Sports Image Classification Using EfficientNet

Srilekha Somanchi, Yaswanth Phani Kommineni, Akshay Praveen Nair
Department of Computer Science
Amrita Vishwa Vidyapeetham,
Amritapuri, India

***Abstract*—Recognizing the sport played in the image from the pose and surroundings is easy for humans but might not be for the machines given that there are vast number of sports played around the world. In this paper, we illustrate the use of various EfficientNet models along with PyTorch framework for a strange version of image classification. Python is used as a programming language because it comes together with Pytorch framework. Input data mainly focuses on around 11,000 images categorized into 73 [9] classes based on the sport played. Sport classification done using improved EfficientNet with inverted bottleneck architecture of MobileNet-V2 [5],[7] as a backbone is a robust and efficient architecture for multi-sport classification that achieves state-of-the-art results. The results that we obtained are pretty good in comparison to the existing pretrained ConvNet architectures [1] with a classification rate of 98.35%.**

I. Introduction

Sport classification is an important task. Extensive data is available on sports on a day-to-day basis via various broadcasting medium like television, social media in the form of images and videos. There are thousands to millions of new images on sports that are available everyday on the internet. Classifying these images into their respective category helps many people in many ways. A layman can identify which sport that he is watching easily. A person who wants to search for a particular sport can get his job done easily if all the sports are segregated well. This problem of classifying images of sports into respective categories can be done using Neural Networks. Convolutional Neural Networks also known as (ConvNets) [7] are developed at a fixed resource budget and further scaled up for better accuracy if more resources are available. Scaling up of ConvNets is widely used to improve accuracy of many neural networks. To go even further, we use neural architecture search to design a new baseline network and scale it up to obtain a family of models, called EfficientNets, which achieve much better accuracy and efficiency than previous ConvNets. Hence, we used EfficientNet in our problem to classify 73 different sports images into their respective classes.

II. Related Work

Few people have worked on the similar problem and published papers [1],[2],[3],[4]. One such paper is "Robust Sports Image Classification Using InceptionV3 and Neural Networks" by Ketan Joshi, Vikas Tripathi, Chitransh Bose and Chaitanya Bhardwaj [1]. They've used Inception-V3 CNN model to classify the sport images based on the environment and related surroundings. They've obtained an accuracy of 96.64% with a dataset containing only six sports/classes. They've also compared the CNN (Inception-V3) model with classical Machine Learning algorithms like Random Forest, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM).

The second paper we will be talking about in this section is "Analysis of sports image detection technology based on machine learning" by Wenrui Yang [2]. In this paper she used sports video as a source to study the application of sports image detection models. Image detection technology edge detection, gray-scale preprocessing, object capture, target recognition, etc. are mixed to achieve various needs for sports image detection in her study. She also did realized the recognition of athletes, motion recognition and sports behaviour judgement.

III. Methodology

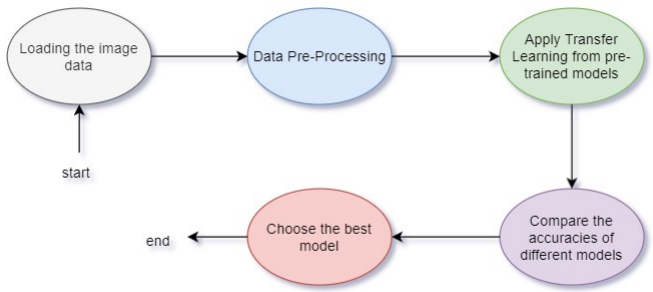

Fig. 1: Block Diagram

Here, a framework is created using pre-trained weights of EfficientNet architecture. This network fulfills 2 main objectives of optimizing the accuracy and FLOPS (floating point operations per second) [7].

We have finalised the models **Efficientnet-B4** and **Efficientnet-B6**. Various hyper parameters are tuned which include the number of epochs, batch size of gradient descent algorithm, gradient descent method, learning rates of gradient descent algorithm and different EfficientNet models. All the above parameters were tuned in the best way possible for the model that gave a maximum validation accuracy of 95% and hence the parameters were used that way. For further more details about hyper parameter tuning, please have a look at the subsection *Summary of Hyper Parameter tuning* in the section.

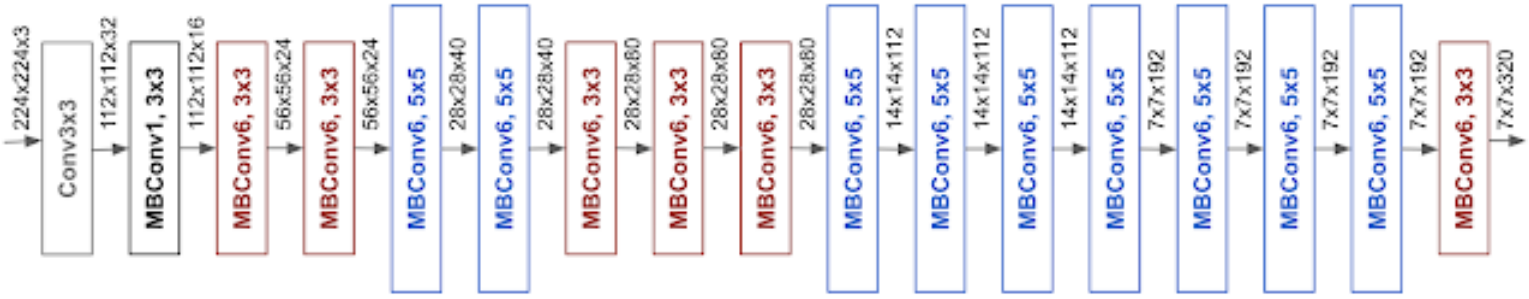

Fig. 2: The network architecture of EfficientNet

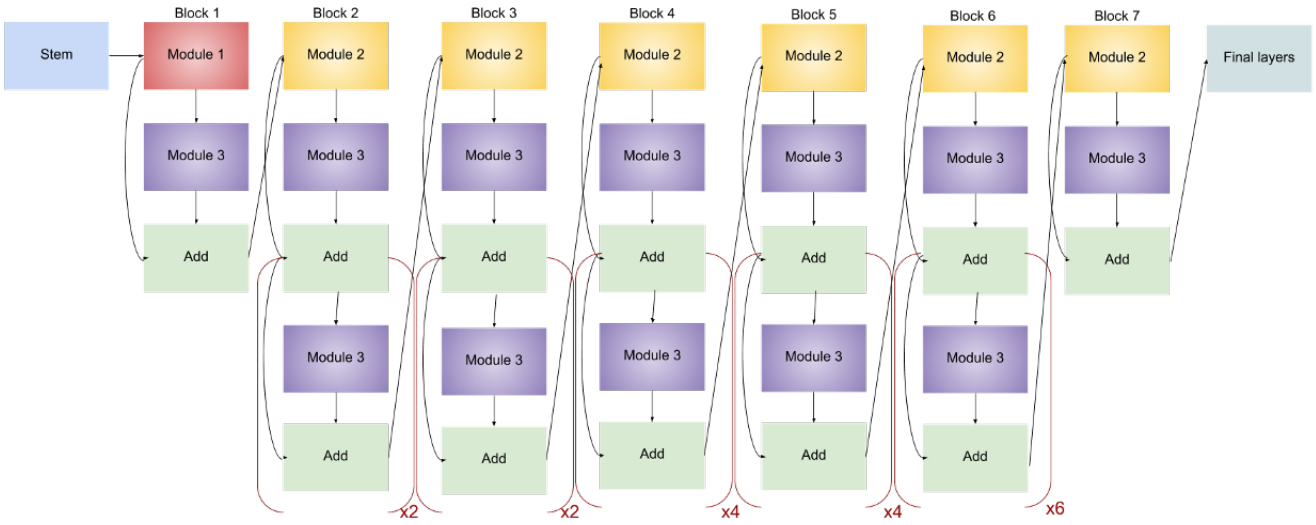

Fig. 3: Architecture of EfficientNet-B4

A. *EfficientNet Architecture*

EfficientNet is a ConvNet architecture. It is scaling method uniformly scales all dimensions of depth/width/resolution using a compound coefficient. It uses a compound coefficient that uniformly scales network width, depth, and resolution in a structured way. The compound scaling strategy is supported by the instinct that assuming the image is larger, the architecture needs more layers to build the open field and more channels to catch all the fine-grained designs on the larger image. The base EfficientNet-B0 architecture is based on the inverted bottleneck residual blocks of MobileNet-V2 along with squeeze-and-excitation blocks [7]. The architecture is built on a standard base using which all the experimenting with the architectures is done on all the eight models and final layers [5].

After this every one of them contains 7 modules. These modules further have a fluctuating number of sub-modules whose number is expanded as we move from EfficientNet-B0 to EfficientNet-B7. EfficientNet-B0 has a sum of 237 layers and in EfficientNet-B7 comes out to an aggregate of 813 layers. This large number of layers can be produced using 5 modules. Module 1 — This is utilized as a beginning stage for the sub-blocks. Module 2 — This is utilized as a beginning stage for the principal sub-module of the multitude of 7 primary modules aside from the first one. Module 3 — This is associated as a skip association with every one of the sub-blocks. Module 4 — This is utilized for consolidating the skip association in the primary sub-blocks. Module 5 — Each sub-block is associated with its past sub-block in a skip association and they are joined utilizing this module. These modules are additionally consolidated to frame sub-blocks which will be utilized with a particular goal in mind in the modules. Sub-block 1 — This is utilized just utilized as the primary sub-block in the main module. Sub-block 2 — This is utilized as the primary sub-block in the wide range of various modules. Sub-block 3 — This is utilized for any sub-block with the exception of the first in every one of the modules.

Methodology

B. *Hyperparameter Tuning*

We experimented individually with different architectures of EfficientNet and ended up with similar results. In the first series of tuning, we started with a batch size of 4 and 1 epoch

Stage	*Operator*	*Resolution*	*Channels*	*Layers*
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	28 × 28	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

Model	*Batch Size*	*Epochs*	*Gradient Descent Algorithm*	*Accuracy*
B0	4	1	Vanilla	65%
B0	16	4	Vanilla	88%
B2	32	6	Vanilla	89%
B5	4	5	Vanilla	90%
B5	8	4	Adam	93%
B1	16	4	Adam	93.5%
B4	16	4	Adam	95%
B6	16	4	Adam	95%

(epoch = pass over the entire data) and a standard vanilla gradient descent algorithm for which the accuracy obtained was 65%, this seems like Under-fit. We then increased batch size to 16 and epochs to 4 (so that the number of iterations remains same) and obtained an accuracy of 88%. There was a drastic increase in accuracy from 65 to 88%. Then, we advanced from EfficientNet-B0 architecture to EfficientNet-B2 and increased batch size to 32 with the number of epochs as 2 for which we obatained an accuracy of 89%. Not quite an improvement. We then tweaked with the GD algorithm to improve our accuracy further and it indeed paid off. We changed from Vanilla gradient descent algorithm to Adam gradient descent algorithm as it is a better performing GD algorithm in many scenarios and ended up with an accuracy of 93.5%.

In the second series of tuning, we started directly with EfficientNet-B5 architecture with a batch size of 4 and 5 epochs and obtained an accuracy of 90%. The accuracy gradually increased by switching from Vanilla to Adam and on increasing batch size. The maximum accuracy we were able to obtain was 95% for EfficientNet-B6/B4 with batch size of 16 and epochs of 4. The summary of different hyper parameters tuned on different EfficientNet architectures in improving the accuracy are shown in the table below. The table is summarized based on our tweaking in the increasing order of accuracies. The last two rows in the table give the summary of the models that gave us the best validation accuracy of approximately 95% and are hence applied on the test data.

IV. Experimental Results and Performance Evaluation

A. DataSet

The dataset used here comprised of 11,166 images of 73 different sports [9] that are used for classification. The dataset is taken from Kaggle - a community that has many open datasets that can be used for different purposes. This dataset does not contain any duplicate images. All the images in the dataset are of the size 224X224X3 and in jpg format. The experimental setup used is google colaboratory. The dataset was divided into train (10,436 images), validation (365 images) and test splits (365 images).

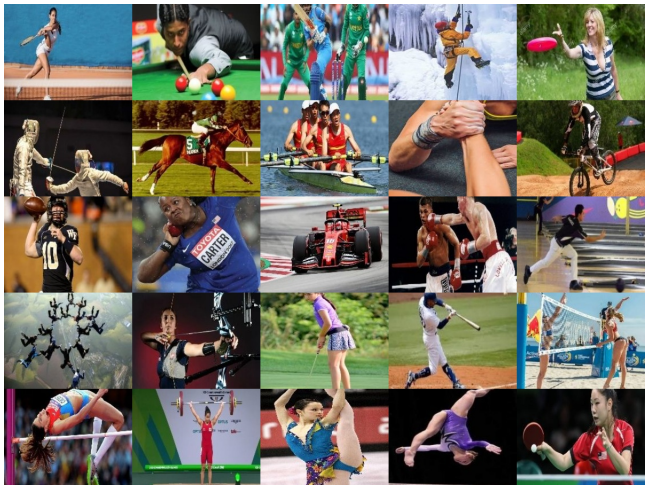

Fig. 4: Images corresponding to some of the sports in our dataset

B. Comparison

Though ResNet-50 converged quickly (compared to EfficientNet models), it was not able to obtain state of art accuracy. ResNext-101 also converged quickly. Infact, ResNext obtained 95% accuracy in around 1000 iterations, but it barely crossed 96 mark whereas EfficientNet models comfortably did that.

Model	*Accuracy (percentage)*
DenseNet-201	69.71
MobileNet-V2	91.78
ResNet-50	92.32
ResNext-101	96.71
EfficientNet-B4	98.36
EfficientNet-B6	97.53

The above is a comparison table that contains accuracies of different neural network models that were implemented on our dataset. DenseNet-201, MobileNet-V2 and Resnet-50 were already existing models on this dataset that gave accuracies of 69.71 %, 91.78 % and 92.32 % respectively. The above three models were implemented using tensorflow framework. The last three models in the table, ResNext-101, EfficientNet-B4 and EfficientNet-B6 were implemented by us that gave the highest accuracies of 96.71 %, 98.36 % and 97.53 % respectively in comparison to the other existing models. We built using pytorch framework. When compared with already existing models, we can see that the latest and complex neural network architectures seem to be giving better results in classification problems.

C. Results

Fig. 5: Some of the misclassified images from our dataset

Test loss graph and train loss graph for EfficientNet-B4 and EfficientNet-B6 are represented in Fig. 6 and Fig. 6 respectively. Throughout the training process of our models, the difference between train accuracy and validation accuracy was relatively small. Hence, we can say that our models were not over-fitting. They are not under-fitting as well because, validation and test accuracy are close to train accuracy.

First, we had a less number of iterations. Then, we increased the number of iterations until the accuracy became stable.

Fig. 5 show some of the misclassified images by the EfficientNet model that we implemented that gave us an accuracy of 98 %. In Fig 5.1, the sport is predicted as horse jumping when the actual sport is horse racing. Both the sports have horse in common and the characteristics of both the sports are very similar. In Fig 5.2, the sport is predicted as football when the actual sport is baseball. Here, in the image the player is throwing a ball that is locked in his palms making the ball not clearly visible. The players angle or pose is pretty close to that of a players pose in football. In Fig 5.3, shot put is predicted as javelin. Even there, the pose of the person while playing that sport is the same but the object used is different (ball and javelin stick in shot put and javelin respectively). There, the model failed in differentiating between the objects used. In Fig 5.4, snow boarding is predicted as surfing. Here, the object used are pretty close which is a board but the medium in which the sport is played is different. The pose is also close in both the sports. Here, the model failed in distinguishing between water and snow. In Fig 5.5, water polo is predicted as volleyball. Here also, the sport played is very close to each other and the object used is also the same which is a ball. In Fig 5.6, sport figure skating men is classified as figure skating pairs. In both the cases, the sport played is almost same. Here, the model failed in differentiating between individual players and players in pairs. In Fig 5.7, baseball is misclassified into class golf. The angle of the shot played is similar in both the sports. The pose in both sports that comes after playing a shot is pretty similar. The bat used is a little different. Here, the model failed in predicting the object in playing the sport. In fig 5.8, sport rugby is misclassified as basketball. In both the sports, the way of playing the game is same but the shape if the ball used in different. In rugby, the ball used is oval in shape and in basketball it is circular ball. Here, the model failed to predict the shape of the ball used. In fig 5.9, sport surfing is misclassified into sport figure skating woman. Here, the pose is same when the sport is being played but the medium in which it is played is different. The justification for misclassification here goes same as that pointed out in Fig 5.4. From all the above justifications, we concluded that there are a very few sports out of 73 sports in total that got misclassified. We used EfficientNet-B4 and EfficientNet-B6 architectures in classifying these results. Our model gave an accuracy of 98 % on test data. The remaining 2 % that went into the misclassification was justified above as per our intuitions.

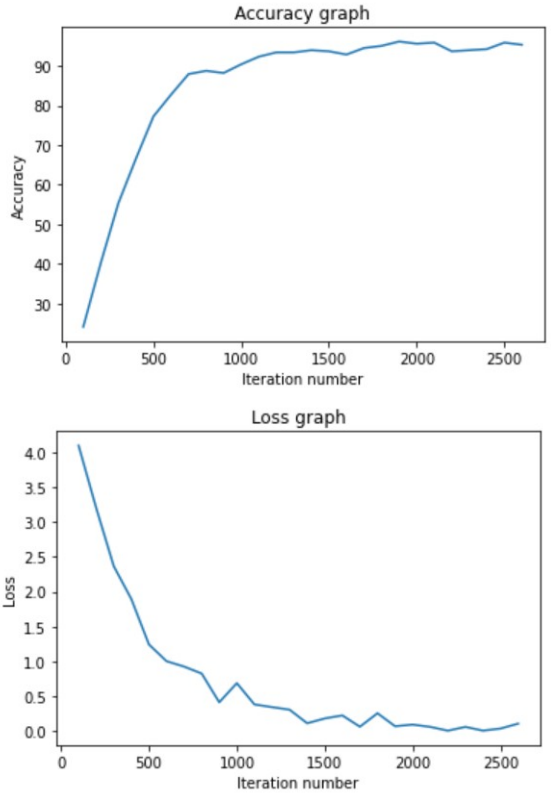

Fig. 6: Validation loss and accuracy graphs for EfficientNet-B4

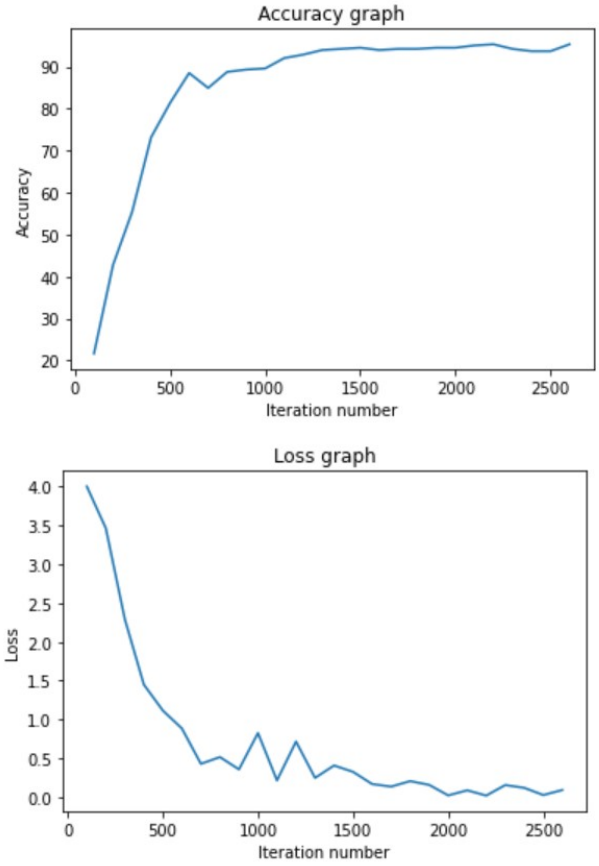

Fig. 7: Validation loss and accuracy graphs for EfficientNet-B6

V. CONCLUSION

The model was built on EfficientNet (B6 and B4) architectures that gave around 95% accuracy on validation data, 98% accuracy and 98% accuracy on test data. Many of the images that comes under the remaining 2% got misclassified because of the similar poses. They can be distinguished by a human (who knows the differences between the respective sports). The model was able to identify the poses correctly but was not able to distinguish the surroundings. Both EfficientNet-B4 and EfficientNet-B6 models converged quickly using the less number of iterations compared to other CNN models. We want to conclude by saying that EfficientNet models perform better than other the CNN models (that exist till date) like DenseNet-201, MobileNet-V2, ReseNet-50, ResNext-101 ,etc.

VI. REFERENCES

1. Ketan Joshi, Vikas Tripathi, Chitransh Bose and Chaitanya Bhardwaj ”Robust Sports Image Classification Using InceptionV3 and Neural Networks” (2020).
2. Wenrui Yang ”Analysis of sports image detection technology based on machine learning” (2019).
3. Shaohui Su ”Research on classification and application of sports images based on visual attention analysis” (2021).
4. Youna Jung, Eenjun Hwang, Wonil Kim ”Sports Image Classification through Bayesian Classifier” (2003).
5. Vardan Agarwal, ”Complete Architectural Details of all EfficientNet Models” (2020).
6. Yu-Liang Hsu, Hsing-Cheng Chang, Yung-Jung Chiu, ”Wearable Sport Activity Classification Based on Deep Convolutional Neural Network” (2019).
7. Mingxing Tan, Staff Software Engineer and Quoc V. Le, Principal Scientist, Google AI, ”EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling”, ICML 2019 paper.
8. Bruno Artacho, Andreas Savakis, Rochester Institute of Technology, Rochester, NY ”OmniPose: A Multi-Scale Framework for Multi-Person Pose Estimation” (2021).
9. Dataset - Gerry, ”73 Sports Image Classification” , Kaggle (2021).
10. Mehenag Khatun, Julker Nine, Md. Forhad Ali, Pritom Sarker, Varendra University ”Fruits Classification using Convolutional Neural Network”, July (2020).
11. Siddharth Das, ”CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…”, Medium, Analytics Vidya (2017).
12. Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al- Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie & Laith Farhan ”Review of deep learning: concepts, CNN architectures, challenges, applications, future directions” (2021).
13. Mingxing Tan, Quoc V. Le “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” (2019).
14. Anand Borad, ”Image Classification with EfficientNet: Better performance with computational efficiency”, Medium (2019).
15. Archit Semwal, Durgesh Mishra, Vineet Raj, Jayanta Sharma, Ankush Mittal, ”Cricket Shot Detection from Videos”, 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT).
16. Ashok Kumar, Javesh Garg, Advisor: Dr. Amitabha Mukerjee, ”Cricket Activity Detection”, Dept. of Computer Science and Engineering, ashokkrm,javeshg,amit @iitk.ac.in.
17. Mihai Lazarescu, Svetha Venkatesh, Geoff West, ”Classifying and Learning Cricket Shots Using Camera Motion”, Australasian Joint Conference on Artificial Intelligence AI 1999: Advanced Topics in Artificial Intelligence pp 13-23.
18. Anik Sen, Kaushik Deb, Pranab Kumar Dhar, Takeshi Koshiba, ”CricShotClassify: An Approach to Classifying Batting Shots from Cricket Videos Using a Convolutional Neural Network and Gated Recurrent Unit” (2021).
19. Satya Mallick, ”EfficientNet: Theory + Code”, LearnOpenCV (2019).