Generic Object Classification Using Deep Convolutional Neural Networks

Keerthy Prasannan[1], Anju Rachel Oommen[2], Smita C Thomas[3]
[1, 2, 3]Dept of Computer Science and Engineering
[1, 2, 3]Mount Zion College of Engineering, Kadammanitta, Pathanamthitta, Kerala

Abstract- *In machine learning a convolutional neural network is a class of deep feed forward artificial neural network that has successfully been applied to analyzing visual imagery. Image classification can perform some pretty amazing feats, but a large drawback of many image classification applications is that the model can only detect one class per image. With an object detection model, not only can you classify multiple classes in one image, but you can specify exactly where that object is in an image with a bounding box framing the object.Traditional neural network is good at image classification have many more parameters and take a lot of time if trained on CPU.*

I. INTRODUCTION

Image classification can perform some pretty amazing feats, but a large drawback of many image classification applications is that the model can only detect one class per image. With an object detection model, not only can you classify multiple classes in one image, but you can specify exactly where that object is in an image with a bounding box framing the object.Network that we will implement in this project is smaller and simpler (than the ones that are used to solve real-world problems) so that you can train this on your cpu as well. While training, images from both the classes(guitar/violin) are fed to a convolutional layer which is followed by 2 more convolutional layers. After convolutional layers, we flatten the output and add two fully connected layer in the end. The second fully connected layer has only two outputs which represent the probability of an image being a guitar or violin.The TensorFlow Models GitHub repository has a large variety of pre-trained models for various machine learning tasks, and one excellent resource is their object detection API. The object detection API makes it extremely easy to train your own object detection model for a large variety of different applications. Whether you need a high-speed model to work on live stream high-frames-per-second (fps) applications or high-accuracy desktop models, the API makes it easy to train and export a model.

A Convolutional Neural Network (CNN) is a powerful machine learning technique from the field of deep learning. CNNs are trained using large collections of diverse images.

From these large collections, CNNs can learn rich feature representations for a wide range of images.

II. LITERATURE REVIEW

A. Hong, Z. Chi, G. Chen and Wung,proposed the first suggestion that the information content derived from an image can be classified into three levels; they are Low level: Include visual features such as colour, texture,shape, spatial information and motion.Middle level: Include the existence or arrangement of specific types of objects, roles and scenes.High level: Include impressions, emotions and meaning features. Examples include objects or scenes with emotional or religious significance.Feature extraction is a process to extract low-level image features such as colour, texture, shape and spatial information.Its objective is to capture the essential characteristics of the patterns [1].

Z. Wang, R. Datta, J. Dhiraj and J. Li, in their study offered the features are used to represent an image instead of using the original pixel values. Feature extraction can thus be used to transform the input data and in some way find the best-input representation for NN . In this study, colour and texture are used to represent flower images for building classification model.Colour is an important feature in an imageclassification since it helps to narrow down the possible image categories. In fact, colour histogram of the guitar violin region has been used to characterize the colour feature of the images [2].

R. Al-Tayeche and A. KhalilCui, designed a colour feature extraction, the choice of a colour space is important. A colour space is a multi-channel space which represents the different components of colour in an image. Most colour spaces are three dimensional. For example, Red Green Blue(RGB) colour space will assign each pixel into three colour intensities which provides useful primary information for representing colour features of images. However, the RGB colour space is influenced by intensity and illumination from sun or camera lighting and do not correspond to equal perception of colour dissimilarity [3]

M. E. Osadebey, showed one possible solution to overcome the RGB problem is by transforming the RGB colour space to Hue Saturation Value(HSV) colour space since HSV colour space is closer to human colour vision. [4].

I. J. Tsang and I. R. Tsang., in their study offered Texture has been one of the most important characteristics which has been used to classify and recognize objects and scenes defines texture as the uniformity, density,coarseness, roughness, regularity, intensity and directionality of discrete tonal features and their spatial relationships.[5]

J. R. Sveinsson and J. A. Benediktsson, proposed the texture descriptors can be classified into three categoriesFeatures that are computed in the spatial domain.The function of GLCM is to calculate the approximate image properties.[6]

III. EXISTING SYSTEM

Image classification can perform some pretty amazing feats, but a large drawback of many image classification applications is that the model can only detect one class per image.

LIMITATIONS

- The model can only detect one class per image.
- A lot of training period is required.
- Number of parameters are considered.

IV. PROPOSED SYSTEM

The proposed system with an object detection model, not only can you classify multiple classes in one image, but you can specify exactly where that object is in an image with a bounding box framing the object.

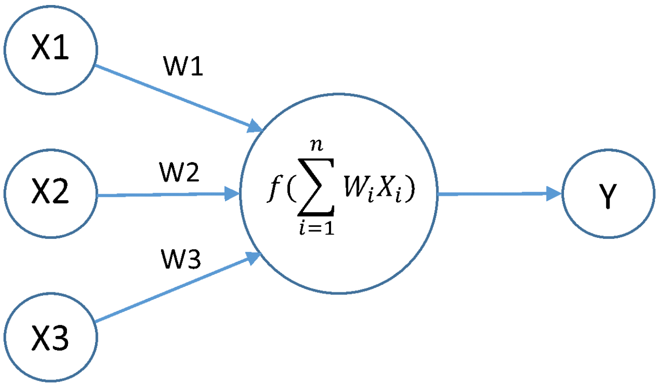
Network that we will implement in this paper is smaller and simpler (than the ones that are used to solve real-world problems) so that you can train this on your cpu as well. While training, images from both the classes(guitar/violin) are fed to a convolutional layer which is followed by 2 more convolutional layers. After convolutional layers, we flatten the output and add two fully connected layer in the end. The second fully connected layer has only two outputs which represent the probability of an image being a guitar or a violin.In this paper overcomes the limitations of existing system.

NEURAL NETWORK

Neural Networks are essentially mathematical models to solve an optimization problem. They are made of neurons, the basic computation unit of neural networks. A neuron takes an input(say x), do some computation on it(say: multiply it with a variable w and adds another variable b) to produce a value (say; z= wx+b). This value is passed to a non-linear function called activation function(f) to produce the final output(activation) of a neuron. There are many kinds of activation functions. One of the popular activation function is Sigmoid, which is: $f(x) = \frac{1}{1+e^{-x}}$ The neuron which uses sigmoid function as an activation function will be called Sigmoid neuron. Depending on the activation functions, neurons are named and there are many kinds of them like RELU, TanH etc. One neuron can be connected to multiple neurons.

SIGMOID NEURON

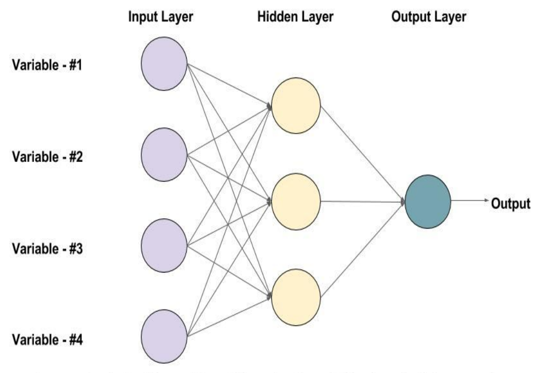
The neuron which uses sigmoid function as an activation function will be called Sigmoid neuron. Depending on the activation functions, neurons are named and there are many kinds of them like RELU, TanH etc(remember this). One neuron can be connected to multiple neurons, like this. In this example, you can see that the weights are the property of the connection,i.e. each connection has a different weight value while bias is the property of the neuron.This is the complete picture of a sigmoid neuron which produces output y: $f(x) = \frac{1}{1+e^{-x}}$



Sigmoid neuron

LAYERS

Stack neurons in a single line, it's called a layer; which is the next building block of neural networks.



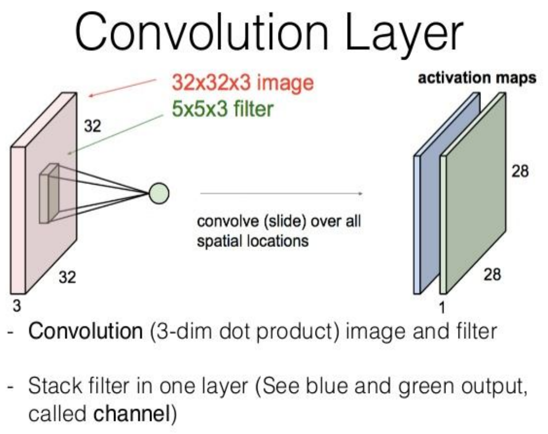
An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Feed Forward Neural Network

The neurons in green make 1 layer which is the first layer of the network through which input data is passed to the network. Similarly, the last layer is called output layer as shown in red. The layers in between input and output layer are called hidden layers. In this example, we have only 1 hidden layer shown in blue. The networks which have many hidden layers tend to be more accurate and are called deep network and hence machine learning algorithms which uses these deep networks are called deep learning. Typically, all the neurons in one layer, do similar kind of mathematical operations and that’s how that a layer gets its name(Except for input and output layers as they do little mathematical operations)

CONVOLUTIONAL NEURAL NETWORK

Convolution is a mathematical operation that’s used in single processing to filter signals, find patterns in signals etc. In a convolutional layer, all neurons apply convolution operation to the inputs, hence they are called convolutional neurons. The most important parameter in a convolutional neuron is the filter size, let’s say we have a layer with filter size 5*5*3. Also, assume that the input that’s fed to convolutional neuron is an input image of size of 32*32 with 3 channels.


Convolution Layer
32x32x3 image
5x5x3 filter
activation maps
32
32
3
28
28
1
convolve (slide) over all spatial locations
- Convolution (3-dim dot product) image and filter
- Stack filter in one layer (See blue and green output, called channel)

Convolutional layer

Let’s pick one 5*5*3(3 for number of channels in a colored image) sized chunk from image and calculate convolution(dot product) with our filter(w). This one convolution operation will result in a single number as output. We shall also add the bias(b) to this output. In order to calculate the dot product, it’s mandatory for the 3rd dimension of the filter to be same as the number of channels in the input. i.e. when we calculate the dot product it’s a matrix multiplication of 5*5*3 sized chunk with 5*5*3 sized filter. In this case, we slide our window by 1 pixel at a time. If some cases, people slide the windows by more than 1 pixel. This number is called stride.If you concatenate all these outputs in 2D, we shall have an output activation_map of size 28*28(can you think of why 28*28 from 32*32 with the filter of 5*5 and stride of 1).

Typically, we use more than 1 filter in one convolution layer. If we have 6 filters in our example, we shall have an output of size 28*28*6. As you can see, after each convolution, the output reduces in size(as in this case we are going from 32*32 to 28*28). In a deep neural network with many layers, the output will become very small this way, which doesn’t work very well. So, it’s a standard practice to add zeros on the boundary of the input layer such that the output is the same size as input layer. So, in this example, if we add a padding of size 2 on both sides of the input layer, the size of the output layer will be 32*32*6 which works great from the implementation purpose as well. Let’s say you have an input of size N*N, filter size is F, you are using S as stride and input is added with 0 pad of size P. Then, the output size will be:(N-F+2P)/S +1

CORRECT (WEIGHTS/PARAMETERS)

Once you have decided the architecture of the network; the second biggest variable is the weights(w) and biases(b) or the parameters of the network. The objective of the training is to get the best possible values of the all these parameters which solve the problem reliably. For example, when we are trying to build the classifier between guitar and violin, we are looking to find parameters such that output layer gives out probability of guitar as 1(or at least higher than violin) for all images of guitar and probability of guitar as 1((or at least higher than violin) for all images of guitar.

The best set of parameters using a process called Backward propagation, i.e. you start with a random set of parameters and keep changing these weights such that for every training image we get the correct output. There are many optimizer methods to change the weights that are mathematically quick in finding the correct weights. GradientDescent is one such method(Backward propagation and optimizer methods to change the gradient is a very complicated topic. But we don't need to do it manual now as Tensorflow takes care of it).

We start with some initial values of parameters and feed 1 training image(in reality multiple images are fed together) of dog and we calculate the output of the network as 0.1 for it being a guitar and 0.9 of it being a violin. Now, we do backward propagation to slowly change the parameters such that the probability of this image being a guitar increases in the next iteration. There is a variable that is used to govern how fast do we change the parameters of the network during training, it's called learning rate. If you think about it, we want to maximise the total correct classifications by the network i.e. we care for the whole training set; we want to make these changes such that the number of correct classifications by the network increases. So we define a single number called **cost** which indicates if the training is going in the right direction. Typically cost is defined in such a way that; as the cost is reduced, the accuracy of the network increases. So, we keep an eye on the cost and we keep doing many iterations of forward and backward propagations(10s of thousands sometimes) till cost stops decreasing. There are many ways to define cost. One of the simple one is mean root square cost. Let's say yprediction is the vector containing the output of the network for all the training images and actual is the vector containing actual values(also called ground truth) of these labeled images. So, if we minimize the distance between these two variables, it would be a good indicator of the training. So, we define the cost as the average of these distances for all the images:

$cost=0.5\sum i=(yactual-yprediction)$

This is a very simple example of cost, but in actual training, we use much more complicated cost measures, like cross-entropy cost. But Tensorflow implements many of these costs so we don't need to worry about the details of these costs at this point in time.After training is done, these parameters and architecture will be saved in a binary file(called model). In production set-up when we get a new image of guitar/violin to classify, we load this model in the same network architecture and calculate the probability of the new image being a guitar/violin. This is called *inference* or *prediction*.

For computational simplicity, not all training data is fed to the network at once. Rather, let's say we have total 1600 images, we divide them in small batches say of size 16 or 32 called *batch-size*. Hence, it will take 100 or 50 rounds(iterations) for complete data to be used for training. This is called one *epoch, i.e.*in one epoch the networks sees all the training images once. There are a few more things that are done to improve accuracy but let's not worry about everything at once.

V. CONCLUSION

Trained Model and data: In the git repository, I have only added 500 images for each class. But it takes more than 500 images of guitar/vioin to train even a decent classifier. So, I have trained this model on 2400 images of each class. This Mini-guitar-violin-dataset is a subset of github guitar-violin dataset and is not owned by us. The file guitar-violin-model.data-00000-of-00001 contains the trained weights(values of variables) of the network. So, once we have recreated the graph, we shall restore the weights. In order to get the prediction of the network, we need to read & pre-process the input image in the same way(as training), get hold of y_pred on the graph and pass it the new image .

VI. ACKNOWLEDGMENT

We would like to thank, first and foremost, Almighty God, without his support this work would not have been possible. We would also like to thank all the faculty members of Mount Zion college of engineering, for their immense support.

REFERENCES

[1] A. Hong, Z. Chi, G. Chen and Wung. Z, 2003."Region-of- interest basedflower images retrieval," Proc. IEEE International Conference of Acoustics, Speech, and Signal Processing, pp. 589−592

[2] Z. Wang, R. Datta, J. Dhiraj and J. Li, , 2008. "Image Retrieval: Ideas,Influences, and Trends of the New Age,"

ACM Transactions on Computing Surveys, Vol. 20, pp. 1-10.

[3] R. Al-Tayeche and A. Khalil, 2003.“CBIR: Content Based Image Retrieval,”Final Report. Carleton University.

[4] M. E. Osadebey, 2006.“Integrated Content-based image retrieval using Texture, Shape and Spatial Information”, Master Thesis report in Media Signal Processing, Umea University, Sweden.

[5] I. J. Tsang and I. R. Tsang, 1998 “Handwritten character recognition based on moment features derived from image partition,” Proceedings International Conference on Image Processing (ICIP ’98)

[6] J. R. Sveinsson and J. A. Benediktsson, 1997.“Feature extraction for neural network classifiers using wavelets and tree structured filter banks.Remote Sensing- A Scientific Vision for Sustainable Development”,IEEE International on Geoscience and Remote Sensing 1997 (IGARSS'97)

[7] M. E, Nilsback and A. Zisserman, 2006. "A visual vocabulary for flowerclassification," In CVPR, Vol. 2, pp. 1447–1454,.

[8] E. Aulia, 2005"Heirarchical Indexing for Region based image retrieval,” A dissertation submitted to the Graduate Faculty of the Louisiana State University and Agricultural and Mechanical College,.

[9] J. Smith, 2001 “Color for Image Retrieval, Image Databases: Search and Retrieval of Digital Imagery”, John Wiley & Sons, New York, pp.285-311