Rock Paper Scissors Game for Mobile Platforms

Rahmi Cemre Ünal, Yozdzhan Yalmaz
Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi, 34220 Istanbul, Türkiye
rahmicemreunal@gmail.com, yilmaz.ozcn@gmail.com

***Özetçe* —Bu projede, "Taş-Kağıt-Makas" oyununun üç el duruşunun görüntülerini mobil platform uygulamarında sınıflandırabilmek için bir mimari tasarlanmalıdır. Bu nedenle, görüntünün farklı el duruşlarını tanımlamak için derin öğrenme modeli oluşturulmalıdır. Sonuçlara dayanarak, mobil ve gömülü görüş uygulamaları için MobileNet adında verimli bir model kullanılmış ve daha iyi performans için sonuna katmanlar ekleyerek veri setimiz üzerinde eğitilmiştir.**

Anahtar Kelimeler—Yapay sinir ağları, derin öğrenme, görsel sınıflandırma.

***Abstract*—In this project, an architecture should be designed in order to classify the images of the three hand postures of "Rock-Paper-Scissors" in mobile platform applications. Therefore, a deep learning model should be developed to define different hand postures of the image. Based on the results, an efficient model called MobileNet was used for mobile and embedded vision applications and trained on our data set by adding layers to the end for better performance.**

Keywords—Artificial neural networks, deep learning, visual classification.

I. Introduction

CNNs [1] (Convolutional Neural Networks) have become ubiquitous in computer vision over the years. The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy. However, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. In real world applications especially for mobile ones, the user expect quickest response possible. Our project which is a image recognition model that can work on mobile platforms has been designed to achieve this goal in the most effective way.

II. Model architecture selection

A. Transfer Learning

Our efforts to create our own model did not yield the desired efficiency. At that point we decided to use MobileNet [2] with Transfer Learning [3].Many different approaches can be generally categorized into either compressing pretrained networks or training small networks directly. Mobilenet is class of network architectures that allows a model developer to specifically choose a small network that matches the resource restrictions (latency, size) for their application. MobileNet primarily focus on optimizing for latency but also yield small networks. Many model architecture focus only on size but do not consider speed. Our expectation also satisfied with prediction accuracy of MobileNet.

III. MobileNet Architecture

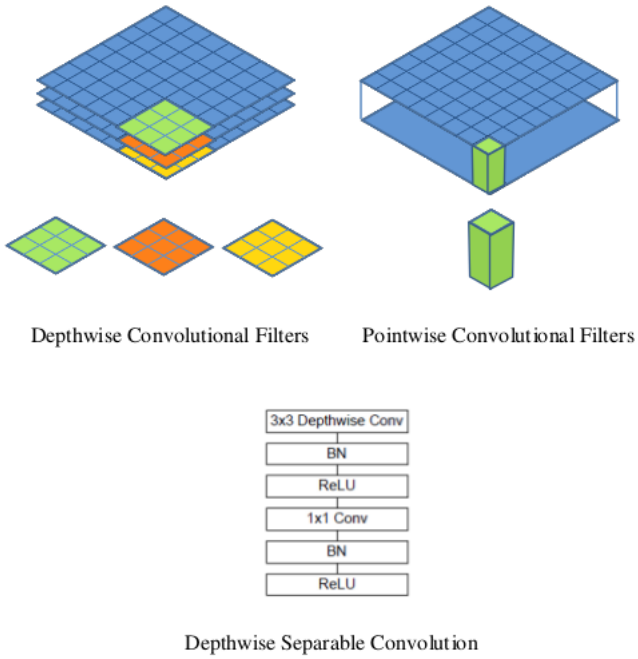

Figure 1 MobileNet's Filters Visualization

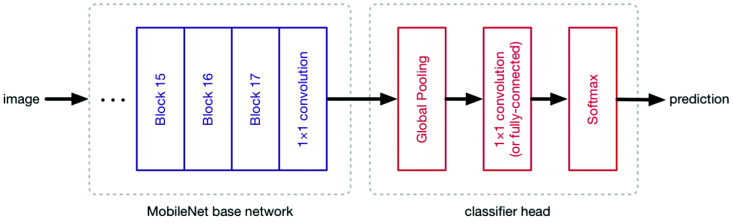

Figure 2 MobileNet Visualization

The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1-1 convolution called a pointwise convolution. For MobileNet the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a point to point (1-1) convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining.

This factorization has the effect of drastically reducing computation and model size.

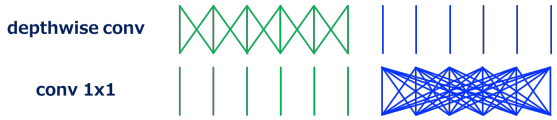

Figure 3 Depthwise and Pointwise Convolution

A. Equations

Standard convolutions have the computational cost of (1) where the computational cost depends multiplicatively on the number of input channels M, the number of output channels N the kernel size $D_k \times D_k$ and the feature map size $D_F \times D_F$. MobileNet models address each of these terms and their interactions. First it uses depthwise separable convolutions to break the interaction between the number of output channels and the size of the kernel. The standard convolution operation has the effect of filtering features based on the convolutional kernels and combining features in order to produce a new representation. The filtering and combination steps can be split into two steps via the use of factorized convolutions called depthwise separable convolutions for substantial reduction in computational cost.

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \tag{1}$$

Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions. We use depthwise convolutions to apply a single filter per each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output of the depthwise layer. MobileNets use both batchnorm and ReLU nonlinearities for both layers.

Depthwise convolution with one filter per input channel (input depth) can be written as (2) where $\hat{\mathbf{K}}$ is the depthwise convolutional kernel of size $D_K \times D_K \times M$ where the m_{th} filter in $\hat{\mathbf{K}}$ is applied to the m_{th} channel in $\mathbf{F}$ to produce the m_{th} channel of the filtered output feature map $\hat{\mathbf{G}}$.

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \tag{2}$$

Depthwise convolution has a computational cost of: (3)

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \tag{3}$$

Depthwise convolution is extremely efficient relative to standard convolution. However it only filters input channels, it does not combine them to create new features. So an additional layer that computes a linear combination of the output of depthwise convolution via 1×1 convolution is needed in order to generate these new features.

The combination of depthwise convolution and 1×1 (pointwise) convolution is called depthwise separable convolution.

Depthwise separable convolutions cost (4) which is the sum of the depthwise and 1×1 pointwise convolutions.

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \tag{4}$$

By expressing convolution as a two step process of filtering and combining we get a reduction in computation of: (5)

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} \frac{1}{N} + \frac{1}{D_K^2} \tag{5}$$

The accuracy-loss ratios of the model during the training and test stages are shown in the Table 1.

Table 1 Accuracy - Loss of every state

	Accuracy	**Loss**
Training	0.9591	0.1504
Validation	0.9569	0.1571
Test	0.9529	0.1543

The designed model has been trained with the data set containing rock paper scissors photographs, and as a result a low-sized CNN model has emerged which is able to make fast and effective decisions and is compatible with mobile platforms.

IV. Dataset

The majority of the data set required for model training was taken from shared hand gestures photographs for project developers over the Internet. Some of them were pulled together by the project developers. The total number of inputs has been increased by data generation algorithms in order to achieve the target accuracy rate of our project. One of the important factors in the selection of the shared data set on the Internet is that it consists of photos that are difficult to predict.

In the development process of our model, which was trained with a challenging data set, it was aimed to produce a good estimate although the classification of the photos was difficult. With a total of 10201 photographs consisting of 3397 pieces of stone, 3375 pieces of paper and 3429 pieces of scissors, the model was intended to work with high accuracy on photographs taken from all angles and in different styles.

As shown in the table 2, 60% of the photographs in the data set were reserved for the training process, 20% for the follow-up with validation and 20% for the testing of the accuracy of the model at the end of the training.

Table 2 Dataset

	ROCK	**PAPER**	**SCISSORS**	**TOTAL**
Training	2064	2012	2044	6120
Validation	629	710	701	2040
Test	704	653	684	2041
TOTAL	3397	3375	3429	10201

A. Some of the difficult examples in the dataset

In the Figure 4, the skin and background tones are very similar to each other.

Figure 4 Rock

In the Figure 5, the hand gesture is bad angled.

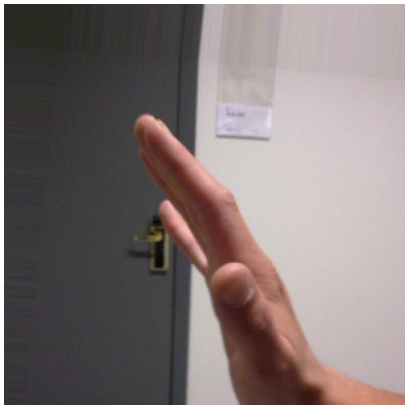

Figure 5 Paper

In the Figure 6, the hand gesture and legs in the background is quite similar so it might cause miss prediction.

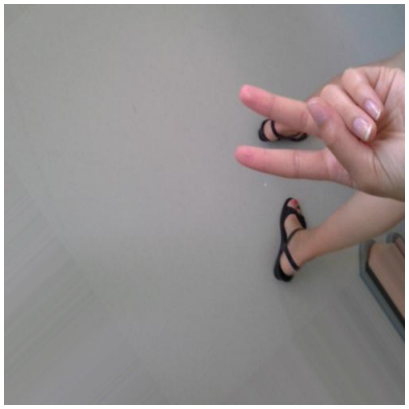

Figure 6 Scissors

B. Confusion Matrix

At the end of the training process, the confusion matrix of the accuracy of the estimates of the classes is shown in figure 7. The model was mostly confused the scissors with rock gesture.

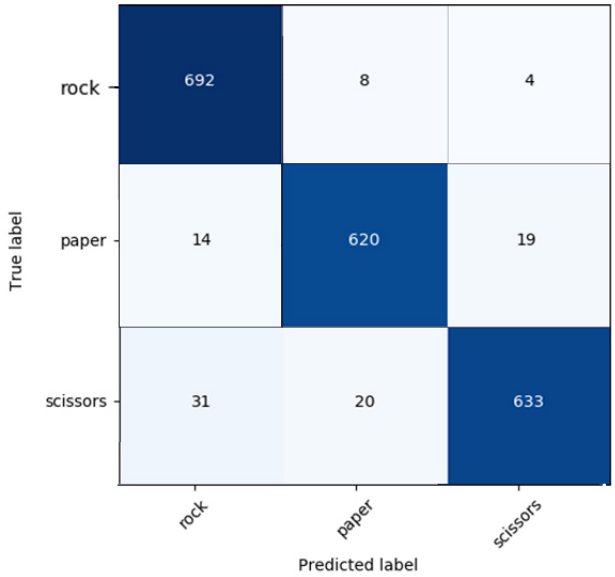

Figure 7 Confusion Matrix

V. Converting Model File

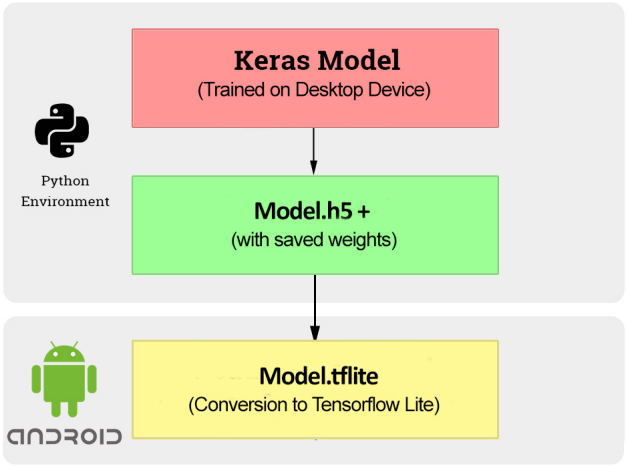

Figure 8 Converting Visualization

Our Keras[4] based model file, which works with high accuracy on the desktop, had to be saved and converted to ".tflite" extension type with Tensorflow infrastructure in order to use it on mobile platform.

Alternative conversion processes found were implemented through Google Colab due to Tensorflow version and developer platform incompatibilities.

VI. Conclusion

Within the scope of this project, an overview of deep learning and how to use the methods and techniques for convolution neural networks were examined in general.Particularly for our subject, a high-speed and accurate architecture was chosen to distinguish between stone, paper and scissor images.

The data set has been expanded with data replication techniques to increase accuracy.At the beginnig of process the data set was very small, the problem of over-adaptation of the model was solved by increasing the data set.The training of model, which can work on desktop computers, was completed with a difficult data set.

The model target is achieved with an accuracy rate of 95% - 97%, suitable for situations where critical factors such as background complexity, image quality, etc. are not optimal.

Afterwards, model weights were saved and converted to tensorflow [5] based model file to create a high accuracy model that can work on smart phones. It was tested that the model weights were maintained at the end of the conversion.

References

[1] (2019). [Online]. Available: https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad

[2] (2019). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919310658

[3] (2019). [Online]. Available: https://towardsdatascience.com/what-is-transfer-learning-8b1a0fa42b4

[4] (2019). [Online]. Available: https://keras.io

[5] (2019). [Online]. Available: https://www.tensorflow.org