**Classify sports images using Transfer learning**

**1. Introduction**

The hello word of CNN is the digit recognition in MNIST dataset. That itself has around 70,000 images and all the images are nicely centered. Also, like the figure 1 below, they are gray-scale images, so they have only 1 channel and therefore they don’t need many features for classification. This makes even a fully connected model like ANN to come out with high accuracy.

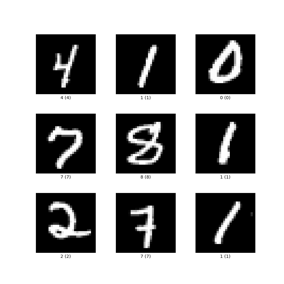
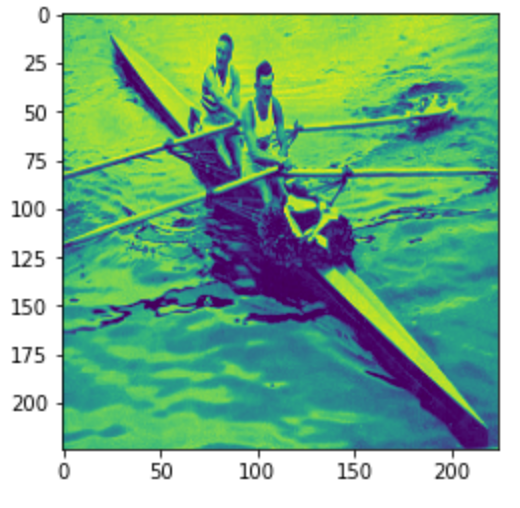


Figure 1.MNIST Dataset

But in the current situation of sports images, there are only 11,000 images covering 73 sports. The shape of the images is (224,224,3) but all the images are very different. There are color images but have different colors, zoom ranges, light levels, images where only the equipment is present while others with the player on the field etc. For example, the figure below shows the image that has only one channel. Therefore, a lot of preprocessing and feature variables will be required compared to MNIST data to achieve better performance and prevent overfitting.



**Figure 2. Image that has only one channel**

To compare the performance for two datasets with simple CNN model, I made a simple CNN model of the following Conv-Conv-Pool-Conv-Conv-Pool architecture for MNIST data and then trained it. The model was compiled with optimizer=’adam’, loss=’categorical crossentropy’, and MNIST data was trained with epochs=10 with 32 batch size. The test accuracy of the trained CNN model for MNIST data set was about 99.4%. It has very high accuracy without lots of preprocessing or very deep model.

테이블이(가) 표시된 사진

자동 생성된 설명

**Figure 3. Architecture of model**

For the same model above, I trained the model for sports dataset without any preprocessing. I just changed input shape and output shape of the model to apply the model for sports dataset. The test accuracy of the trained CNN model for sports dataset was about 18.4%. Also, in the process of training, validation accuracy continued to vibrate around 15%. The result means this simple CNN model is not suitable for Sports dataset. Hence, my main goal of this paper is to overcome the problems above mentioned and make suitable model for this dataset.

In the training process, the model has about 143 images for each class. Thus, to improve model performance, I will use a data augmentation method to increase the number of training images for each class.

**2. Related works.**

1. Transfer learning

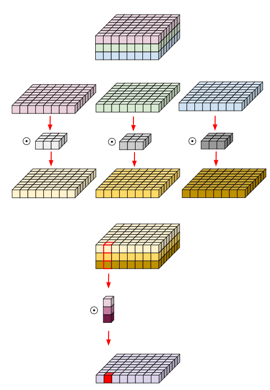
In traditional machine learning methods, the training data and test data are taken from the same domain and their input feature space are the same. In contrast to the traditional ML, transfer learning collects data from different domains and make use of these information to improve the learner. Thus, if the training data is not enough to train the learner, or if it has different kinds of input space features, then the need for transfer learning occurs. There are already many machine learning applications of the transfer learning such as text sentiment classification, human activity classification, and image classification.

Transfer learning is effective even when the number of learning data is small, and the learning speed is fast. It has the advantage of providing much higher accuracy than learning without transfer learning.

1. MobileNetV2

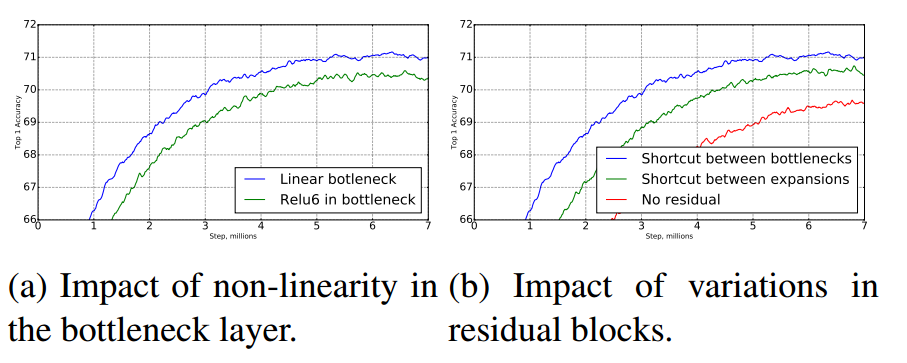
MobileNetV2 is a network that has solved the trade-off problem between speed and accuracy to some extent by making the model light-weight so that it can operate in real time, also the accuracy does not decrease too much.

In traditional convolutions, channels and filters are considered simultaneously to create the final output. However, the author of MobileNetV2 paper said that since cross-channels correlation (similarity between input channels) and spatial correlation (relationship between a filter and one specific channel) are completely independent, there is not any problem even if the channel and filter are separately trained. In fact, if I calculate the amount of computation, traditional convolutions are **Input image size x Input image channel x kernel size² x output channel**, but the computational amount of Depthwise Separable Convolutions is **Input image size x Input image channel x (kernel size² + output channel)**. Therefore, when the kernel size is set to 3, the amount of computation is reduced by 8 to 9 times.

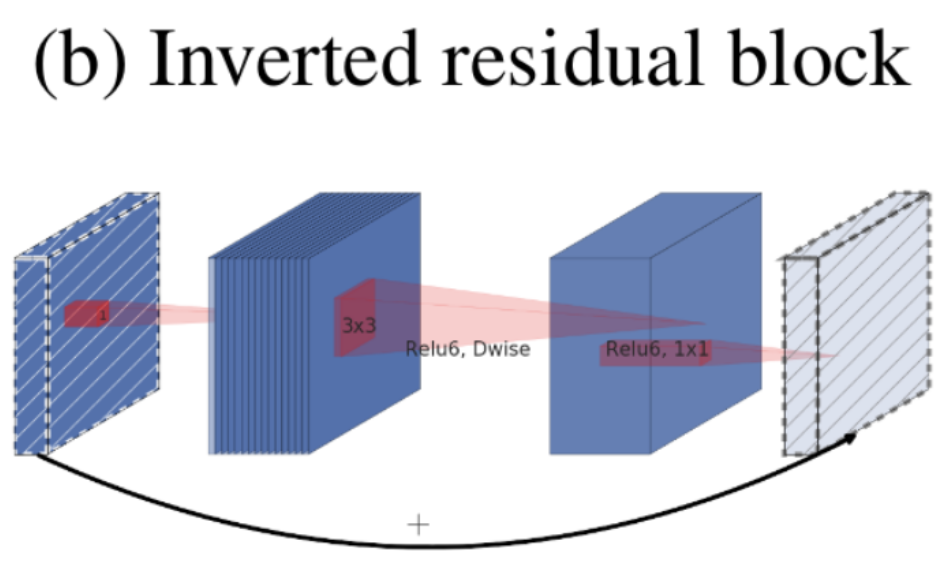


**Depthwise-separable Convolution**

Linear Bottleneck is a structure to reduce the amount of computation. A layer with a low number of channels in this BottleNeck structure uses a linear function. This is because information loss occurs when the nonlinear function ReLU is used for a layer with a small number of channels. The figure below shows the performance table when using non-linear functions and linear functions. Linear function performs better than ReLU.



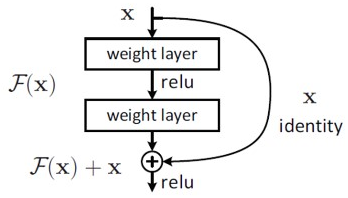
However, when the number of channels is large, important information is preserved even with ReLU. Therefore, Inverted residuals are proposed. As mentioned earlier, to use ReLU, we have to extend the channel. So, we extend the channel by pointwise convolutions and proceed with depthwise convolutions using ReLU. Finally, reduce the number of channels back to the original in Linear Bottlenecks.



(3) ResNet50 V2

ResNet is a model that first introduced Deep Residual Learning for Image Recognition. Before ResNet, other models tried to increase the total number of layers but noticed that the performance rather decreased, and the calculation speed becomes very slow. This was caused by the notorious vanishing/exploding gradients problem. To overcome this problem, some techniques (batch normalization, initialize parameters) applied. However, If the number of layers exceeds a certain number, it still becomes a problem.

ResNet successfully solved this problem by implementing a residual learning. By adding a shortcut connection, the model assures the minimum gradients to be 1 instead of 0 which causes the vanishing gradients problem. A short cut is a structure that is added immediately without a parameter, so there is no difference for the model except that addition is added. Therefore, there is no increase in computation through a short cut connection, but there is an effect of simplifying the forward or backward path because input and output are connected by crossing several layers.



By using the residual learning, Deep networks can be easily optimized, and accuracy can be obtained due to increased depth.

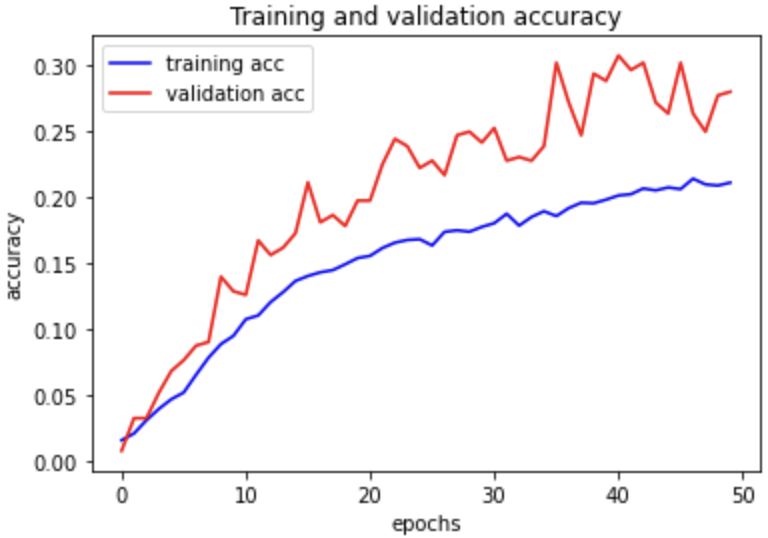
**3. Preprocessing**

First, I discarded the instances that do not have the shape of (224, 224, 3). Therefore, 9 instances from train set, 1 instance from test set are discarded.

The result of discarding some instances, dataset consists of 10407 train sets, 365 valid sets, 364 test sets. But I have to classify the images into 73 classes. It means one label has almost 150 images, which seems not enough to train the model. So, to train the model better, I use data augmentation technique to increase the dataset.

Among various data augmentation methods, I use augmentation technique of shifting, rescaling, rotating, shearing, zooming, and flipping method for the training and validation dataset. For the test dataset, I only applied rescaling method with the ratio same as the one applied on training and validation data.

In order to examine the effects of data preprocessing including data augmentation, the model introduced in the introduction step was equally applied to the dataset after preprocessing. In order to reduce the parameters to be estimated, the image size of (224,224,3) was changed to (50,50,3) and the model was fitted. The test accuracy of the trained CNN model was about 26.1%. This is a significant increase from 18.4% accuracy from the test accuracy in introduction step.



**Training and validation accuracy**

**4. Model Selection**

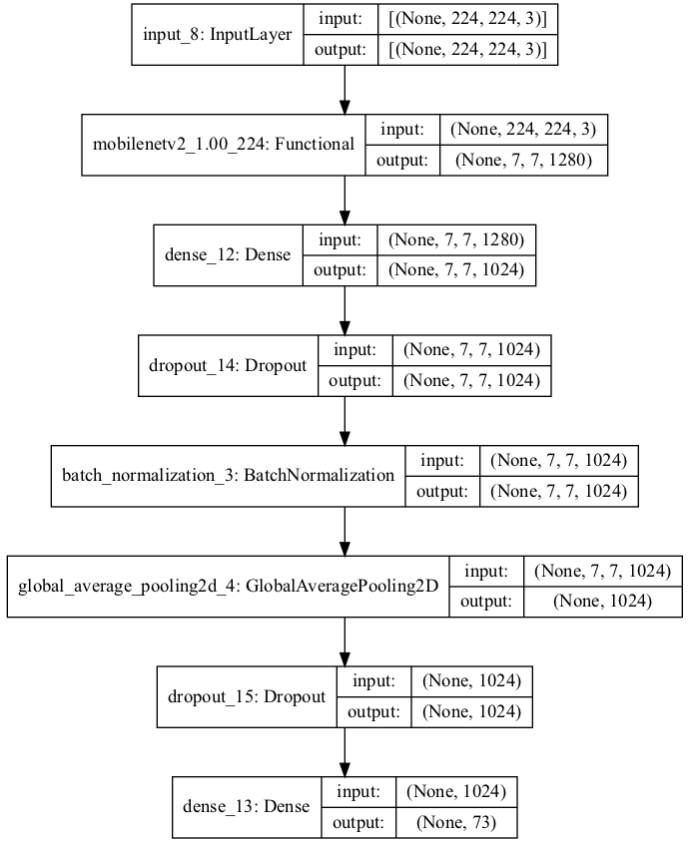
I tested MobileNetV2 and ResNet50v2.

**4.1 MobileNetV2**

**4.1.1 Implementation**

I used a MobileNetV2 architecture provided by keras.applications library. After preprocessing data with random rotation, shifting, and cropping, I performed transfer learning with training set. Also, I set the image size as (224,224,3) since the MobileNetV2 was made with the image with that size.

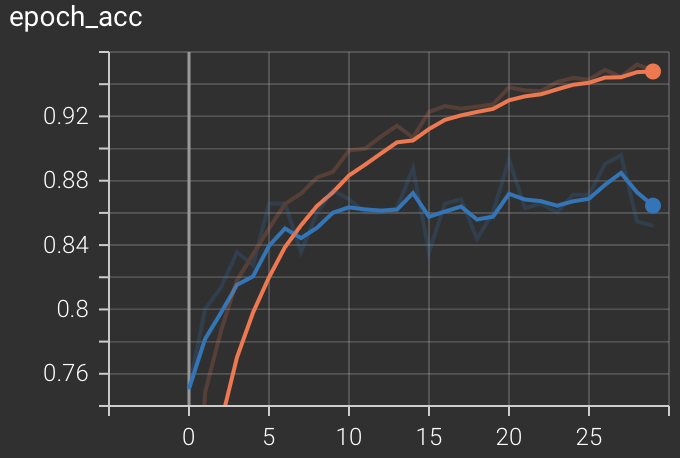
First, I make all MobileNetV2 layers not trainable to use weights since they are already trained. After the MobileNetV2 model, I add 1024 nodes layer and GlobalAveragePooling2D layer with some dropout to make the data 2D tensor. After that, I used 73 nodes layer with softmax activation function to classify 73 sports images.



Lastly, model is compiled with optimizer=’adam’ and loss=’categorical crossentropy’ and trained the model 30 epochs with 128 batch sizes.

**4.1.2 Results**

Figure below shows the training and validation accuracy for the MobileNetV2 architecture.



The orange line is the train accuracy, and the blue line is the validation accuracy. After 10 epochs, the model converged to a state of 0.85~0.87 accuracy on the validation set. The model scored 0.18 loss and 0.939 accuracy on the test set.

**4.2 ResNet50v2**

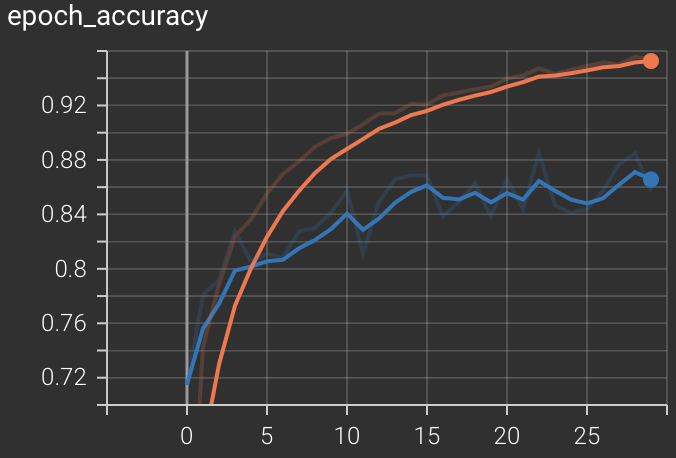
**4.2.1 Implementation**

I used a 50-layer ResNet architecture provided by keras.applications library. Also, I used same preprocessed dataset with above MobileNetV2.

After the ResNet model, I attached a 2048 x 1024 FC layer followed by a ReLU function, Dropout and batch normalization. Finally, I flattened the results and used a softmax layer for classification. This architecture is same with above MobileNet model.

For training, I used a batch size of 128 and 30 epochs. Weights were optimized with an adam optimizer with a categorical cross-entropy loss. I evaluated the models with accuracy.

**4.2.2 Results**



After 15 epochs the model converged to a state of 0.85 accuracy on the validation set. I used the best performed model to test performance on the test set.

The final model scored 0.16 loss on categorical cross-entropy and 0.945 accuracy on the test set. With the results, I can say that the transfer learning strategy with the ResNet model was highly effective on dataset. By the result, I saved the model using ResNet50V2 as a final model.

**5. Conclusion**

5.1. Summary

First, a simple CNN model was applied to the sports dataset, but the performance was not very good. Next, I do the preprocessing (data augmentation, remove the poor data... etc) and applied the preprocessed data to the same model as above. Although there was some performance improvement, the accuracy of the test set did not exceed 30%.

Therefore, I thought that a different method would be needed, and I tried to improve the performance by taking the weights of the already trained model. The MobileNetV2 model and the ResNet50 model were used for this, both showing an accuracy of nearly 95%. Therefore, I can say that this model can classify the 73 sports images well.

**[References]**

1. Guoliang K, Xuanyi D, Liang Z, Yi Y. PatchShufe regularization. arXiv preprint. 2017
2. Cecilia S, Michael JD. Improved mixed-example data augmentation. ArXiv preprint. 2018.
3. Daojun L, Feng Y, Tian Z, Peter Y. Understanding mixup training methods. In: IEEE access. 2018. p. 1.
4. Connor Shorten and Taghi M.Khoshgoftaar, A survey on Image Data Augmetation for Deep Learning. In Journel of Big data 2019.
5. Z. He, "Deep Learning in Image Classification: A Survey Report," 2020 2nd ITCA, 2020, pp. 174-177
6. Mark Sandler, Anerew Howard, Menglong Zhu. MobileNetV2: Inverted Residuals and Linear Bottlenecks. The IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520
7. HE, Kaiming, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.
8. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.
9. X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.
10. Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." IEEE Transactions on knowledge and data engineering 22.10 (2009): 1345-1359.