Identify plays based upon video footage

Karam Shbeb
Innopolis Unversity
k.shbeb@innopolis.university

Mohammad Shahin
Innopolis Unversity
m.shahin@innopolis.university

Mahmood Darwish
Innopolis Unversity
m.darwish@innopolis.university

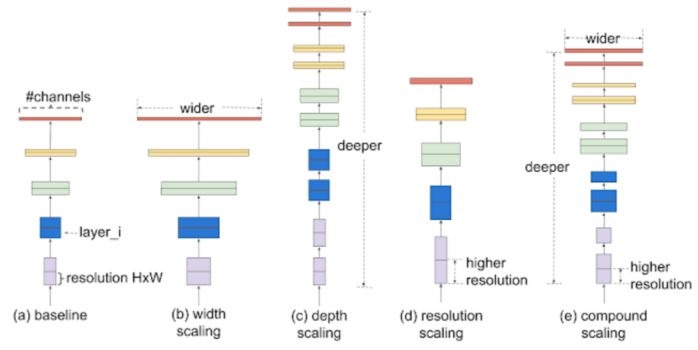
***Abstract*—This document is the Project Deliverable D1.3 for the Practical Machine Learning and Deep Learning course. Our project aims to detect football (soccer) passes, including throw-ins and crosses. Github repository.**

I. EFFICIENTNET MODEL

In D1.2 we introduced our model based on EfficientNet[1], and we used ”tf efficientnet b5 ap” implementation from the ”Timm” library.

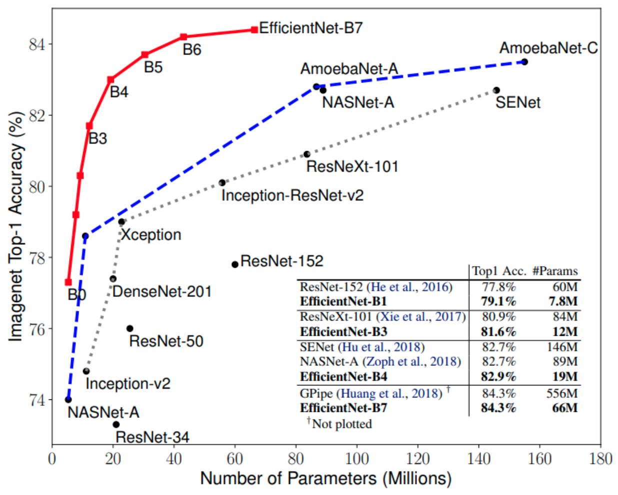
A. What is EfficientNet model

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.



B. Why EfficientNet

We use EfficientNet as our baseline model because they provide better capacity for more data[3] and we have relatively large data in our training dataset. Despite being suitable for large data EfficientNet surpasses many competitors in terms of accuracy such as VGG, ResNet, AlexNet, Iercep-tion v4, SqueezeNet, and DensenNet[4]. According to the author of EfficientNet [1] the architecture of this model can be sorted into 7 groups (B0 to B7). Even though B6 achieves the second highest accuracy after B1[1] we chose B6 because it's a light version of B7 and it's robust against adversarial attacks[2]. Choosing a robust algorithm is crucial in our case because our model is designed to run outdoors with inconstant intensity and lighting.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

II. TRAINING THE BASE-LINE MODEL

Since we split the video into frames and pass it into the model we turned the problem into 4 classes of classification (background, challenge, play, and throw-in). We split our training data into both validation and training data and feed it into the model. Since we are using a pre-trained model from ”Timm”, we train our model using this library which provides checkpoints. Therefore, we don't need to train the whole model continuously, and we can train each epoch or more separately. Timm provides a script that takes the checkpoints and averages the weights from the last few ones using the script provided by Timm to save it as a model. This feature is very important in our case because we don't have suitable hardware to run the model continuously. Finally, we plan to train our model with over 30 epochs but at this level, we trained only using one epoch due to the lack of feasible computational resources. However, we plan to overcome this problem by using a proper cloud service instead of our local GPUs.

III. TESTING THE BASE-LINE MODEL

Our testing data consists of video recordings from one full game and four half-games, the other half of each game being in the training set. In testing phase we use the same scheme that we used in training phase, we split our video into frames and we match it with the events using the timestamp written on the labeled test data. We run our model on the testing data and we extract the predictions of our model in order to compare it with the real labels. Firstly, we extract both scoring intervals and ground-truth events. Secondly, we map each event class to its prevalence and we create a table of detections with a column indicating a match to a ground-truth event. Thirdly, we remove detections which occur outside of scoring intervals. Finally, we compare our predictions with the real labels taking tolerance into account. The accuracy of our model is 7% which is relatively low but a good excuse for this accuracy is the low number of trained epochs (only one). However, training on more epochs would show if this reason is a pretext for the low accuracy or not.

IV. Work distribution

In this section of the work, Karam and Mohammad explored possible variants of the EfficientNet model and conducted a small research regarding the compromises associated with them. Mahmood and Mohammad trained and tested the model. Karam was also responsible for writing the report.

Bibliography cited

[1] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: (2019). DOI: 10.48550/ARXIV.1905.11946. URL: https://arxiv.org/abs/1905.11946.

[2] Cihang Xie et al. “Adversarial examples improve image recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 819–828.

[3] Qizhe Xie et al. “Self-training with noisy student improves imagenet classification”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10687–10698.

[4] Pan Zhang, Ling Yang, and Daoliang Li. “EfficientNet-B4-Ranger: A novel method for greenhouse cucumber disease recognition under natural complex environment”. In: *Computers and Electronics in Agriculture* 176 (2020), p. 105652.