2019

Car Image Classification Using Deep Neural Networks

Mingchen Li
Colby College

Classifying Car Image Using Deep Neural Network

Mingchen Li

Department of Comupter Science

Colby College, USA

Abstract

Image classification is widely used in many fields of study. Deep neural networks are proven to be effective classifier structure due to its massive parameters and training capability. This paper outlines the development of Deep Neural Network in recent years and applied them on a Car image dataset in order to compare their performances.

Keywords—Deep Network, Image Classifier

I. INTRODUCTION

Deep Neural Network is one of machine learning methods. It is one specific type of Artificial Neural Network that contains multiple layers between the image data input and the output classification. These layers are either linear or non-linear functions that extract important messages from the data. Based on their specific performance, layers have different designs such as convolutional, pooling, fully connected. The history of Neural Network as a machine learning design starts as early as 1980s. However, it was not until early 21st century when there are GPU implemented Neural Network. The primary obstacle for the development of Neural Network is the computational power it requires. For an effective Neural Network to be properly trained, it usually requires millions of parameters to be calculated repetitively. For computer in the early 21st century this is still a major challenge for the hardware. [1]

This paper focus on four designs of Deep Neural Network: VGG16, VGG19, Inception V3, and Xception. The VGG 16, VGG 19 both belong to the VGG family of Neural Networks. The VGG family is proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition. (Citation needed) They share the same design structure, that is the layers are grouped into different subgroups. Then layer them together as one. The Inception V3 and Xception model share similar design structure. The Inception V3 model is developed by Google researchers lead by Chiristian Szegedy(citation). The Xception is designed by Francois Collet. Instead building with subgroup of layers, these are built with inception modulo.

Is one design structure superior than another? This is the question this paper would like to answer. Chronologically the Inception family are newer models. They have the advantage of having less parameters which save the computation efforts. However, the VGG networks are also more proven in other studies. Furthermore, which design model is superior under similar design structure? In order to find out the answer, this paper uses the Car dataset to train each model and compare their classification rate.

II. DEFINITION

A. Convolution Layer

The Convolution Layer is consisted of a set of filters, some named as kernels. The filters take a subset of the input data at a time, but some are applied across the full input by sweeping over the input. The first operation performed by this layer are linear functions. Then the result will go through an activation function that is commonly non-linear. Finally, the layer combines outputs from potentially multiple channels with a simple linear function with bias. The details of the structure are shown in the image below[6]:

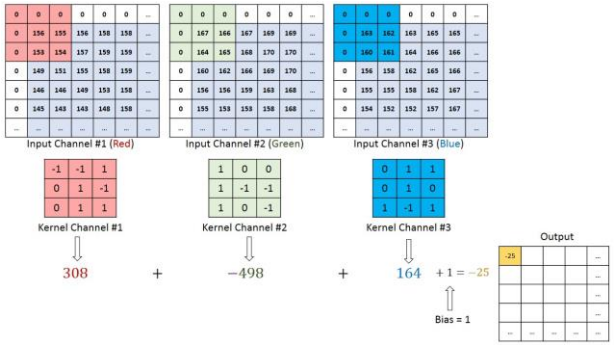

Figure 1 Convolution Layer

The primary objective of convolution layer is extracting the features from the input data. When dealing with high dimensional data such as images, it is impractical to parse every data point at once. Thus, the convolution layer only looks at a subset of the data first, then it traverses the filter with this size across the whole data. The size of the filter is one of the hyperparameters named as receptive field of the neuron.

The depth of the output volume controls the number of the neurons in a layer that connect to the same region of the input volume. The stride of the layer defines how many pixels the filter moves for each time step. The zero padding defines the size of zeros at the edge of the dataset. As shown in the image above, due to the size limit of the filter, putting zeros at the edge of the data is convenient in preserving the size of the filter throughout the convolution process.

B. Pooling Layer

The pooling layer is responsible for decreasing the computational power required to process the data through dimensionality reduction. There are two commonly used pooling algorithm: max pooling and average pooling. The max pooling extracts the largest element in the filter while the average pooling computes the average in the filter. The

pooling function can be either linear or non-linear. The max pooling can extract the most important features like edges in the image. The average pooling on the other hand will smooth out the result data. There is no definite proof of one pooling algorithm performing better than the other. However, in extreme image classification cases, such as images with a lot of sharp contrast and edges, max pooling is expected to perform better than average pooling.

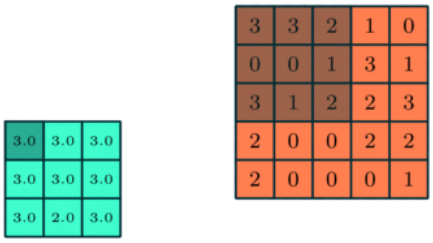

Figure 2 Pooling Layer

C. Fully Connected layer

The Fully Connected layer is an extreme case of convolution layer. Each input data is connected to the output of the layer with a weight. The receptive field is the size of the input data and it does not need any zero padding. Just like the convolutional layers, Fully Connected layer often contains one non-linear activation step afterwards. In general, the Fully Connected layer is the most computation intensive layer as it computes n inputs with n weights, each weight has the size of n whereas most common convolution layers have the weight of size less than n.

D. Number of Parameter

In Deep Neural Networks the number of parameters is an efficient method to estimate the amount of calculation needed. In convolution layer it has two kinds of parameters: weights and biases. The total number of parameters for a convolution layer is the sum of weight and biases. The size of the kernel is one of the significant factors that affect the number of parameters. For a convolution layer with filter size of 3×3, input of size 25 with 3 channels, the number of weights in this layer is $3 \times 3 \times 25 \times 3$. The number of biases is the size of the input data. Thus, the number of parameters in this convolution layer equals to $3 \times 3 \times 25 \times 3 + 25$.

For Pooling layer there is no parameters since the algorithm relied only on the hyperparameters such as pool size, stride and zero paddings.

The Fully Connected layer is often the last couple layers in a Deep Neural Network structure. Hereby we need to consider the case when the previous layer is a Fully Connected layer and the case when the previous layer is a convolution layer. If previous layer is a Fully Connected layer, then the weight for current layer is the number of neurons in earlier layer times the number of neurons in current layer. The number of bias is the same as the number of neurons in current layer. For a Fully Connected layer that follows after a convolutional layer, the number of weights is the size of the output image time the number of kernels in the previous layer times the number of neurons in the current layer. The number of bias is the number of neurons in current layer as well.

The number of parameters for a multiple layered Deep Neural Network is the sum of parameters in each layer. [3]

III. Building Models

In this section we introduce our models of interests in details. They are VGG16, VGG19, Inception V3, Xception. As mentioned in introduction, these four models have two different building strategies. This strategy will be further disclosed as well.

A. VGG 16

The VGG 16 is a deep neural network build with 21 layers. It contains five max pooling layers, three Fully Connected layers and thirteen convolutional layers. With five pooling layers, the network conduct convolution calculation on five different sizes of input data. Each input image is restricted to the size of 224×224 pixels. The input data is hereby the size of $224 \times 224 \times 3$ since images has three distinctive RGB channels. The input data is resized into ultimately $1 \times 1 \times 1000$ at the very end of the study. The detailed architecture is shown below:

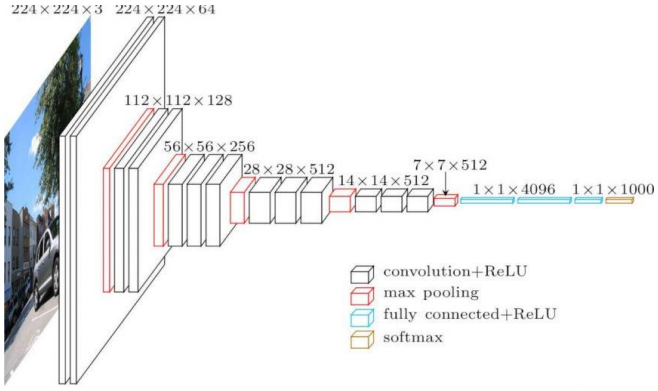

Figure 3 VGG16 Architecture

The VGG family of Deep Neural Network is improved over AlexNet which is the first efficiently GPU implemented Deep Neural Network in 2012 [1]. Compare to the eight layers structure and large-sized filters, the VGG family explored the possibility of adding more layers while reducing the size of the filter to keep the computation level manageable. In AlexNet there are one 11×11 kernel sized layer and one 5×5 kernel sized layer as shown in the image below

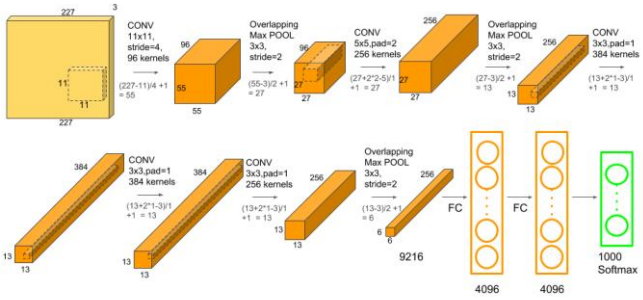

Figure 4 AlexNet Architecture

In VGG networks, the size of the kernel is modified into 3×3 which drastically reduced the size of the computation in each layer. As disclosed earlier, the number of parameters of

a convolution layer with 3×3 filter size is much smaller than a convolution layer with 11×11 filter size. If we assume all other parameters are consistent, then the ratio of weights between the two is $\frac{3^2}{11^2}$. Thus when we replace one convolution layer with 11×11 filter with three convolution layers of 3×3 filters, we are actually reducing the number of parameters in our model since the weight ratio is $\frac{11^2}{3 \times 3^2} = \frac{121}{81}$. As result, The VGG 16 structure is able to create deeper network while preventing the network's parameter to increase at exponential rate. For a 16-layered structure, it has 138.36 million parameters while AlexNet's 8-layer structure has 62.39 million parameters. Thanks to the depth it created, the VGG 16 outperforms its predecessors with top 5 error rate of 7.3% versus 16.4 of that for AlexNet in ImageNet Large Scale Visual Recognition Challenge.

B. VGG 19

Following the same strategy, the VGG 19 is built by adding one more convolution layer in each training step. It increases the number of convolution layer from 13 to 16. However, its error rate is not significantly better than that of VGG 16[8].

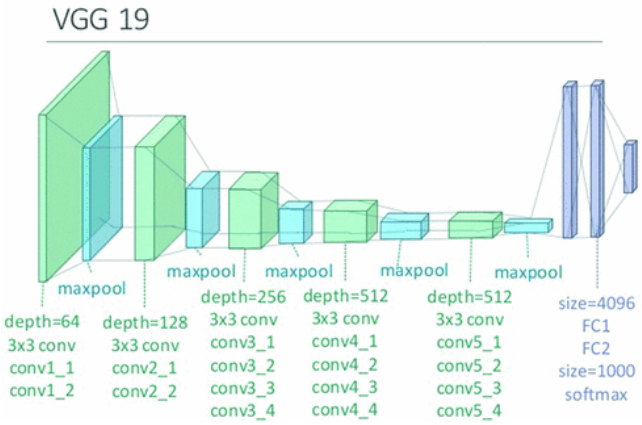

Figure 5 VGG19 Architecture

Despite the VGG architectures are performing well compare to their predecessors, there are inherent flaws in these designs. The first problem is that they are very slow to train. Because the Fully Connected layer in this architecture have over one hundred million parameters, the sum of parameters is incredibly large. This causes the amount of calculation needed for training to be very large. The second problem is that the size of filter is consistent throughout which causes some messages to be overlooked.

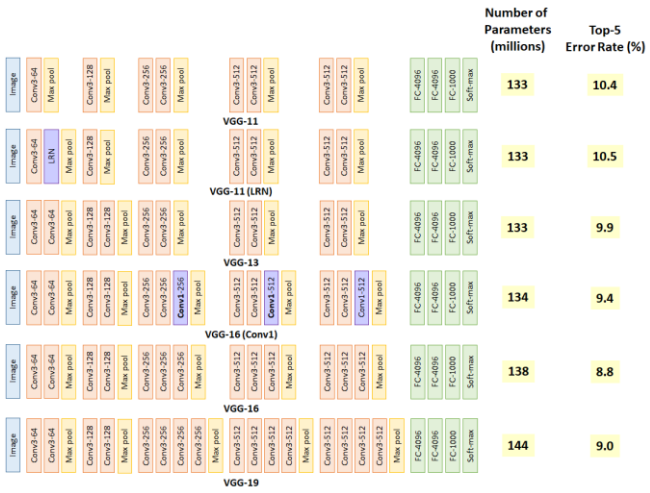

Figure 6 Architecture and Number of Parameters in VGG family

C. Inception V3

In order to tackle the problem of VGG network mentioned above, the inception model is developed by Google scholars. The key building block of Inception architecture is the Inception Module. It consists of multiple channels of convolution layers that explores multiple sizes of kernel. After processing the input data with a 1×1 convolution layer to reduce computation complexity, the data goes through each channel of convolution layer in parallel. Then the outputs are concatenated together as one.

But why should we try different sizes of kernel? The main reason is that the main object in an image varies in size. For example, the cat object is likely to occupy small portion of an image as well as half of the image. Our classifier aims to be capable to classify both cases as cat category. Thus, it becomes necessary to be able to have parameters of different kernel sizes. The image below discloses a canonical Inception Modulo [4]:

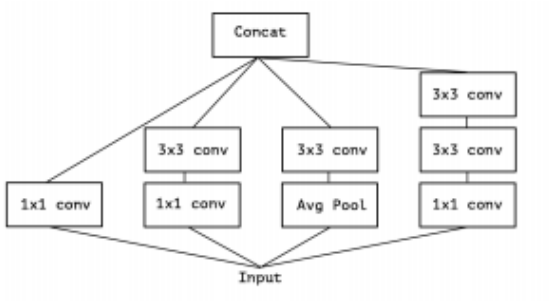

Figure 7 Inception Modulo

Notice that on the right most channel, there are two 3×3 filters. This design aims to reduce parameter as well. Using the same calculation method mentioned above, the two 3×3 filters have less weights than that of a 5×5 filter. The weight ratio is $\frac{\{3 \times 3 + 3 \times 3\}}{5 \times 5} = \frac{18}{25}$. This design builds a good local network within a bigger network.

Another advantage of Inception Architecture is that it is able to remove the Fully Connected layer in the very end of the network. This result is revolutionary such that the model is capable of stacking more layers while keeping the parameters very low. For the Inception architecture shown

below, it only requires 5 million parameters which is unbelievably low compare to the VGG architecture.

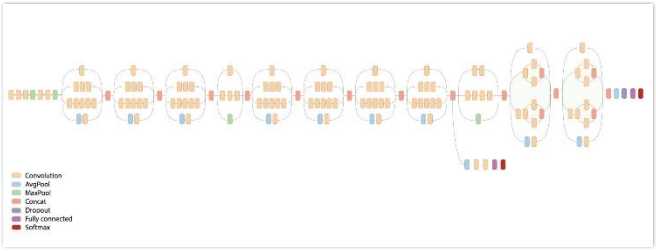

Figure 8 Inception V3 Architecture

D. Xception

The Xception model is designed by François Chollet in 2017. It is an improvement over the Inception model and it is built over the assumption that the mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled. In Inception for example, we cut the $2 \times 2 \times 4$ data into smaller, say 2×4 data using a 1×1 convolution layer. The Xception architecture look at each 2D data separately and then conduct the 1×1 convolution. This combination of 2D and 1D mapping is less computationally intensive than training in a 3D dataset[5]. The following image illustrate this process:

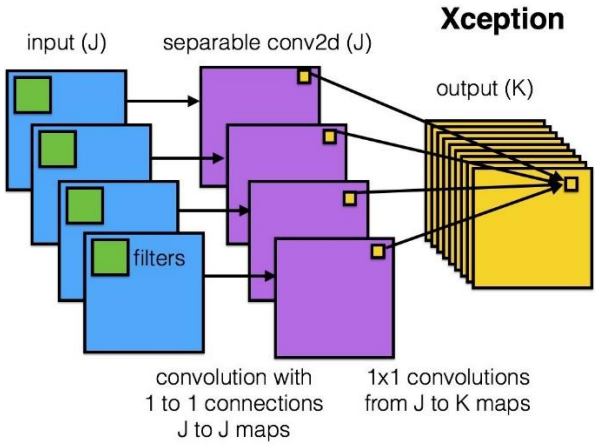

Figure 9 Xception Architecture

The Xception model has the similar number of parameters to that of InceptionV3.It consists of 36 convolution layers which are put into 14 Inception modulo. Compare to InceptionV3, Xception slightly outperforms InceptionV3 in the ImageNet dataset which consists of 17,000 classes.

IV. Results

In this paper we focus on classification of car images provided by Stanford University. The "Cars" data set consists of 16185 images of 196 classes of cars. The data is split into 8144 training images and 8041 testing images. The image split is roughly at 50-50. Classes are at the level of *Make, Model, Year.* For instance, the image below all come from the class *Audi S5 Coupe 2012*.[2] As reader might notices, within the same class there are same model with multiple colors, shot angles and sizes. The discrepancy of image data allows us to test for the performance of Inception modulo. The implementation of models mentioned above are included in Keras package. With the same hyperparameters in each model, each model is trained and tested.

Figure 10 Sample Data Image

Figure 11 Sample Data Image

Figure 12 Sample Data Image

Each model is trained with different number of epochs. The training is conducted on Colby SSH Secure Shell. The hardware includes two NVIDIA Tesla P100-PCIE-16GB GPUs. When the number of epochs reaches over 50, the performance of Inception modulo based models overtook the VGG models. Among all four models, the Xception architecture performs the best over 100 epochs. Below disclose some results in graph:

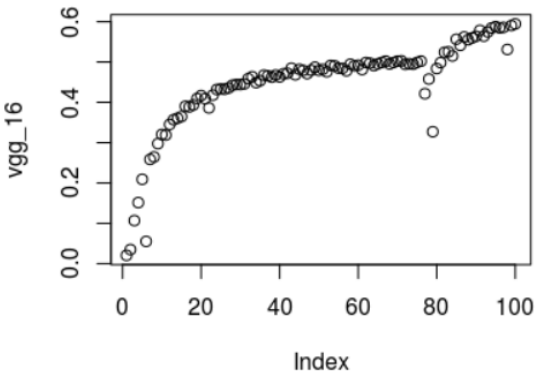

Figure 13 VGG16 Classification Rate Over 100 Epochs

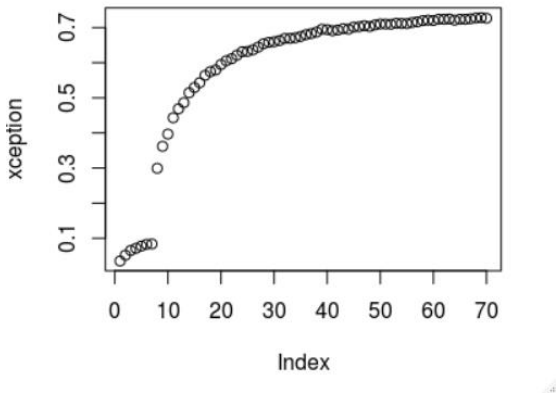

Figure 14 Xception Classification Rate Over 70 Epochs

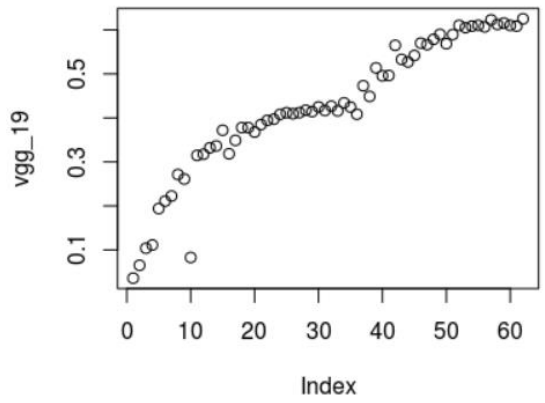

Figure 15 VGG19 Classification Rate Over 60 Epochs

TABLE I. CLASSIFICATION RATE OVER 100 EPOCHS

MODEL	VGG-16	VGG-19	INCEPTIONV3	XCEPTION
TOP1 CLASSIFICATION RATE (OVER 100 EPOCHS)	0.5945	0.625125	0.67584	0.72675

Figure 16 Classification Rate Results

DISCUSSION

As expected, the InceptionV3, Xception perform better due to their advanced architectures. The training time for these inception-based models also surpass that of VGG based models due to the small number of parameters. There are no signs that these models have experienced overfitting problem during classification. In conclusion, the Inception-modulo based architecture performs better on Cars data set when the number of epochs is under 100. This result is consistent to the result from Derrick Liu and Yushi Wang from Stanford University, who trained multiple models on the same data set, but with huge number of epochs of over 50,000[7]. Due to their heavily repetitive training, the classification rate for each model is higher than this paper's result. Notice that the Inception model is referenced as GoogLeNet in figure 17.

	Top 1 accur.	Top 5 accur.	Final loss
Baseline SVM	0.031		55.48
Baseline ConvNet	0.062		42.73
CaffeNet fine-tuned	0.447		2.62
CaffeNet partial-train	0.418		2.61
CaffeNet full-train	0.417		3.21
CaffeNet scratch	0.005		5.31
GoogLeNet fine-tuned	0.774	0.943	1.25
GoogLeNet partial-train	0.775	0.943	1.28
GoogLeNet full-train	**0.800**	**0.951**	1.09
GoogLeNet scratch	0.347	0.636	6.42
VGGNet fine-tuned	0.789	0.942	**1.01**
VGGNet scratch	0.008	0.031	5.51

Figure 17 Top 1 Classification Rate from Works by Liu and Wang

ERROR ANALYSIS

In classification results the VGG16 classification rate drops when number of epochs reaches 80. There is no significant result that explains the reason behind this incident and after repetitively training and testing the model, the discrepancy disappears from time to time. Similar discrepancy occurs at other models as well.

FUTURE WORK

Although the results have given a definitive answer under Cars data set, there is still much to explore.

Additional models: Within the scope of this project it is possible to explore more architectures provided by Keras package. Models such as ResNet, MobileNet, DenseNet can shed different lights to the study of Deep Neural Networks.

Furthermore, given the knowledge of VGG and Inception models, it is also possible to design one unique Deep Neural Network that utilize Inception Modulo as well as common building components such as Convolution Layers.

More Datasets: Our work with the Cars dataset can be migrated to other works as well. When we have tested the models with multiple, distinctive data sets, the results become more reliable.

More Epochs: Due to the limit of hardware, it is unfortunate that models in this study are limited to be trained under 100 epochs. However, in the future it is possible to conduct training with more epochs so that we are able to understand the potency of each model better.

Application: Given that our trained model already achieved 72.7% accuracy, it is possible to put it into test in real life. For example, it is possible to create a real time classification application toward security camera that recognize the car model then a vehicle is in sight. This implementation can be done by taking the screen shot of the video and conduct classification over multiple scree shots. Furthermore, the model can be implemented on facial recognition as well, which potentially can be a great tool for security purposes.

ACKNOWLEDGMENT

I would like to thank **Professor Bruce Maxwell**, **Professor Stephanie Taylor** for their advice and guidance during this project and **Randall Downer** for providing the GPU which allows the training to be much faster.

REFERENCES

[1] Kumar Chellapilla; Sid Puri; Patrice Simard (2006). "High Performance Convolutional Neural Networks for Document Processing". In Lorette, Guy (ed.). Tenth International Workshop on Frontiers in Handwriting Recognition. Suvisoft.

[2] Stanford Cars Datasets. (2019). [online] Available at: https://ai.stanford.edu/~jkrause/cars/car_dataset.html [Accessed 22 May 2019].

[3] Nayak, S. (2019). Number of Parameters and Tensor Sizes in a Convolutional Neural Network (CNN) | Learn OpenCV. [online] Learnopencv.com. Available at: https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/ [Accessed 22 May 2019].

[4] Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. [online] Available at: https://arxiv.org/abs/1610.02357 [Accessed 22 May 2019].

[5] Kdnuggets.com. (2019). An Intuitive Guide to Deep Network Architectures. [online] Available at: https://www.kdnuggets.com/2017/08/intuitive-guide-deep-network-architectures.html/2 [Accessed 22 May 2019].

[6] Li, F., Johnson, J. and Yeung, S. (2017). CNN Architectures.

[7] Liu, D. and Wang, Y. (2015). Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning. [online] Available at: http://cs231n.stanford.edu/reports/2015/pdfs/lediurfinal.pdf [Accessed 22 May 2019].

[8] Neurohive.io. (2019). VGG16 - Convolutional Network for Classification and Detection. [online] Available at: https://neurohive.io/en/popular-networks/vgg16/ [Accessed 22 May 2019].