Sports Products Multi-Class Classification

Ahmed Abdo Zaid
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
Cairo, Egypt
azad011@uOttawa.ca

Hosam mahmoud ibrahim
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
Cairo, Egypt
hmahm074@uottawa.ca

Abdulrahman Ahmed
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
Cairo, Egypt
aahme275@outtawa.ca

Eslam Abdelraheem Shabaan Khalaf
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
Cairo, Egypt
ekhal066@uottawa.ca

***Abstract*—The usage of several computational intelligence approaches in sports is growing. Our project's goal is to categorize 11 different sports, or their equipment based on their visual representations. This model may be used to classify sports-related photos. Preprocessing and feature extraction are required because there are many identical features in this image. As a result, we work to create a model for training the conventional models that rely on elements like colors, textures, and item shapes that are created by hand. As a result, we must create a model that can distinguish between the various categories. We work to create deep learning (DL) approaches and methods that employ CNNs. The classification of sports images requires additional work to prepare the data we collect from a variety of sources, train, test, and validate (480 images for training, 40 for each class), (55 images for testing, 5 for each class), (55 images for validation, 5 for each class), observe the accuracy, collect additional images to train the model on, and adjust for hyperparameters. convolutional neural networks are used to automatically extract a deep representation of training data for image classification and produce results that are close to human performance. We will first construct a conventional ML model and evaluate accuracy; subsequently, we will construct a CNN model and contrast them to increase accuracy. we applied three models the accuracy score for the first, Alex-net, was 60%, and the second, VGG, had an accuracy score of 87%. Finally, we selected the Res-Net model as the winner. Its performance was 89 percent.

Keywords*— *ResNet, VGGModel, AlexNet.

I. Introduction

Society's interest in sports is expanding, and there are many sports available now that many people are unaware of. Our research will use convolutional neural networks (CNNs) and several deep learning approaches and pre-trend models to perform multi-class classification for 11 different types of sports or their equipment, such as a ball, baseball bat, baseball gloves, bowling bin, dumbbell, boxing gloves, football helmets, racket, sports shoes, t-shirt, and whistle. We believe that in addition to increasing the training data, the size of the validation and test sets should also be raised. Our champion model was chosen from a group of three proposed models based on its accuracy, recall, f1-score, and loss curves. The accuracy score for the first, Alex-net, was 60%, and the second, VGG, had an accuracy score of 87%. Finally, we selected the Res-Net model as the winner. Its performance was 89 percent, placing it first.

II. Related Work

This section includes a brief overview of earlier research that dealt with the problem of classifying images for sports-related information, including both conventional machine learning methods and created deep learning techniques. In the first instance, a preliminary step must be taken to gather the pertinent features from the photos that will later serve as the input for the machine learning technique. In the second case, on the other hand, both the features and the classifier are automatically extracted from the learning process.

"Robust Sports Image Classification Using InceptionV3 and Neural Networks" presents a framework for the classification of various categories of sports using Random Forest, K-Nearest Neighbors, and Support Vector Machine (SVM). This framework has achieved an average accuracy of 96.64%. Another robust framework using neural networks and InceptionV3 has achieved the best accuracy as compared to other classifiers due to its ability to handle large datasets efficiently. These accuracies can also be increased as there is scope for the rectification of images; some images are blurry and dark, which can be preprocessed. This framework has successfully achieved an average accuracy of 96.64% over six categories, which demonstrates the effectiveness of the framework and can be used for the detection and classification of various sports activities.

"Going Deeper with Contextual CNN for Hyper-spectral Image Classification" used a deep convolution neural network (DCNN) unlike existing simple CNN, it can optimal explore local conceptual intersections by the help of the relationship of the neighboring pixels, the proposed architecture tested on three benchmarks datasets: the Indian Pines dataset (145X145 image size), the Salinas dataset (512X217 image size) and the university of Pavia dataset (610X340 image size).

In "Deep Image Classification of a Wild Data Set for Olympic Sports" (2016), it shows how to increase the success of image classification by using a deep learning technique. It used "GoogleLetNet," which is the topology for the convolutional neural networks (CNN), to train 26 of the Olympic sports and 5362 images. It used "GoogleLetNet", which is the topology for the Convolutional Neural Networks (CNN), to train 26 of the Olympic sports and 5362 images. It can reduce the number of parameters in the network and replace the fully connected layers with sparse ones, so the decision of classification was spread through this network. The authors collected these images from the web using "Google’s Custom Search Engine tool." They split these images into three parts (training, validation, and testing) as 3763, 802, and 797. They also used data augmentation to increase the dataset that became 22522 images. To solve the imbalanced dataset, they used oversampling. So, they categorized some sports easily. After using the augmented, it did better in 7 classes while doing worse in 12 others. After using the balanced, it did better in 12 classes while doing worse in 12 others. Therefore, after using data augmentation, the best basketball forecast was 0.4963. The best handball forecast, using balanced data, was 0.999. Due to a conflict between two classes with similar probability, it therefore projected both very good results in some circumstances and bad results in others.

III. DATA MANIPULATION

i. Data Collection & Segmentation

The sports products dataset is made up of images from various online websites. The size of this dataset is 550 images that were segmented into 11 classes, as shown in Figure III (1). 11 classes were split into training, validation, and testing data sets of 440, 55, and 55 images, respectively.

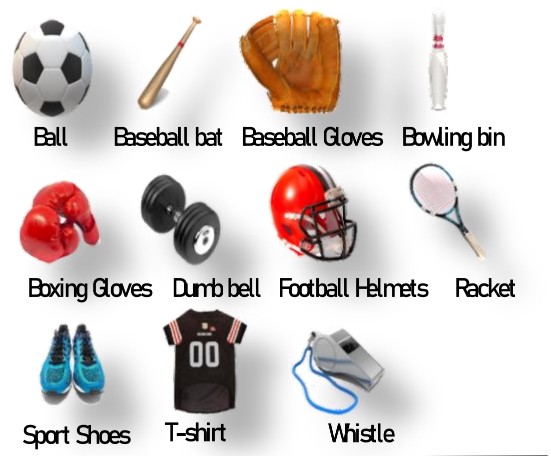


Figure III (1): Sports products dataset

These images were resized to 224 x 224 pixels as shown in Figure III (2) that were shuffled.

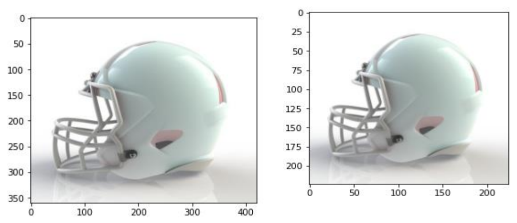


Figure III (2): resize 224 x 224 pixels.

ii. Data Augmentation

By using data augmentation on these photos, it is possible to increase the size of the train set by modifying the original data through rotation (by 90 degrees), shearing (by 0.5), zooming (by 0.2), and horizontal flipping as shown in Figure III (4). Making the model learn on all sides of the images will make learning and predicting easier and prevent overfitting.

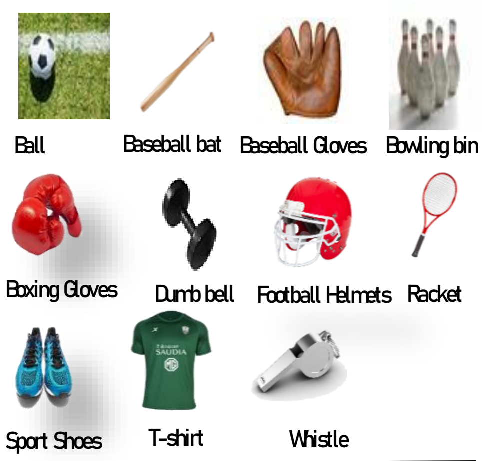


Figure III (3): Before applying Data Augmentation for Sports products

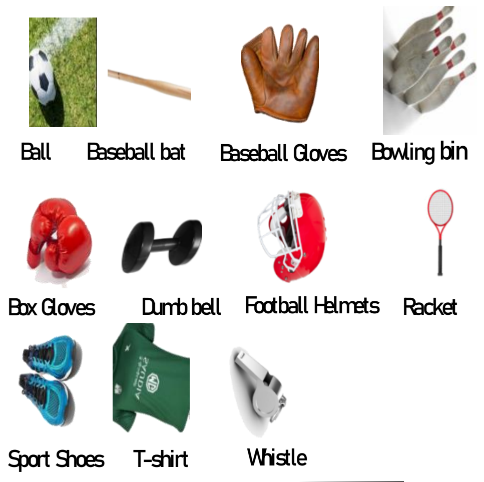


Figure III (4): After applying Data Augmentation for Sports products

IV. METHODS

i. The AlexNet CNN architecture

It is a convolutional neural network that has 8 layers deep, was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, this architecture won the 2012

ImageNet ILSVRC challenge with a 17% top-5 error rate while the second-best achieved 26% top-5 error rate, AlexNet network was trained using over a million images from the ImageNet dataset with 1000 different classes As a result of learning rich feature representations for a diverse set of images, AlexNet pertained model can easily classify images into 1000 object categories.

ii. *VGG16 model*

The VGG deep convolutional neural network was designed by Andrew Zisserman and Karen Simonyan. Visual Geometry Group, situated at Oxford University, is abbreviated as VGG. This model finished second in the ILSVRC-2014 competition with a classification performance of 92.7%. In order to deal with large-scale pictures, the VGG model employs a relatively modest convolutional filter size of 3 * 3 to assess the depth of layers. The authors offered a set of VGG models ranging in layer length from 11 to 19.

iii. *ResNet152 Model*

ResNet152 architecture is one of best pertained models out there, it was released in 2015 by Microsoft Research Asia and it have achieved very successful results in the ImageNet and MS-COCO competitions. The central idea behind these models, residual connections, has been shown to significantly improve gradient flow, allowing for the training of much deeper models with tens or even hundreds of layers.

ResNet152, VGG16 are trained on more than 14 million images with 1000 different classes, and we have found that among these classes there are a lot of sports objects in this dataset like: basketball, soccer ball, tennis ball, golf ball, football helmet, baseball, volleyball, jersey, T-shirt, dumbbell, running shoe, whistle, and racket.

iv. *Architecture Tuning*

For the Hyperparameters tuning we have some options such as:

a) Trying experiment with several loss functions.
b) Trying different activation functions.
c) Trying different layers with different Neurons.
d) Trying different optimizers.
e) Trying different batch sizes and different epochs.

For the Freezing /without freezing we have some options such as:

a) Trying to add more layers while freezing the architecture layers.
b) Trying to add our own layers while unfreezing the architecture layers.
c) Trying to unfreeze some of the architecture layers, freeze some, and add our own layers.

v. *Champion model*

We have found that the tuned ResNet152 model gave us 89% accuracy on out test dataset.

vi. *Evaluation/Visualization*

we have tested our different models as it will be discussed In Experiments section and also, we have done T-SNE visualization to our champion model to be able to see what our champion misclassified and why.

vii. *Final Test & Deployment*

we have saved our champion model as an h5 file, after that we have created a web application using flask API, and now we are able to test our champion model live with real sports images from internet, we did that and it performed well.

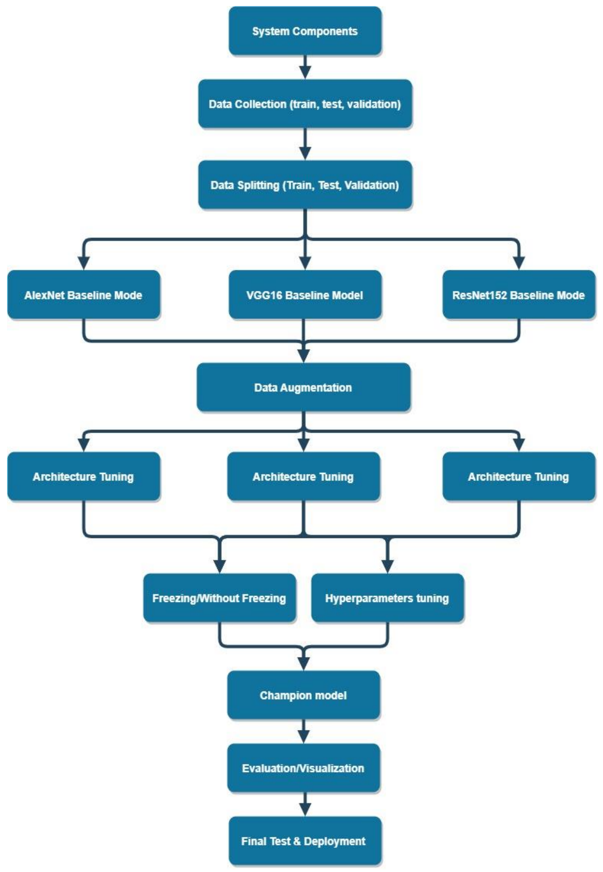


Figure IV (1): System Components

V. EXPERIMENTS

i. *Alex-net Model*

a) Alex-net -Baseline Model

We started by implementing the Alex-net baseline model. We Wrote Alex-net Implementation then We added our soft-max output layer. The model was then trained across 100 epochs using a batch size of 512, Adam as the optimizer, and sparse categorical cross-entropy as the loss and the data for the train is relatively small in order to learn the model from it, so it was 440 images for each class 40 images, so I made a data augmentation on the training, so the size of the image for the training increased and became 1080 for the training. I noticed a gap between training and validation from the beginning. The model's performance on testing data was 69 percent, which was poor accuracy due to a lack of training data from which learn.

```
              precision    recall  f1-score   support

           0       0.33      0.20      0.25         5
           1       1.00      1.00      1.00         5
           2       0.83      1.00      0.91         5
           3       0.50      0.80      0.62         5
           4       1.00      0.20      0.33         5
           5       0.83      1.00      0.91         5
           6       0.50      0.20      0.29         5
           7       1.00      1.00      1.00         5
           8       0.56      1.00      0.71         5
           9       0.80      0.80      0.80         5
          10       0.40      0.40      0.40         5

    accuracy                           0.69        55
   macro avg       0.71      0.69      0.66        55
weighted avg       0.71      0.69      0.66        55

Text(69.0, 0.5, 'True classes')
```

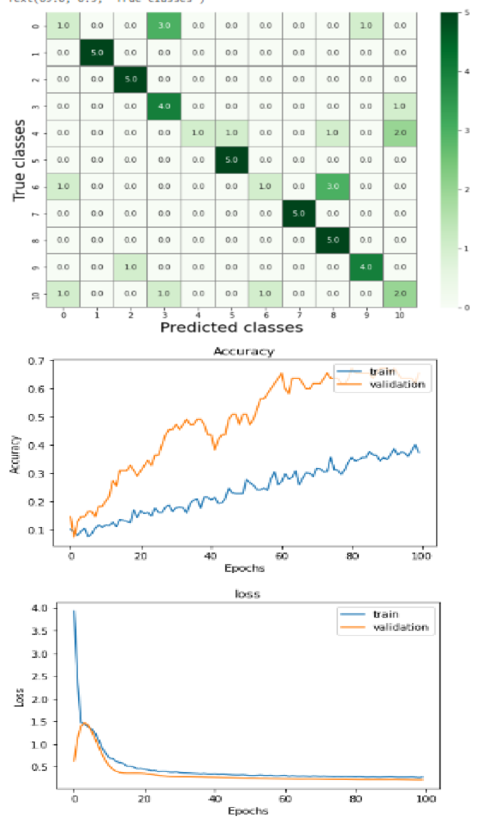


Figure V (1): Evolution of Alex-net model

```
              precision    recall  f1-score   support

           0       1.00      0.80      0.89         5
           1       0.57      0.80      0.67         5
           2       1.00      0.80      0.89         5
           3       1.00      0.60      0.75         5
           4       0.50      0.60      0.55         5
           5       0.56      1.00      0.71         5
           6       1.00      0.40      0.57         5
           7       1.00      0.40      0.57         5
           8       0.50      0.80      0.62         5
           9       0.80      0.80      0.80         5
          10       1.00      1.00      1.00         5

    accuracy                           0.73        55
   macro avg       0.81      0.73      0.73        55
weighted avg       0.81      0.73      0.73        55

Text(69.0, 0.5, 'True classes')
```

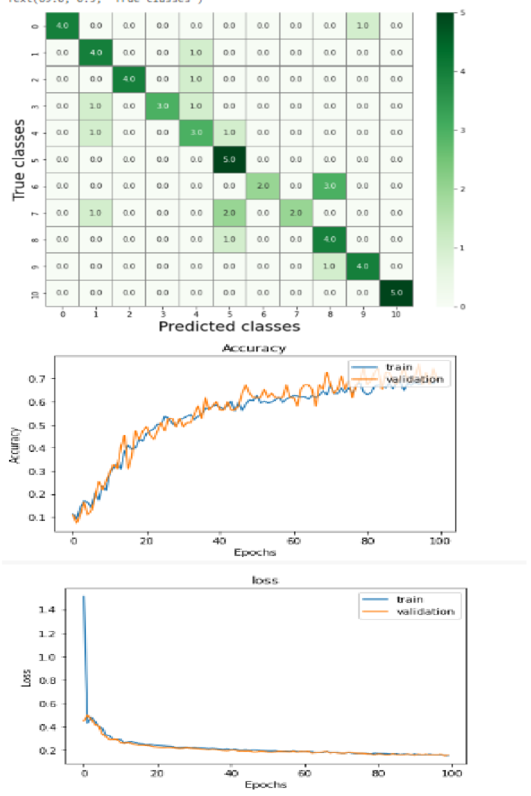


Figure V (2): Evolution of **Alex-net -enhanced Model**

The baseline result was poor, so we added some layers to our model and remove drop-out

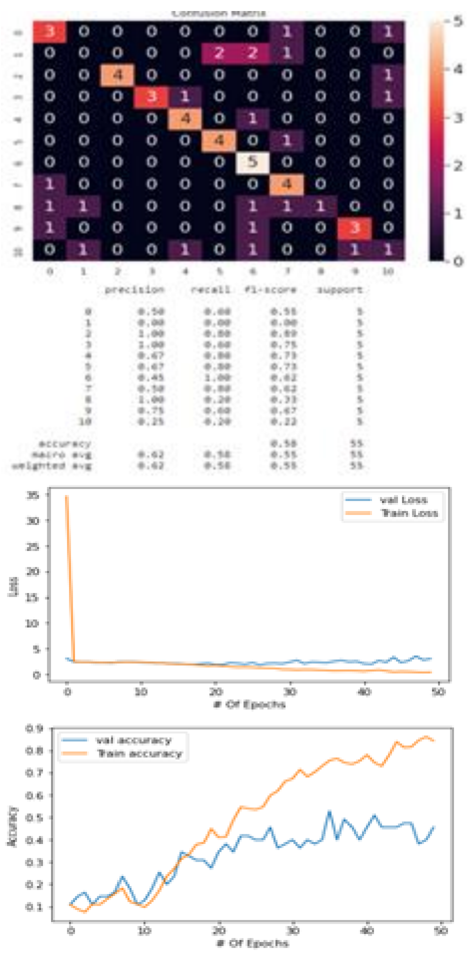
b) Alex-net -enhanced Model

We first added two layers with 2048 and 1024 neurons Relu proved to be the most effective among the activation functions we tried, including leaky-relu, and tanh. We also tried sgd as an optimizer, but somehow it performed poorly, so We used Adam as my optimizer, and sparse categorical crossentropy as my loss function with a batch size of 512 and 100 epochs the accuracy of the model on the test dataset was 73 percent and the under-fitting problem was solved.

ii. VGG16 model

a) VGG-Baseline model

We started by implementing the VGG16 baseline model. We unfroze the VGG layer's layers and introduced our own input, as well as two levels with 128 and 64 neurons with relu activation functions, before adding our output layer. The model was then trained across 100 epochs using a batch size of 64, Adam as the optimizer, and sparse categorical cross-entropy as the loss. We noticed a gap between training and validation at 30 epochs, and in loss, this will lead to an overfitting issue. The model's performance on testing data was 58 percent, which was poor accuracy due to a lack of training data from which learn.



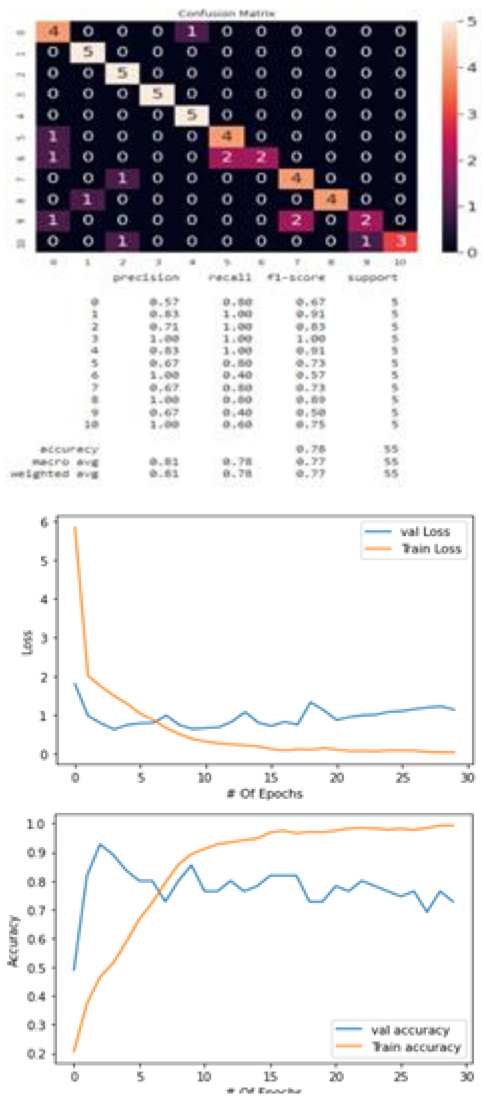
```
              precision    recall  f1-score   support

           0       0.50      0.60      0.55         5
           1       0.00      0.00      0.00         5
           2       1.00      0.80      0.89         5
           3       1.00      0.60      0.75         5
           4       0.67      0.80      0.73         5
           5       0.67      0.80      0.73         5
           6       0.45      1.00      0.62         5
           7       0.50      0.80      0.62         5
           8       1.00      0.20      0.33         5
           9       0.75      0.60      0.67         5
          10       0.25      0.20      0.22         5

    accuracy                           0.58        55
   macro avg       0.62      0.58      0.55        55
weighted avg       0.62      0.58      0.55        55
```

Figure V (3): Evolution of VGG-Baseline model

The baseline result was poor because the data for the train is relatively small in order to learn the model from it, so it was 440 images for each class 40 images, so I made a data augmentation on the training, so the size of the image for the training increased and became 1080 for the training, and this is an acceptable percentage to some extent in order to learn the model from it and get to know on the different classes with less errors and higher accuracy. So, we tuned another two different vgg models after data augmentation.

b) First VGG-tuned and Hyperparameter Optimization

We first added three layers with 2048, 1024, and 512 neurons each after freezing the vgg16 layers. Relu proved to be the most effective among the activation functions we tried, including leaky relu, and tanh. We also tried sgd as an optimizer, but somehow it performed poorly, so I used Adam as my optimizer, and sparse categorical crossentropy as my loss function with a batch size of 64 and 30 epochs. After the tenth epoch, the training loss keeps getting worse as the number of epochs goes up, the training accuracy goes up until it reaches its highest value, which is 100%, while the training loss keeps getting smaller. and this case is known as an overfitting problem. The accuracy of the model on the test dataset was 78 percent, and the model misclassified some samples from the classes zero, six, and nine, and their names as follows: "ball," "football helmet," and "sports t-shirt".

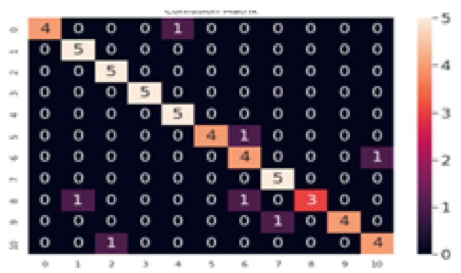


```
              precision    recall  f1-score   support

           0       0.57      0.80      0.67         5
           1       0.83      1.00      0.91         5
           2       0.71      1.00      0.83         5
           3       1.00      1.00      1.00         5
           4       0.83      1.00      0.91         5
           5       0.67      0.80      0.73         5
           6       1.00      0.40      0.57         5
           7       0.67      0.80      0.73         5
           8       1.00      0.80      0.89         5
           9       0.67      0.40      0.50         5
          10       1.00      0.60      0.75         5

    accuracy                           0.78        55
   macro avg       0.81      0.78      0.77        55
weighted avg       0.81      0.78      0.77        55
```

Figure V (4): Evolution of First VGG-tuned and Hyperparameter Optimization.

c) Second Tuned VGG Model

We noticed in the previous experiment that the model's performance was good regardless the overfitting problem, we tried to use the same architecture of the previous experiment, but we modified it by adding some batch normalization and dropout layers to prevent the model from overfitting, The model learned in a better way, regardless there was some variation in the validation accuracy, we also noticed that the validation loss was not larger that much than the training loss, the gap between training and validation accuracy was also relatively small, This new architecture successfully eliminate the overfitting problem and the test accuracy of the model was 87 percent.



```
              precision    recall  f1-score   support

           0       1.00      0.80      0.89         5
           1       0.83      1.00      0.91         5
           2       0.83      1.00      0.91         5
           3       1.00      1.00      1.00         5
           4       0.83      1.00      0.91         5
           5       1.00      0.80      0.89         5
           6       0.67      0.80      0.73         5
           7       0.83      1.00      0.91         5
           8       1.00      0.60      0.75         5
           9       1.00      0.80      0.89         5
          10       0.80      0.80      0.80         5

    accuracy                           0.87        55
   macro avg       0.89      0.87      0.87        55
weighted avg       0.89      0.87      0.87        55
```

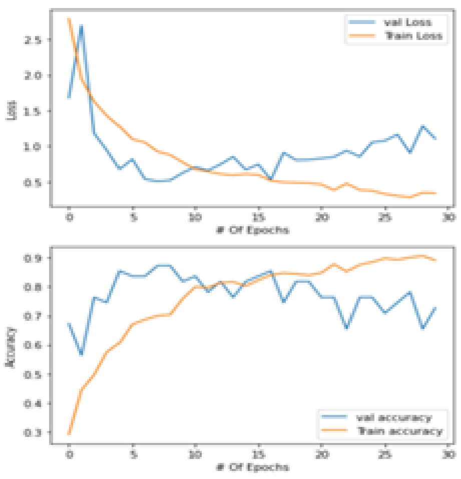
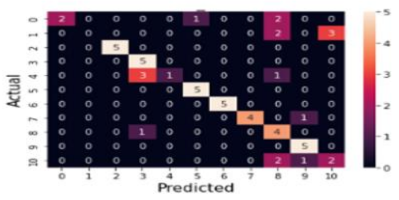


Figure V (5): Evolution of Second Tuned VGG Model

iii. ResNet152 Model

a) ResNet152 Baseline Model

We have tried this baseline model before applying the data augmentation step, We have unfrozen the all the layers to be able to update all the weights of the architecture and see what this model will do, and I only added 2 Dense layers (128, 64) and an output layer (softmax) above the ResNet layers with pooling parameter “max”, with epoch size = 30 and batch size = 64, the ResNet152 Baseline model gave us an accuracy of 69% on the test accuracy.



```
              precision    recall  f1-score   support

           0       1.00      0.40      0.57         5
           1       0.00      0.00      0.00         5
           2       1.00      1.00      1.00         5
           3       0.56      1.00      0.71         5
           4       1.00      0.20      0.33         5
           5       0.83      1.00      0.91         5
           6       1.00      1.00      1.00         5
           7       1.00      0.80      0.89         5
           8       0.36      0.80      0.50         5
           9       0.71      1.00      0.83         5
          10       0.40      0.40      0.40         5

    accuracy                           0.69        55
   macro avg       0.72      0.69      0.65        55
weighted avg       0.72      0.69      0.65        55
```

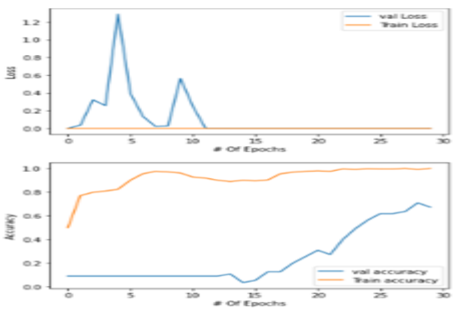
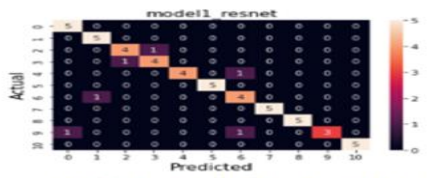


Figure V (6): Evolution of ResNet152 Baseline Model

b) ResNet152 Second Model

After the baseline model we have applied the data augmentation step on the training data, and then we have tried different options in terms of model architecture and its parameters.

We have tried different optimizers (“SGD”, “Adam”, and “Adamax”) and “Adam” gave me the highest test accuracy. We have tried different values (“avg”, “max”) for the parameter pooling in ResNet architecture and “avg” gave me the highest test accuracy. We have tried different dense layers with different numbers of neurons and the architecture below gave me the highest test accuracy.



```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       0.83      1.00      0.91         5
           2       0.80      0.80      0.80         5
           3       0.80      0.80      0.80         5
           4       1.00      0.80      0.89         5
           5       1.00      1.00      1.00         5
           6       0.67      0.80      0.73         5
           7       1.00      1.00      1.00         5
           8       1.00      1.00      1.00         5
           9       1.00      0.60      0.75         5
          10       1.00      1.00      1.00         5

    accuracy                           0.89        55
   macro avg       0.90      0.89      0.89        55
weighted avg       0.90      0.89      0.89        55
```

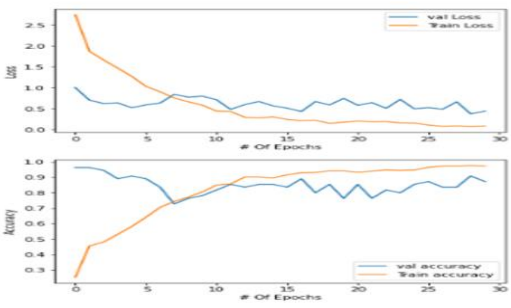
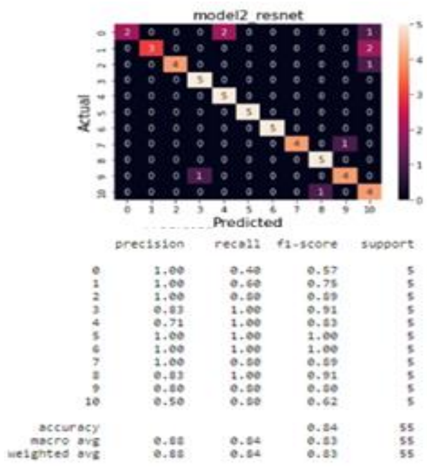


Figure V (7): Evolution of ResNet152 Second Model

c) ResNet152 Third Model

Here we were trying to reduce the gap between the training accuracy and the validation accuracy, so We added some Dropout and Batch normalization layers.



```
              precision    recall  f1-score   support

           0       1.00      0.40      0.57         5
           1       1.00      0.60      0.75         5
           2       1.00      0.80      0.89         5
           3       0.83      1.00      0.91         5
           4       0.71      1.00      0.83         5
           5       1.00      1.00      1.00         5
           6       1.00      1.00      1.00         5
           7       1.00      0.80      0.89         5
           8       0.83      1.00      0.91         5
           9       0.80      0.80      0.80         5
          10       0.50      0.80      0.62         5

    accuracy                           0.84        55
   macro avg       0.88      0.84      0.83        55
weighted avg       0.88      0.84      0.83        55
```

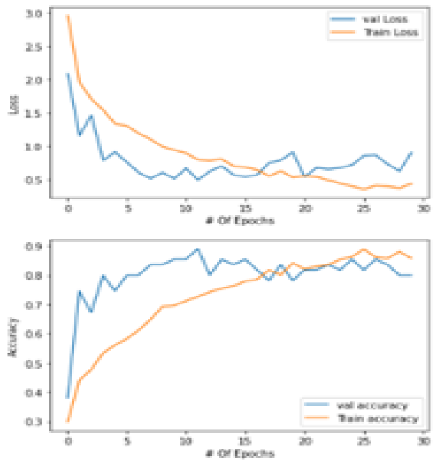


Figure V (8): Evolution of ResNet152 Third Model

After we got our champion model we have plotted a T-SNE Visualization to evaluate the performance of the champion model and to know which classes that the model miss classified.

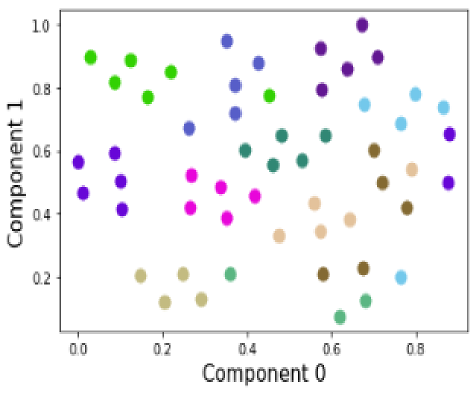


Figure V (9): T-SNE of ResNet152 Third Model

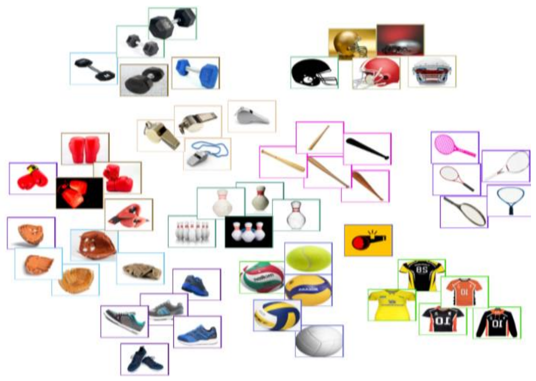


Figure V (10): T-SNE Visualization of ResNet152 Third Model

iv. *Final Test & Deployment*

We have test out champion model by using real images from the internet (t-shirt with soccer paint on it) and the model correctly classified it.

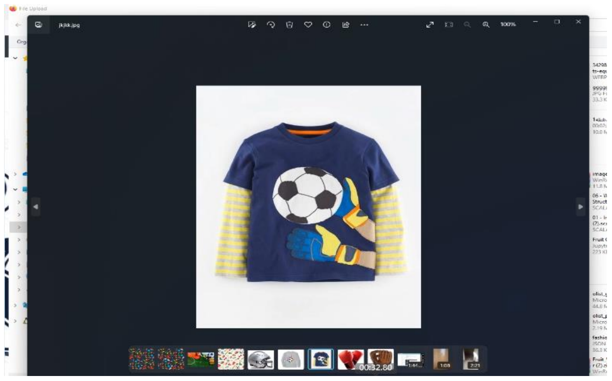
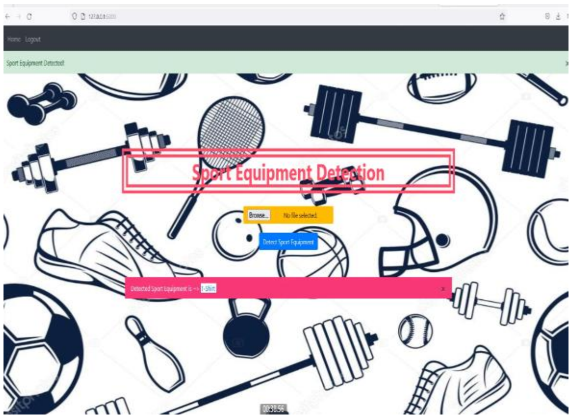


Figure V (11): Tricky image to test the model.



Sport Equipment Detected!

Detected Sport Equipment is --> T-Shirt

Figure V (12): Champion **Model Correct Perdition (T-Shirt).**

VI. CONCLUSION

we found that there are a lot of options to play with and different things to try in the transfer learning architectures, after we got our champion model which is ResNet152, we conclude that this is reasonable thing because the ResNet152 architecture is a big and complex architecture that contains different layers, also we learnt a lot of new stuff in this project like deployment techniques and we learned also different visualization and tuning techniques, the table in below show the different models test accuracies that we have obtained.

	Base-Model	First-Model	Second-Model
AlexNet	69%	73%	-
VGG16	50%	78%	87%
ResNet152	69%	89%	84%

Figure VI (1): The accuracies of all Models

VII. FUTURE WORK

we considering to transform this single label classification problem into multiple label classification problem, to let our model to predict different sports products in a single image with a confidence score for each product in that image.

VIII. REFERENCES

[1] Joshi, K., Tripathi, V., Bose, C., & Bhardwaj, C. (2020). Robust sports image classification using InceptionV3 and neural networks. Procedia Computer Science, 167, 2374-2381.

[2] Lee, H., & Kwon, H. (2017). Going deeper with contextual CNN for hyperspectral image classification. IEEE Transactions on Image Processing, 26(10), 4843-4855.

[3] Deep Image Classification of a Wild Data Set for Olympic Sports, Daniel Ferreira Moreira, C. Vasconcelos, A. Paes, Luis Velho less, Published 2016
link:https://www.addlabs.uff.br/workpedia2016/documentos/Artigo_08.pdf

[4] IMAGENET 1000 class list (no date) IMAGENET 1000 Class List - WekaDeeplearning4j. Available at: https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/ (Accessed: December 1, 2022).

[5] (no date) ResNet-152 - Wolfram Neural Net Repository. Available at: https://resources.wolframcloud.com/NeuralNetRepository/resources/ResNet-152-Trained-on-ImageNet-Competition-Data/ (Accessed: December 1, 2022).

[6] (no date) ImageNet. Available at: https://image-net.org/ (Accessed: December 1, 2022).

[7] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

[8] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015b). Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567