	128x128	SGD	0.9536
Model 4	.0001	128x128	SGD	0.9462
Model 7	.001	256 x 256	SGD	0.9384
Model 9	.001	64 x 64	SGD	0.9497
Model 10	.001	64 x 64	Rmsprop	0.9697
Model 11	.001	256 x 256	Rmsprop	0.9653
Model 12	.001	128x128	Rmsprop	0.9731
Model 13	.0001	64x64	Adam	0.9233
Model 14	.0001	128x128	Adam	0.9731
Model 15	.001	256 x 256	Adam	0.9311

Overall Dataset

Our overall dataset is relatively well-balanced, as we mentioned in the introduction. We wanted to see how well the model could identify all 22 categories so that we could better understand how well it was picking up on the four different splits we looked at. Overall results were not quite as good in terms of accuracy which is not surprising. We had issues with run times and GPU crashing since we tried to use the entire dataset and its thousands of images so that we

Accuracies on the entire Dataset 80/10/10 Split				
	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	.001	128x128	SGD	0.6110
Model 2	.0001	128x128	SGD	0.6944
Model 3	.0001	256 x 256	SGD	0.5977
Model 4	.0001	64 x 64	SGD	0.6116
Model 5	.0001	64 x 64	Rmsprop	0.7360
Model 6	.0001	256 x 256	Rmsprop	0.6393
Model 7	.0001	128x128	Rmsprop	0.6892
Model 8	.0001	64x64	Adam	0.7108
Model 9	.0001	128x128	Adam	0.6454
Model 10	.0001	256 x 256	Adam	0.7328

Accuracies on the entire Dataset 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	.001	128x128	SGD	0.4330
Model 2	.0001	128x128	SGD	0.6220
Model 3	.0001	256 x 256	SGD	0.4598
Model 4	.0001	64 x 64	SGD	0.4742
Model 5	.0001	64 x 64	Rmsprop	0.6671
Model 6	.0001	256 x 256	Rmsprop	0.7190
Model 7	.0001	128x128	Rmsprop	0.7092
Model 8	.0001	64x64	Adam	0.7112
Model 9	.0001	128x128	Adam	0.7015
Model 10	.0001	256 x 256	Adam	0.7320

could optimize results. We could still count on accuracies in the low-70's. **Tables 9 & 10** in the appendix feature the full extent of our analysis.

Conclusions

Overall in our results we see this trend that if the dataset has too many images, the model had a harder time in recognizing each sport, especially shown in the overall dataset. We see in the end if the model is given sports that are very different from each other, the model has a higher accuracy compared to those who are very similar- which was not a surprise. One of the few surprises is how well the models did with similar sports that did not have many positions and descriptive images. Learning rate did best in the lower values like 0.0001 compared to 0.001 or .01. The top optimizer type we kept seeing for each dataset was RMSprop and Adam. The input size 256 x 256 seems to yield the best results as well.

We faced a couple of difficulties throughout the project. One of the first issues was getting the model to run. We had issues getting the data to process the images in a way that would make logical predictions during training and testing. Getting our training, testing and validation sets to work was something that also took a long time due to the fact that we had issues with sizing and the fact that the dataset has a lot of pictures in it. The platform that we used was Google Colab and at times depending on the size of the dataset, it took too long to run even with or without the GPU setting. Occasionally the GPU would crash or time out, causing us to have numerous delays in our results. Therefore, with the number of datasets that we had, it was time consuming to do each separately.

Even though we set the seed for consistent results, we would get different accuracies that would not make sense, and then if we ran it again it would look more understandable. That caused the dataset to have inconsistency especially in the overall dataset. When dealing with the different splits we kept having very stark differences only until we ran the result again we would have similar numbers, but then the time would run out causing the code to crash and restart again. Loading the different datasets was tedious since everything was in one file and we had to be careful in terms of uploading the files on google drive since sometimes the system would fail to upload one picture which caused problems later in uploading the training, testing, and validation set. Even using Colab and google drive had it drawbacks. Though, it was the best in terms of sharing and collaborating for the both of us. We were able to share everything and keep track that way. Limitations we obviously had were run times, at times the datasets were not that big and each dataset had a different number of pictures which caused inconsistent results.

Improvements and further evaluations we wanted to approach and do next time are pre-processing the images to see if it improves the accuracy. We would also like to use a pre-trained model like resnet50 or something geared toward sports images. In certain datasets we would like to have used T-sne and also implemented cross validation within our model.

Future implementations in real life settings and applications that we wanted to use involving our model is maybe a mobile application to help identifying different sports with a

picture someone takes and maybe applying more features to it. Another application of these models would be sport analysis in helping identify not only the sport, but be able to tell different positions or throws for example. Promotion and advertisement companies can use these models to help identify sports images and transfer that to analysis of many images in a real world setting.

Works Cited

Kesorn, Kraisak, and Stefan Poslad. "Enhanced Sports Image Annotation and Retrieval Based Upon Semantic Analysis of Multimodal Cues." *Advances in Image and Video Technology*, Springer Berlin Heidelberg, pp. 817–28, doi:10.1007/978-3-540-92957-4_71.

Podgorelec, V., Pečnik, P., & Vrbančič, G. (2020). Classification of Similar Sports Images Using Convolutional Neural Network with Hyper-Parameter Optimization. *Applied Sciences*, *10*(23), 8494. https://doi.org/10.3390/app10238494

Rallet, Benedicte. "Sport Image Classification with Neural Networks." *Medium*, Jovian, 22 July 2020, https://medium.com/jovianml/sport-image-classification-with-neural-networks-16929b9f7932

Ranjan Rath, Sovit . (2020) *Sports Image Dataset [Kaggle]*. Version 1. Retrieved from https://www.kaggle.com/sovitrath/sports-image-dataset

Russo, M. A, Filonenko, A. and K. Jo, "Sports Classification in Sequential Frames Using CNN and RNN," 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT), Busan, Korea (South), 2018, pp. 1-3, doi: 10.1109/ICT-ROBOT.2018.8549884.

Zhang, X. (2021). Application of Convolution Network Model Based on Deep Learning in Sports Image Information Detection. *E3S Web of Conferences*, *233*, 02024. https://doi.org/10.1051/e3sconf/202123302024

Appendix

Table 1: Accuracies when Classifying Cricket vs Baseball- 80/10/10 Split

	Epochs (10,25,50)	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.5948
Model 2	25	.001	128x128	SGD	0.5854
Model 3	50	.001	128x128	SGD	0.6990
Model 4	50	.0001	128x128	SGD	0.7146
Model 5	50	.0001	64x64	SGD	0.7826
Model 6	50	.0001	256x256	SGD	0.4260
Model 7	50	.0001	256 x 256	Rmsprop	0.7198
Model 8	50	.0001	64x64	Rmsprop	0.7615
Model 9	50	.0001	128x128	Rmsprop	0.7198
Model 10	50	.0001	64x64	Adam	0.7719
Model 11	50	.0001	128x128	Adam	0.7302
Model 12	50	.0001	256 x 256	Adam	0.7250

Table 2: Accuracies when Classifying Cricket vs Baseball- 70/20/10 Split

	Epochs (10,25,50)	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.6156
Model 2	25	.001	128x128	SGD	0.6104
Model 3	50	.001	128x128	SGD	0.7188
Model 4	50	.0001	128x128	SGD	0.6625
Model 5	50	.001	256 x 256	SGD	0.7667
Model 6	50	.001	64 x 64	SGD	0.6323
Model 7	50	.001	64 x 64	Rmsprop	0.6375
Model 8	50	.001	256 x 256	Rmsprop	0.7875
Model 9	50	.001	128x128	Rmsprop	0.7198
Model 10	50	.001	64x64	Adam	0.6885
Model 11	50	.001	128x128	Adam	0.7094
Model 12	50	.001	256 x 256	Adam	0.8385

Table 3 :Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 80/10/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.5123
Model 2	25	.001	128x128	SGD	0.5241
Model 3	50	.001	128x128	SGD	0.6391
Model 4	50	.0001	128x128	SGD	0.6115
Model 5	50	.001	64x64	SGD	0.5207
Model 6	50	.001	256x256	SGD	0.4931
Model 7	50	.001	256 x 256	Rmsprop	0.6227
Model 8	50	.001	64x64	Rmsprop	0.6967
Model 9	50	.001	128x128	Rmsprop	0.6975
Model 10	50	.001	64x64	Adam	0.6620
Model 11	50	.001	128x128	Adam	0.6542
Model 12	50	.001	256 x 256	Adam	0.6789

Table 4: Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 70/20/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.3668
Model 2	25	.001	128x128	SGD	0.3668
Model 3	50	.001	128x128	SGD	0.4720
Model 4	50	.0001	128x128	SGD	0.6578
Model 5	50	.001	256 x 256	SGD	0.4945
Model 6	50	.0001	64 x 64	SGD	0.6713
Model 7	50	.0001	64 x 64	Rmsprop	0.6422
Model 8	50	.0001	256 x 256	Rmsprop	0.6421
Model 9	50	.0001	128x128	Rmsprop	0.6398
Model 10	50	.0001	64x64	Adam	0.6399
Model 11	50	.0001	128x128	Adam	0.6199
Model 12	50	.0001	256 x 256	Adam	0.6555

Table 5: Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 80/10/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.5123
Model 2	25	.001	128x128	SGD	0.5241
Model 3	50	.001	128x128	SGD	0.6391
Model 4	50	.0001	128x128	SGD	0.6115
Model 5	50	.001	64x64	SGD	0.5207
Model 6	50	.001	256x256	SGD	0.4931
Model 7	50	.001	256 x 256	Rmsprop	0.6227
Model 8	50	.001	64x64	Rmsprop	0.6967
Model 9	50	.001	128x128	Rmsprop	0.6975
Model 10	50	.001	64x64	Adam	0.6620
Model 11	50	.001	128x128	Adam	0.6542
Model 12	50	.001	256 x 256	Adam	0.6789

Table 6: Accuracies when Classifying Tennis vs. Table Tennis vs. Badminton - 70/20/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.3668
Model 2	25	.001	128x128	SGD	0.3668
Model 3	50	.001	128x128	SGD	0.4720
Model 4	50	.0001	128x128	SGD	0.6578
Model 5	50	.0001	256 x 256	SGD	0.4767
Model 6	50	.0001	64 x 64	SGD	0.6713
Model 7	50	.0001	64 x 64	Rmsprop	0.6422
Model 8	50	.0001	256 x 256	Rmsprop	0.6421
Model 9	50	.0001	128x128	Rmsprop	0.6398
Model 10	50	.0001	64x64	Adam	0.6399
Model 1`	50	.0001	128x128	Adam	0.6199
Model 12	50	.0001	256 x 256	Adam	0.6555

Table 7: Accuracies when Classifying Basketball vs. Swimming - 80/10/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.6979
Model 2	25	.001	128x128	SGD	0.7471
Model 3	50	.001	128x128	SGD	0.8066
Model 4	50	.0001	128x128	SGD	0.7580
Model 5	50	.001	64x64	SGD	0.7760
Model 6	50	.001	256 x 256	SGD	0.7886
Model 7	50	.001	64 x 64	Rmsprop	0.9334
Model 8	50	.001	256 x 256	Rmsprop	0.9718
Model 9	50	.001	128x128	Rmsprop	0.9483
Model 10	50	.001	64x64	Adam	0.9664
Model 11	50	.001	128x128	Adam	0.9232
Model 12	50	.001	256 x 256	Adam	0.9640

Table 8: Accuracies when Classifying Basketball vs. Swimming - 70/20/10 Split					
	Epochs (10, 25, 50)	Learning Rate (.01, .001, .0001)	Image input size	Optimizer type (SGD, rmsprop, adam)	Validation Accuracy
Model 1	10	.001	128x128	SGD	0.5958
Model 2	25	.001	128x128	SGD	0.9115
Model 3	50	.001	128x128	SGD	0.9536
Model 4	50	.0001	128x128	SGD	0.9462
Model 5	50	.001	256 x 256	SGD	0.9384
Model 6	50	.001	64 x 64	SGD	0.9497
Model 7	50	.001	64 x 64	Rmsprop	0.9697
Model 8	50	.001	256 x 256	Rmsprop	0.9653
Model 9	50	.001	128x128	Rmsprop	0.9731
Model 10	50	.0001	64x64	Adam	0.9233
Model 11	50	.0001	128x128	Adam	0.9731
Model 12	50	.001	256 x 256	Adam	0.9311

Table 9: Accuracies on the entire Dataset 80/10/10 Split				
	Learning Rate (.01, .001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.6110
Model 2	.0001	128x128	SGD	0.6944
Model 3	.0001	256 x 256	SGD	0.5977
Model 4	.0001	64 x 64	SGD	0.6116
Model 5	.0001	64 x 64	Rmsprop	0.7360
Model 6	.0001	256 x 256	Rmsprop	0.6393
Model 7	.0001	128x128	Rmsprop	0.6892
Model 8	.0001	64x64	Adam	0.7108
Model 9	.0001	128x128	Adam	0.6454
Model 10	.0001	256 x 256	Adam	0.7328

Table 10: Accuracies on the entire Dataset 70/20/10 Split				
	Learning Rate (.01, .001, .0001)	**Image input size**	**Optimizer type (SGD, rmsprop, adam)**	**Validation Accuracy**
Model 1	.001	128x128	SGD	0.4330
Model 2	.0001	128x128	SGD	0.6220
Model 3	.0001	256 x 256	SGD	0.4598
Model 4	.0001	64 x 64	SGD	0.4742
Model 5	.0001	64 x 64	Rmsprop	0.6671
Model 6	.0001	256 x 256	Rmsprop	0.7190
Model 7	.0001	128x128	Rmsprop	0.7092
Model 8	.0001	64x64	Adam	0.7112
Model 9	.0001	128x128	Adam	0.7015
Model 10	.0001	256 x 256	Adam	0.7320