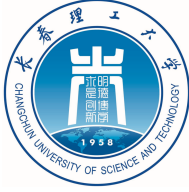


学号： Z18050042



长春理工大学  
王大绪  
Changchun University of Science and Technology

# 硕士学位论文(毕业)论文

基于 ResNet 和 GoogleNet 网络的烟雾识别算法

研究生 : OGUCHI EBUKA PHILIP  
学科专业 : 计算机科学与技术  
类别领域 : 计算机应用技术  
指导教师 : 方明

Smoke recognition algorithm based on ResNet and GoogleNet networks

二〇二〇年六月

分类号： 0436.3

密 级： 可公开

U D C：                     

编 号：                     

## 基于 ResNet 和 GoogleNet 网络的烟雾识别算法

学位授予单位及代码： 长春理工大学 (10186)

学科专业名称及代码： 计算机科学与技术(080300)

类别领域名称及代码： 计算机应用技术(080300)

研 究 方 向： 计算机视觉                      申请学位级别： 硕士

指 导 教 师： 方明                              研 究 生： OGUCHI EBUKA PHILIP

论文起止时间： 2019.9-2020.06

# **Smoke recognition algorithm based on ResNet and GoogleNet networks**

by

OGUCHI EBUKA PHILIP

Dissertation submitted to

**Changchun University of Science and Technology**

in partial fulfillment of the requirement

for the degree of

**Master of Engineering** (工学硕士用)

Supervisor

Professor Fang Ming

June, 2020



## 长春理工大学学位论文原创性声明

本人郑重声明：所提交的硕士学位论文《基于 ResNet 和 GoogleNet 网络的烟雾识别算法》是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：  \_\_\_\_\_ 2020 年 06 月 06 日


## 长春理工大学学位论文授权使用授权书

本学位论文作者及指导教师完全了解长春理工大学硕士、博士学位论文版权使用规定，即：研究生在校攻读学位期间论文工作的知识产权单位属长春理工大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为长春理工大学。同意长春理工大学保存并使用学位论文；同意长春理工大学向国家图书馆、中国学术期刊（光盘版）电子杂志社、中国科学信息研究所送交学位论文的复印件和电子版，允许论文被查阅和借阅。（涉密论文在解密后遵守此规定）

论文涉密情况约定：（打“√”选择）

本论文非涉密论文。

本论文属于涉密论文，在 \_\_\_\_\_ 年解密后使用本授权书。

作者签名：  \_\_\_\_\_ 2020 年 06 月 06 日

导师签名： \_\_\_\_\_ 2020 年 06 月 06 日



## ABSTRACT IN CHINESE

人工智能已经通过各种各样的电子设备和工具改变了人类的交流方式。运行某种机器学习或深度学习算法的照相设备已经能处理各种识别任务，已经有超过 97% 的智能手机搭载了此类智能成像设备。安装在公司、工厂和公路等地的成像设备也在运行各种各样的算法来处理不同的问题。

在一些特定场合判断一张照片中是否有烟雾存在非常重要。它能帮助我们更容易地检测烟雾并提前预警。烟雾识别技术有助于减少火灾发生时导致的生命和财产损失。

对于这个课题我们引入两种神经网络架构并分别进行试验验证，通过比较这些模型的运行结果，选出一个在准确率最高的模型。在 GoogleNet 或 ResNet 之类的卷积神经网络上训练数据，并用随机梯度下降算法来使模型平均化，最终正确地分类有烟雾和没烟雾的图片。

这个网络也用真实的数据进行了训练，观察在真实数据集上的分类性能。避免了在线数据集可能不能真实反映真实情况的问题。

本文关于算法架构如何完成分类给出了深入的分析，对于如何识别烟雾给出了很好的解决方案。





# ABSTRACT

Artificial intelligence has changed the way of human communication through a variety of electronic devices and tools. The camera equipment running a machine learning or deep learning algorithm has been able to handle a variety of recognition tasks, and more than 97% of smart phones have been equipped with such intelligent imaging devices. Imaging devices installed in companies, factories and highways are also running various algorithms to deal with different problems. It is very important to judge whether there is smoke in a picture on some specific occasions. It can help. It's easier for us to detect smoke and warn in advance. Smoke identification technology helps to reduce the loss of life and property caused by fire.

For this subject, we introduce two kinds of neural network architecture and test them respectively. By comparing the operation results of these models, we choose a model with the highest accuracy. Training data on convolutional neural networks such as Google net or RESNET, and using random gradient descent algorithm to average the model

The most correct classification of smoke and no smoke pictures. The network is also trained with real data to observe the classification performance on real data sets. The problem that online data sets may not reflect the real situation is avoided.

This paper gives an in-depth analysis on how to complete the classification of algorithm architecture, and gives a good solution for how to identify smoke.

## DEDICATION

This thesis dissertation is dedicated to almighty God and my family whose support and prayers have seen me through my graduate studies.

## ACKNOWLEDGMENTS

The successful completion of this work would not be possible without the Grace of God Almighty and the support of so many people. I would like to thank the Chinese Government Scholarship Board for giving me full time scholarship opportunity to study in China for my graduate degree. I will like to thank my supervisor Prof. Fang Ming for his invaluable guidance, patience and willingness to encourage me during this research. Also, to my laboratory colleagues, friends and everyone that have contributed to this research in one way or the other. My appreciation also goes to my Mentor Mrs. Uche Obi for giving me a push to always make a step-in life. I would like to thank my family for their love, patience and support which has been a guide to me in all I do.

# Table of Content

ABSTRACT IN CHINESE .....	I
ABSTRACT .....	V
DEDICATION .....	VI
ACKNOWLEDGMENTS .....	VII
Table of Content .....	VIII
1. Introduction .....	1
1.1 Background of study .....	1
1.2 History of computer vision: .....	2
1.3 Research Motivation: .....	5
1.4 Problem statement: .....	5
1.5 Limitations: .....	5
1.6 Research outline: .....	5
2. Literature Review .....	7
3. Methodology .....	10
3.1 Image classification: .....	10
3.2 Linear classification: .....	11
3.3 Supervised learning: .....	13
3.3.1 Neural network regression: .....	16
3.3.2. Neural network classification: .....	17
3.4 Loss function: .....	17
3.4.1 Multiclass SVM Loss: .....	17
3.4.2 Cross Entropy Loss / Negative Log likelihood: .....	18
3.5 Optimization: .....	19
3.5.1 Numeric gradient: .....	19
3.5.2 Analytic gradient: .....	20
3.5.3 Vanilla gradient descent: .....	20
3.5.4 Stochastic gradient descent (SDG): .....	21
3.6 Backpropagation: .....	21
3.7 Artificial Neural network: .....	24
3.7.1 Convolutional Neural Networks (CNNs): .....	26
3.7.2 Convolutional Neural Network Architectures: .....	29
3.8 GoogleNet: .....	29
3.8.1 Vanilla Inception module: .....	30
3.8.2 GoogleNet Inception Module: .....	30
3.8.3 Global average pooling: .....	32
3.8.4 The GoogleNet-v1 Architecture: .....	33
3.9 Residual Neural Network (ResNet): .....	34
3.9.0 OUR WORK FLOW: .....	38
3.9.1 Transfer learning: .....	39
3.9.2 Dimension Calculation: .....	40
4. Experiment, Results and Analysis .....	42
4.1 Datasets collection: .....	42
4.1.1 Real smoke dataset: .....	42

4.1.2 Synthetic dataset:.....	43
4.2 Data preparation: .....	43
4.3 Training, Validation and Test data: .....	44
4.4 Data preprocessing: .....	44
4.5 Experimental procedures: .....	46
4.6 Training Requirements and details: .....	46
4.6.1 Training and Hyper parameter optimization and Result: .....	46
4.8 SUMMARY: .....	47
5. Conclusion .....	48
6. Reference .....	49

# List of Tables

Table 1.1: Convolution Network Architectures.....	2
Table 2.0: Dimension Calculation for GoogleNet-v1.....	40
Table 3.0: Dimension calculation for ResNet-34.....	41
Table 4.0: Training and Validation results.....	47
Table 5.0: Testing results. ....	47

# List of figures

Fig1.1 Gemma Frisius camera obscura .....	2
Fig 1.2: ILSVRC error rate from 2012 till 2015[13].....	4
Fig 1.3: ImageNet top 5-error rate for visual recognition.....	4
Fig 3.1 what the computer sees .....	10
Fig 3.2: Simple linear classification network .....	12
Fig 3.3 Representation of various activation function in neural network .....	16
Fig 3.4: Data flow diagram of supervised learning .....	18
Fig 3.5: Simple back propagation for a single neuron. ....	22
Fig 3.6: A simple Computational circuit of backpropagation.....	23
Fig 3.7 biological neuron and its mathematical model.....	25
Fig 3.8: A 3-layer neural network or a 2 hidden layer neural network.....	25
Fig 3.9: A simple Convolutional neural network architecture .....	27
Fig 3.10: Pictorial representation of max pooling and average pooling[42].....	27
Fig 3.11: When Zero padding =1.....	28
Fig 3.12: Vanilla inception module. ....	30
Fig 3.13: GoogleNet inception module.....	31
Fig 3.14: Naïve convolution without 1*1 conv bottleneck. ....	31
Fig 3.15: Introducing a bottle neck by adding 1*1 conv filter.....	32
Fig 3.16: Fully Connected layer (FC) vs Global Average Pooling. ....	33
Fig 3.17: GoogleNet Architecture[4]. ....	34
Fig 3.18: Stacking layers on a plain network[5]. ....	35
Fig 3.19: A simple illustration of skip connection in a deep network.....	35
Fig 3.20: The ResNet- 34[5] Architecture. ....	37
Fig 3.21: Batch Normalization and Relu activation in Residual block. ....	37
Fig 3.22: Variants of Residual Blocks in lower and higher layers of ResNets. ....	38
Fig 3.23: Our Work flow.....	39
Fig 4.1 Setup for smoke data samples capture.....	43
Fig 4.2: Real smoke data and background image samples. ....	45
Fig 4.3: Synthetic smoke and background images download from the internet. ....	45





# 1.Introduction

## 1.1 Background of study

Image classification task is one of the remarkable breakthroughs made in computer vision. With recent trends, computers can now recognize and classify images more than humans do. This is not an easy task but with various research and development in the field of computer vision, state of the art algorithms has been developed to classify images correctly as humans do.

With the advancement of deep learning, image classification has surpassed human capability. As far as you have more data, neural networks can be trained to recognize any images. Image classification is the first step before detection, classification and localization, image capturing etc.

Nowadays, in our daily lives we can use our mobile cameras to recognize various images just by a single click. So many home appliances and gadgets now run on many advance deep learning algorithms that enables the cameras see. Cameras can now classify different kind of dog breeds, cat breads, different types of smoke images whether dense or transparent smoke, static or moving smoke.

The rise of artificial intelligence has deepened the thought of humanity in search for various ways to improve our lives. Various machine learning and deep learning algorithms have been of tremendous help to how computers see and how they interpret what they see. Deep learning is a subset of machine learning which uses deep neural network and lots of data to learn and to predict accurately new data. Convolutional Neural Networks are a special kind of multi-layer neural networks. Like almost every other neural network, they are trained with a version of the back-propagation algorithm. Where they differ is in their architecture. This Convolutional Neural Networks are designed to recognize visual patterns directly from an image with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

There are various Convolution neural networks architectures like LeNet-5[1], Alex Net[2], ZFNET[3], GoogleNet[4], ResNet[5], VGG-NET[6]. These convolutional neural networks can be used for image classification but for the purpose of this research we are going to restrict our self to GoogleNet-v1 and ResNet-34 to classify smoke images.

The purpose of this research is to compare two convolution neural networks for smoke image classification. GoogleNet and ResNet are large scale convolutional networks which have won ImageNet large scale image visual recognition challenge (ILSVRC). GoogleNet won ILSVRC in 2014 with a top-5 error rate of 6.67%, while ResNet won for 2015 with top-5 error rate of 3.6% which is above the human standard measure which is 5.1%. Below is a summary of the various Convolutional neural network architectures:

Table 1.1: Convolution Network Architectures.

YEAR	CONVOLUTIONAL NETWORK	DEVELOPED BY	PLACE	TOP-5 ERROR	NO OF PARAMETERS
1998	LeNet[1]	Yann Lecun et al			60 thousand
2012	AlexNet[2]	Alex Krizhevsky et al	1st	15.3%	60 million
2013	ZFNet[3]	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogleNet[4]	Google	1st	6.67%	4 million
2014	VggNet[6]	Simonyan et al	2nd	7.3%	138 million
2015	ResNet[5]	Kaiming He	1st		
	Human			5.1%	

## 1.2 History of computer vision:

Computer vision started 543million years, B.C. Animals didn't have eyes, they do their work by using other senses. After a short time, the number of species grew rapidly and animals began developing eyes. That was when vision started. Vision system is most advanced sensory system in the visual cortex that enables us to move around, manipulate things, communicate, eat etc. By the 16<sup>th</sup> century the first camera was built based on pin hole camera theory which was very similar to the first eyes with a hole that collect the information and a plane at the back that collect and process the information at the back.

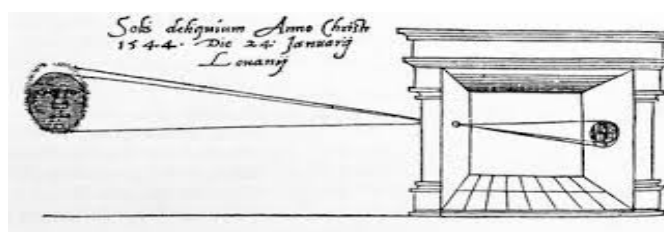


Fig1.1 Gemma Frisius camera obscura[7]

Biologists like Hubel and Wiesel,1959 influenced computer vision greatly with their work on vision using electrophysiology. They wanted to understand the mechanism of vision in animals so they used a cat brain which is similar to human brain from a visual perceptive. The stick an electrode at the back of the cat brain which is where the primary visual cortex is located. They learnt that there are many different types cells in the primary visual cortex of a cat. The primary (simple) cells responds to oriented gradient and edges

and as information builds up in the visual world, complexity sets in vision processing until they can understand the complex world. The first computer vision research was from Larry Roberts, 1963 PhD thesis (Block world) where the visual world was represented as simple geometry shapes. David Marr, 1970 wrote a book on what computer vision is all about. He thought that in order to take an image and represent it fully, we have to pass through some processes. The first process is the primal sketch where the image is represented as simple structures as zero crossings, blobs, edges, bars, end, virtual lines, groups, curves boundaries. Then we move to another process called the  $2\frac{1}{2}D$  sketch where the local surface orientation and discontinuities in depth and in surface orientation of the image is represented. Final we get a 3-D model representation where the image is arranged in terms of surface and volumetric primitives. In the 70's we moved beyond the block world. Brooks & Binford, 1979 proposed an idea on Generalized Cylinder and also Fischler & Elschlager, 1973 proposed an idea on Pictorial Structure. These 2 ideas are similar and tried to explain that every object can be arranged in the form of a cylinder or every object can be arranged by critical part in their elastic distance. In the 80's, David Lowe tries to recognize razors by using lines. All these above works were audacious steps. In the 90's, it was thought that since object recognition was difficult, what if we do image segmentation first which is grouping similar pixels together. One of the notable works was from [8]. In 2001,[9] published an idea on face detection, where bounding boxes were put in faces to recognize it. From the late 90's to the early 2000's, machine learning began to gain momentum. Viola & Jones work used adaboost algorithm for face detection. David Lowe published another paper, Sift & Object recognition,1999. Here he used critical features of an image to recognize another image. In the 2000's some remarkable works were done by [10] and [11]. These two papers showed that the human body is composed of parts that can be put together and also in a histogram format. As we noticed from the 50's to the 70's, 80's, 90's. The internet gained popularity and cameras became important in our daily usage. In the 2000's, we began to have benchmark dataset that can be used to measure the progress of object recognition. One of these datasets is called PASCAL Visual Object Challenge [12] which contains 20 object classes and close to 10000 images per class. So, this dataset was used by scientists and practitioners to test the performance of their algorithms. In 2009, a group of scientists [13] came together to develop a very large dataset because it has been showed that algorithm tend to overfit because of less data. The dataset was called ImageNet, it contains 22000 classes and 14 million images. Starting 2009, the ImageNet team started an international challenge called ImageNet large scale visual recognition challenge and this challenge contains 1000 object classes and 1.4 million images used to test image classification and recognition algorithms. The intuition is if an image produces 5 labels and the top-5 labels include the correct object in the image. Then we can call it a success. Below is a graph showing the error rate of image net large scale visual recognition challenge from 2012 to 2015.

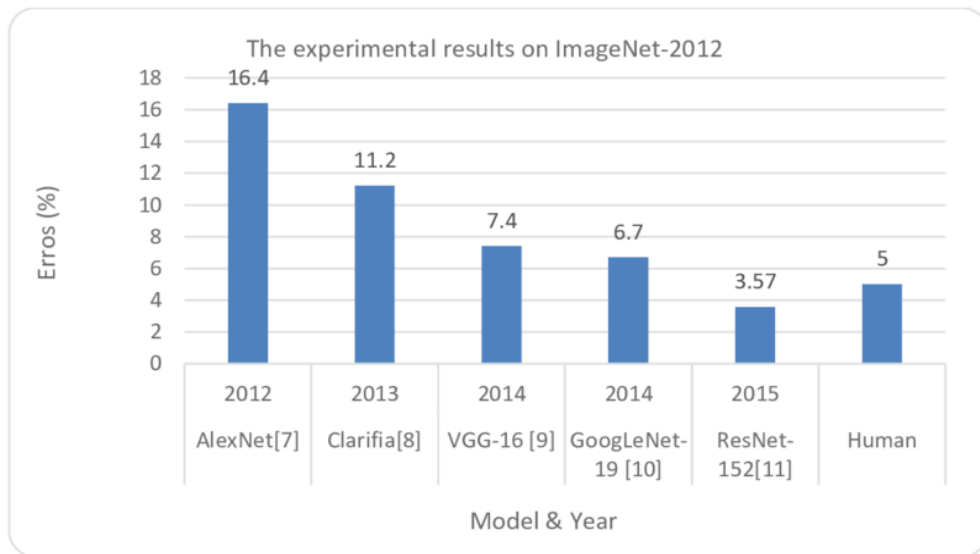


Fig 1.2: ILSVRC error rate from 2012 till 2015[14]

Notice that by 2015 the error rate has already surpassed human recognition ability which was 5%. The image net challenge was used to test for image classification, object localization and object detection. Below is a summary of image net challenge showing the top 5-errors for some years.

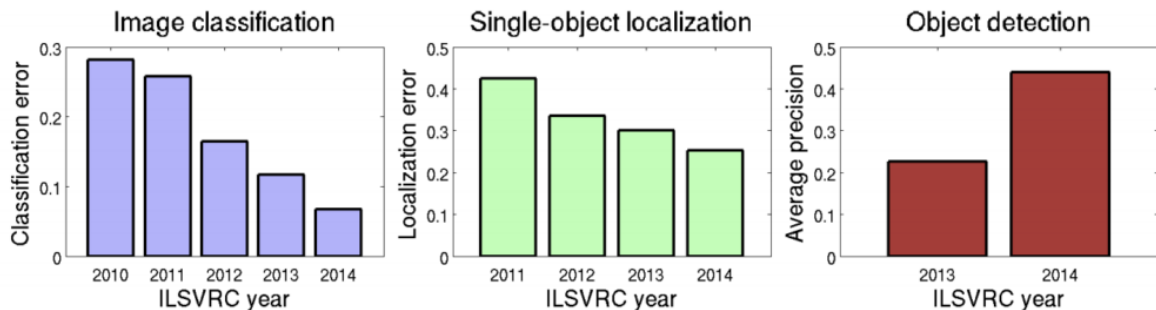


Fig 1.3: ImageNet top 5-error rate for visual recognition[15, 16]

In visual recognition, there are various task that can be done like image classification, object detection, image captioning etc. The adoption of convolutional neural network has become very important for visual recognition. Year 2012 marked the first ImageNet challenge that was won by convolutional network despite the fact that convolutional neural network was invented in 1998 by Yann Lecun[1]. One of the reasons for this is that there was an increase in the computation i.e. number of transistors in computers increased and special G.P.U were introduced to handle large datasets. This enabled researchers to work with larger models to achieve great rewards with less computations and also availability of large dataset also helps in popularizing ConvNets.

### 1.3 Research Motivation:

The motivation for this research was to compare the performance of deeper layer neural network architectures of GoogleNet and ResNet on smoke images. It is a general notion that the most straightforward way of improving performance is to increase the size and depth of the model. This comes with its challenges which might include.

1. Availability of larger datasets.
2. Deeper layer network requires larger number of parameters and are prone to over fitting.
3. Higher computational cost especially when using Fully connected layers.

But this is not the case as various advancements in deep learning have proven that deeper models can consume less computation and be more efficiency. So, the major motivations towards this research to compare two deeper layer network and to show which one is more efficient in smoke image classification.

### 1.4 Problem statement:

Some of the major reasons for carrying out this research is highlighted below:

1. Smoke can be categorized into dense smoke and transparent smoke and our network need to be robust to these categories of smoke.
2. Smoke can also be dynamic and static. Our algorithm should be robust to them
3. Analyses of smoke classification using convolutional neural networks and choosing the best method for use.

### 1.5 Limitations:

One of the limitations of this research is availability of large dataset for training the network. Comparative analysis of this network can be tedious and require large compute.

### 1.6 Research outline:

The rest of this research is outlined in various sections as follows;

Section 2: Discusses about the various works done by several researchers as regards smoke classification and detection and also using deep learning for image recognition.

Section 3: Image classification for visual recognition, linear classification, supervised learning in details, backpropagation, artificial neural network, convolutional neural network, convolutional neural networks architectures, introduction to GoogleNet-v1, Introduction to ResNet-34 and our methodology.

Section 4: Data collection and finetuning, data preparation, experimental and training details, hyper parameter settings and also testing and result.

Section 5: Conclusion

## 2. Literature Review

A lot of interesting and groundbreaking works have done in computer vision especially in image classification and object detection. The amount of visual data over the past decade have tremendously increase over the past 2 decades due to the fact that the number of sensors(smartphones) people carry around is growing at an exponential rate and these smartphones have one, 2 or more cameras in it. Virtually 70 percent of humans have smart phones with cameras and this produces lots of data from videos and images. Visual data are those bits flying around on the internet. Understanding visual data is not easy. Sometimes visual data is called the black matter of the internet. So, there is a need to devices algorithm to help us annotate and understand this data. The advent of convolutional neural network marked an era of deep learning and revolutionize image classification. Recently, more of the advancement made on image classification and object detection uses convolutional neural networks. A lot of work has also been done on smoke classification using this method.

Other researchers have carried out relevant works on smoke classification and detection. [17] studied that there is a certain feature that characterizes smoke which is viz:

1. Color or chrominance features
2. Texture (based on the dispersive distribution of smoke)
3. Edge analyses (based on the assumptions that the direction of smoke diffusion is upward, so that boundary features can be extracted)
4. Motion feature (dynamic and static smoke)

Smoke can be classified into transparent or dense depending on the intensity of burning. There are 2 types of smoke namely; static and dynamic smoke. Smoke is harder to differentiate from its disturbances because the visual characteristics of smoke like color are less trenchancy, [18]. [18] also applied block image processing and optical flow technique to extract the color and motion features of smoke and flame. [19] Used a spatiotemporal local binary pattern to verify the presence of smoke. It was seen that real scenes with an object are similar to smoke with dynamic behavior and also the low resolution of cameras, blurring and adverse weather condition can affect smoke detection so verification of true smoke is important. Their approach was edge analyses of the background object. Smoke blurs out background scene more and more. [20] used a sliding window to analyses suspected smog area and detect smoke by changing the amount of exercise. Also In [20] , He also used fast accumulative motion orientation model based on an integral image. Some Deep learning algorithms can also learn features to classify smoke in videos and also in image, convolutional neural networks are a variant of deep learning that can extract topological properties of smoke. [21] used deep CNN for smoke detection and feature extraction by using faster R-CNN, SSD and R-FCN. [22] also used

a deep convolutional neural network but this time batch normalization was incorporated. Convolutional neural network (CNN) was seen to be able to deal with high-resolution images better and batch normalization helps to solve the problem of internal covariate shift. [23] used CNN to extract the useful features of smoke using learning algorithm. He used convolution neural networks to identify fire in videos. The Convolution Neural Network directly operates on raw RGB frame without the need of the feature extraction stage, the Convolutional Neural Network automatically learns the features from the training data. Here a simple convolutional neural network was used but, in our network, we use deep convolution neural network of 22 layers-GoogleNet[4] and 34 layers-ResNet[5] which make learning feature very detailed.

[24] extracted the haar-like features of smoke and used a dual-threshold AdaBoost with staircase searching technique to classify the extracted features of smoke and finally used dynamic analyses to validate the existence of smoke. Dual threshold algorithm showed prowess compared to a single threshold algorithm in terms of convergence power and ease of training.

Most of the above researchers worked majorly on video-based smoke detection which required the study of real and synthetic videos. Videos gotten from stationary cameras were used or dataset of videos from ideal situations were used but it is very difficult because adverse weather conditions can make the cameras jitter thereby reducing the accuracy of the output. Some other researcher worked on smoke detection and features extraction using the single image (real or synthetic). Through a comparative analysis by [25], showed that the value of the smoke image was higher than that of non-smoke image.[25] classified image smoke with a dual convolutional network using dark channel prior by fusing the dark channel image features with that of the original image features of the smoke to improve performance robustness. [26] proposed a novel based method for detection and separation for the smoke. They achieved this by separating a single frame of smoke into quasi-smoke and quasi background using dual dictionaries and then use image matting to separate the true smoke from the background. Among all the visual features of smoke i.e. motion, color, texture, edges. Texture features seemed to be the most reliable because motion is subject to environment and weather condition, color is subject to the type of burning material and edges can be affected by single images not having enough background information. Due to situations of transparent smoke, extracting visual feature becomes difficult. The authors in [26] also proposed a video-based smoke detection and separation by studying the smoke opacity and smoke component which is quite different from a single image. In [27] a method of smoke component separation and feature extraction from the background was proposed but this is video-based and not image-based. [28] proposed transmission as another feature of smoke.

All the above work has proposed different methods for smoke detection but our research, we are proposing a better way for smoke recognition using the convolutional neural network architectures.



Based on the properties of smoke we notice that it is difficult to detect smoke in videos or images because

1. Foreground object occludes smoke
2. Many objects share a similar color with smoke
3. The texture and shapes of smoke vary irregularly
4. Blurred images due to low camera resolution can affect feature extraction
5. Smoke behaves unpredictably

In [29], ResNet was used as a deeper network in the generator model for generating high resolution image from low resolution image and the output was difficult to be discriminated by the critic. In [30], the author showed the various variants of skip connections and full pre-activation was better i.e. placing batch normalization layer before the non-linearity when arranging the ResNet residual block. GoogleNet was applied in [31] for extreme weather recognition and showed prowess compared to conventional methods of recognition. The GoogleNet applied in [31] was finetuned on the weather datasets to achieve better performance and accuracy. Google Net was also applied in other recognition task like visual object tracking[32], offline hand written Chinese character recognition[33], malware detection[34].

In summary, in this thesis we adopt the end-to-end learning paradigm and use of available neural network architectures for the tasks of smoke image classification, comparison and analyses of result. As a result, our models are more computationally large and require availability of large dataset, homogenous, simpler and yield superior results due to the benefits of end-to-end training and transfer learning from largescale related datasets such as ImageNet [13]. Based on this observation, in this work we would use convolutional neural networks to classify smoke images. In our approach, we will use convolutional neural networks such as GoogleNet-v1 and ResNet-34 as a powerful smoke image classification. This is the main objective of this thesis, that is, to classify images of smoke by supervised means using convolutional networks.

### 3. Methodology

#### 3.1 Image classification:

Image classification is a core task in computer vision where by a system receives some input images e.g. images of cat, frogs etc. and also the system is aware of some fixed set of category labels so the function of this system is to assign the input image(cat) to its label. Image classification might sound easy because image recognition with the eyes by humans is easy. That's because the communication between the brains and the eyes is advanced. The computer sees images differently as humans do. Machines sees image as a block of pixels ranging from  $[0, 255]$ . The task in Image Classification is to predict a single label (or a distribution over labels as shown here to indicate our confidence) for a given image. Images are 3-dimensional arrays of integers from 0 to 255, of size Width x Height x 3. The 3 represents the three-color channels Red, Green, Blue.

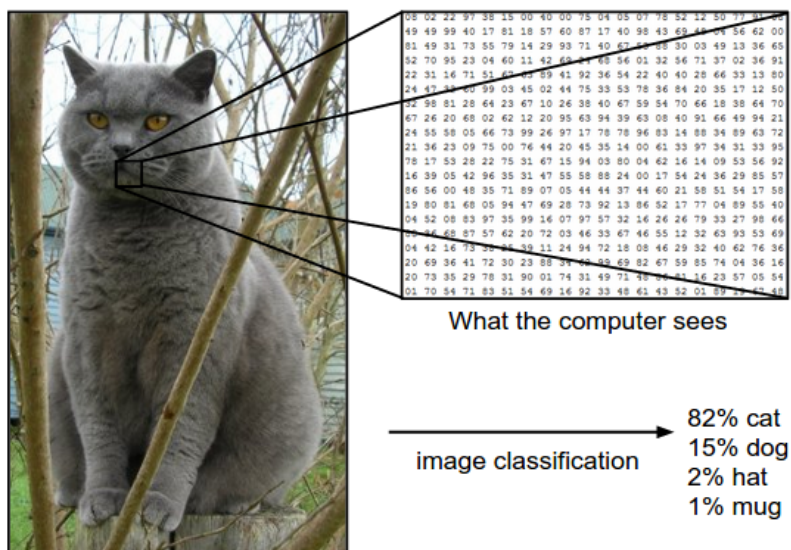


Fig 3.1 what the computer sees[16]

This is a very difficult because there is a gap between the giant cat compare to the pixels numbers the computer sees. One of these problems can occur when the camera changes its position that is the cat remains at its position. Every single pixel changes.

Another problem is degree of illumination or amount of lighting condition, this affects the pixel output. Deformation is also a challenge. Like cats position not as it is supposed. There is also a problem of occlusion where by some part of the cat image is blocked or hidden by an object. It quite easy for the eyes to detect the presence of cat but it is very difficult for the machine, so the algorithm should be robust to that. Another challenge is background clutter where by the foreground looks similar to the background. Intra-class variation is also a problem whereby the camera can't tell different shapes, size, and colors of cat. The algorithm should be robust to that too.

Image classification traces far back as 1986 where [35] used edges and corners to recognize cats. He used the fact that cats have similar patterns. This approach is quite old, because it is not a novel approach in that it can be limited to only a particular object. When we want to classify a different object, we will need to generate another pattern.

Modern approach (data driven approach):

This is as follows

1. Collect a dataset of images and labels
2. Use machine learning to train a classifier
3. Evaluate the classifier on the new images

This approach solves the above limitation by making our algorithm more robust to different images. Here we train the dataset with the labels to give us a model of what it has learnt (memory), then the model is then tested to predict images based on what it has learnt. Example is the nearest neighbor classification which classifies based on the similarity to the model it has been trained on. Nearest neighbor is not really too good because at test time classification is slow which is not the goal because we want our network to be fast at classifying at test time. These nearest neighbor algorithms which compares pixel points label by using the L1 distance which is the sum of the absolute distance between the pixels. Another form of nearest neighbor is K-nearest neighbor which groups a sum of K closest points together. This means that K-nearest neighbors classifier predicts labels based on nearest training examples. By doing so, prevents the problem of having a point inside another class. KNN helps you smoothen out your decision boundaries. K nearest neighbors is also not so good in classification. It is very slow at test time and the distance metrics on pixels are not informative especially when capturing perceptual differences between images. KNN has a number of disadvantages:

- The classifier must remember all of the training data and store it for future comparisons with the test data. This is space inefficient because datasets may easily be gigabytes in size.
- Classifying a test image is expensive since it requires a comparison to all training images.

It also using distance metrics like L1 distance and also L2 distance which takes the square root of the sum of the squares as shown mathematically below. K-nearest neighbor is just like an introduction to image classification but nowadays it's not used.

### 3.2 Linear classification:

This is a more powerful approach to image classification. It has 2 major components:

1. A score function that maps the raw data to class scores

2. And a loss function that quantifies the agreement between the predicted scores and the ground truth labels.

We will then cast this as an optimization problem in which we will minimize the loss function with respect to the parameters of the score function. The scores will be the probability of occurrence of that image. Higher score means higher likelihood and lower scores means lower likelihood.

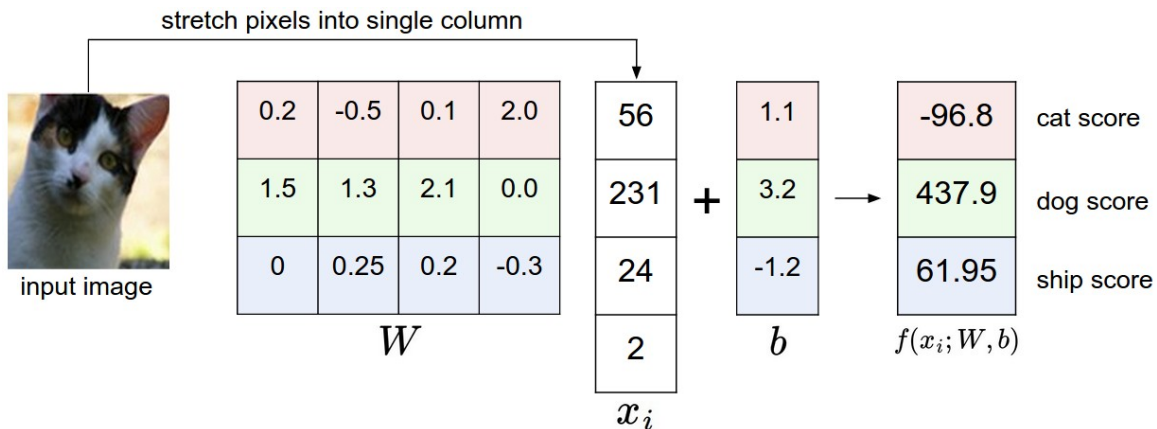


Fig 3.2: Simple linear classification network

The input image pixel of 2\*2 is stretched into a single column which is a 4\*1 matrix. Our weight matrix is 3\*4 and our bias is 3\*1. Notice that the class score is just for 3 images. Cat, dog and ship. Taking a dot product of the input image ( $x_i$ ) and the weight ( $W$ ) then adding the bias term,  $b$  gives us the individual class scores. So, when we have trained our data with this method, we can now test if the model performs well of new data. Linear classifiers are simple and easy to understand.

Image classification is a computer vision task that falls under the supervised learning category. Artificial neural network can be divided into

1. Supervised learning
2. Unsupervised learning
3. Semi-supervised learning

Supervised learning is the learning where by the network is provided with input and set of labels. These labels are a form of assignments like zeros or ones to denotes whether the images occurred or not. For instance, assuming we feed the input to a neural network with images of dogs, frogs and cats. The labels would be vectors of zeros and ones [1, 0,0], [0,1,0], [0,0,1] which signifies that in the first one hot encoding that it's a dog class by assigning 1 to dog and zero out the rest. In the second, the frog class is 1 while other classes are zero and finally in the last encodings, cats are 1 while the rest classes are zeros. The above proceeds are called one hot encoding which is the process of transforming our categorical labels into vectors of zeros and ones. Here every element must be a zero

except the element that corresponds to the actual category. Most times we do this when we are creating our own dataset but if it has been downloaded, most times open source datasets for image classifications comes with the input images and their corresponding labels so you might not need to start assigning labels to data.

So, in supervised learning we have an input and a label. Meanwhile in unsupervised learning, there is an input but no label to assign the input to. Unsupervised learning is quite complicating and not much groundbreaking research have been done. Semi supervised learning combines both supervised and unsupervised learning. It is considered a situation where we have labeled and unlabeled data e.g. pseudo labeling- a process where we take a large dataset, label some of them, train it and then take the unlabeled data and predict on it. Then we will use the prediction to label the unlabeled dataset. After labeling our unlabeled data by using the outcome of the prediction, we then train the entire data. This method helps us reduce the task of manually labeling all the datasets.

### 3.3 Supervised learning:

Many practical problems can be formulated as requiring a computer to perform a mapping  $f: X \rightarrow Y$ , where  $X$  is an input space and  $Y$  is an output space. For instance, in visual recognition  $X$  could be the space of images and  $Y$  could be the interval  $[0, 1]$  indicating the probability of a cat appearing somewhere in the image. Unfortunately, it is difficult to manually specify the function  $f$  by conventional means (e.g. it is unclear how one might write down a program that recognizes a cat. Supervised learning offers a solution to this problem by saying that relative examples  $(x, y) \in X * Y$  can be obtain for the desired mapping. In this example we collect dataset of smoke images and labels.

We assume that our training dataset of  $n$  examples  $\{(x_1, y_1) \dots (x_n, y_n)\}$  made up of independent and identically distributed (i.i.d.) samples from a data generating distribution  $D$ ; i.e.  $(x_i, y_i) \sim D$  for all  $i$ . We then think about learning the mapping  $f: X \rightarrow Y$ , by searching over a set of candidate functions and finding the one that is most consistent with the training examples. More precisely, we consider some particular class of functions  $F$  and choose a loss function  $L(\hat{y}, y)$  that measures the disagreement between a predicted label  $\hat{y}_i = f(x_i)$  for some  $f \in F$  and a true label  $y_i$ . Our objective in learning is to find  $f^* \in F$  that satisfies:

$$f^* = \arg \min_{f \in F} E_{(x,y) \sim D} L(f(x), y). \quad (1.0)$$

We seek a function  $f^*$  that can minimize the expected loss over the data generating distribution  $D$ . In practical applications, once we can identify this function, we can then discard the original training dataset and keep only the learned function  $f^*$ , which we use to map elements of  $X$  to  $Y$ .

Unfortunately, the optimization problem above is intractable because we do not have

access to all possible elements of  $D$  and therefore cannot evaluate the expectation or simplify it analytically without making unrealistically strong assumptions about the form of  $D$ ,  $L$  or  $f$ . However, under the i.i.d. assumption we can approximate the expected loss in Equation 1.0 above with sampling by averaging the loss over the available training data:

$$f^* \approx \arg \min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i). \quad (1.1)$$

So, the above means we can optimize the loss only over the training examples and this replaces the actual objective in equation 1.0 and it is now tractable. But unfortunately, optimizing equation 1.1 instead of equation 1.0 is quite challenging. For example, if we have a function that maps each training set  $x_i$  to a label  $y_i$  and returns a zero elsewhere. Using equation 1.1 will result to high losses for  $D$  that are not in the training set. Therefore, our model will not be able to generalize well to the data generating distribution  $D$  for the given samples  $(x_i, y_i) \sim D$ . And also using equation 1.1. is that different function may produce the same loss but varying generalization with test data. If all we have are the training data then how do we choose among an entire set of  $f \in F$  that all achieve the same loss in Equation 1.1? So, this can be solved by introduction a second term called regularization (R) to penalize the data loss to produce similar loss. Note that loss function tells us how good our current classifier is. The loss over a dataset is the sum of the loss over examples. Equation 1.1 can be called the data loss which tries to see if the model prediction matches training data. Due to the fact that we don't care much about training data performance. We care more on how it performs on test data. i.e. fitting training alone is not too good. So, when we add new samples and try to fit it. It becomes a problem; therefore, the major work of regularization is to help fit in new samples and it works on the assumptions that smaller weights generate simpler model and thus helps avoid overfitting. The new equation becomes

$$f^* = \arg \min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i) + R(f), \quad (1.2)$$

Regularization model should be simple so it works on test data. Even if the training set is worse. Regularization helps the network picks a better function( $f$ ). It is like a penalty to penalize.

$$\text{Total Loss} = \text{Data Loss} + \text{Regularization}. \quad (1.3)$$

So, equation 1.2, the idea of regularization is anything you do to your model that instead of trying to fit all the training data. Tries to penalize the complexity of your data by trying to force it to behave in a type of way and penalizing the loss function helps simplify the model. Example: consider a **linear regression** problem where the given dataset is

100( $n = 100$ ) 2-dimensional points ( $X = \mathbb{R}^2$ ) each represented with a scalar ( $Y = \mathbb{R}$ ). The hypothesis class  $F$  we may consider is the set of linear functions from  $X$  to  $Y$  (i.e.  $F = \{w^T x + b | w \in \mathbb{R}^2, b \in \mathbb{R}\}$ ). Here our hypothesis space has 3 parameters ( $w_1, w_2, b$ ) Where  $w = [w_1, w_2]$ . There are various commonly used regularization which are highlighted below:

1.  $L_2$  Regularization (weight decay): this penalizes the Euclidean norm. it is the squared difference between the target value and the predicted value. It is commonly used and it prevents the weight from becoming too large and affecting prediction.

$$R(w) = \sum_k \sum_l W_{k,l}^2 \quad (1.31)$$

2.  $L_1$  Regularization: this penalizes  $l_1$  norm to encourage sparsity.

$$R(w) = \sum_k \sum_l |W_{k,l}| \quad (1.32)$$

3. Elastic net ( $L_1 + L_2$ ): using both  $L_1$  and  $L_2$  Regularization combine.

$$R(w) = \sum_k \sum_l BW_{k,l}^2 + |W_{k,l}| \quad (1.33)$$

Other types are:

1. Max norm Regularization: this penalizes the max norm
2. Dropout
3. Fancier: Batch Normalization, Stochastic depth etc.

From the above example, our equation becomes:

$$f^* = \arg \min_{w,b} \left[ \frac{1}{n} \sum_{i=1}^n (w^T x_i + b - y_i)^2 \right] + [\lambda(w_1^2 + w_2^2)] \quad (1.4)$$

Where;

$\left[ \frac{1}{n} \sum_{i=1}^n (w^T x_i + b - y_i)^2 \right]$  tries to fit the training data, and  $[\lambda(w_1^2 + w_2^2)]$  is the regularization term where  $\lambda$  is used to specify the strength of the regularization and  $[(w_1^2 + w_2^2)]$  is the  $L_2$  Regularization (weight decay).  $b$  is the bias time, it is an additive property that can be ignored if we wish. It is used to ensure our model shift itself away from the origin. Regularization can help solve overfitting by discouraging complexity of our model by penalizing our loss function (different between the actual value and the predicted value) overfitting is a term we would discuss later in this write-up but for now note that an overfitted model performs well only on training data and poorly on test, or validation data which means that overfitted models cant generalize well.

### 3.3.1 Neural network regression:

Notice the above example is a simple linear regression equation but introducing that procedure to neural network regression it becomes more complex because we introduce an activation function. Activation functions are non-linear functions that helps us to straighten out the output from our different layers. It is also called transfer function. It maps the resulting values in between 0 to 1 or -1 depending upon the type of function. In neural network nonlinear network is used because it has a range unlike linear function which range from negative(-ve) infinity to positive(+ve) infinity).

Below is a representation of some the various useful activation functions in neural network:


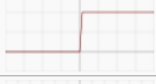

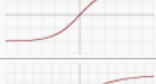

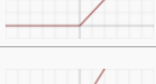

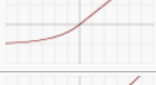

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <sup>[2]</sup>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) <sup>[3]</sup>		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Fig 3.3 Representation of various activation function in neural network[36]

So, the objective equation for a 2-layer neural network that have and activation function and input(  $x$  ) is  $f(x) = w_2 \tanh(W_1^T x + b_1) + b_2$  instead of a linear function  $f(x) = w^T x + b$  becomes:

$$f^* = \arg \min_{w_1 b_1 w_2 b_2} \left[ \frac{1}{n} \sum_{i=1}^n (w_2 \tanh(W_1^T x_1 + b_1) + b_2 - y_i)^2 \right] + [\lambda (\|W_1\|_2^2 + \|W_2\|_2^2)] \quad (1.5)$$



Where:

$\frac{1}{n} \sum_{i=1}^n (w_2 \tanh(W_1^T x_1 + b_1) + b_2 - y_i)^2$  fits the training data and  $[\lambda(\|W_1\|_2^2 + \|W_2\|_2^2)]$  is the regularization term. Where  $W_1, w_2, b_1, b_2$  are all parameters,  $W_1$  is a matrix of size  $H * 2$ ,  $b_1$  is a vector of size  $H$ ,  $w_2$  is a vector of size  $H$ , and  $b_2$  is a scalar. Here  $H$  is an integer (e.g. 100; which is the number of neurons in the hidden layer) and the hyperbolic tangent  $\tanh$  is applied elementwise as the activation function (it squashes values to the interval  $[1, 1]$ ).

### 3.3.2. Neural network classification:

Notice the example shown in fig 3.2 where we try to classify cat images. Rather than predicting scalar quality (continuous value) for each input here we try to discretize the input. Here it can be an image represented in  $N * H$  matrix while the computer sees it as a pixel matrix of rows and columns of  $2*2$  for this example as shown in fig 3.2. the output of  $f$  after dot matrix multiplication gives 3 sets of logits which is then passed through a SoftMax loss function which in turn gives a probability score between 0 to 1. SoftMax (cross entropy) loss function is a type of loss function that normalizes any input given to it from range of 0 to 1 and totally sums to one.

### 3.4 Loss function:

Loss function is a function that gives us a metric of the performance of our model in training, validation and testing data as it tells us how good our current classifier is doing over a given dataset. A function that takes in a weight ( $W$ ) and quantifies how bad or good  $W$  is, is called a loss function. The goal of every deep learning model or neural network architecture is to minimize the loss as much as possible especially on test data. Choosing of loss function depends on the specific problem but for the purpose of this thesis, we would focus on popular loss function for image classification like SoftMax for logistic regression.

There are various loss functions[37] used in deep learning but in image classification, there are two commonly used loss function namely:

1. Hinge Loss / Multi class SVM loss
2. Cross Entropy Loss/ Negative Log likelihood

#### 3.4.1 Multiclass SVM Loss:

Here we are just interested in making the score of the correct label to be greater than the incorrect label but we don't actually give those scores meaning. This loss is mostly used in support vector machines and it is easy to work with and it's nondifferentiable. The SVM "wants" a certain outcome in the sense that the outcome would yield a lower loss (which is good). The SVM loss  $L_{SVM}$  over a give dataset can be calculated as thus:

$$L_{SVM} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad (1.6)$$

Where  $s_{y_i}$  – correct score,  $s_j$  – incorrect score.

$$\begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

### 3.4.2 Cross Entropy Loss / Negative Log likelihood:

Here we want to maximize the log likelihood or for a loss function to minimize the negative log likelihood of the correct class. The goal is for the probability of the true class to be close to 1. SoftMax classifier is a type of binary logistic regression classifier. Unlike the SVM loss function that is difficult to interpret the scores, SoftMax classifier gives a slightly more intuitive output (normalized class probabilities). In the SoftMax classifier, the function mapping  $f(x_i, W) = Wx_i = s$  stays unchanged, but we now interpret these scores as the unnormalized log probabilities for each class and replace the *hinge loss* with a **cross-entropy loss** that has the form:

$$L_i = -\log P(Y = y_i | X = x_i) \quad (1.7)$$

Where:

$$P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

And  $\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$  is the SoftMax function.

Below is the data flow diagram of typical supervised learning using neural network for classification.

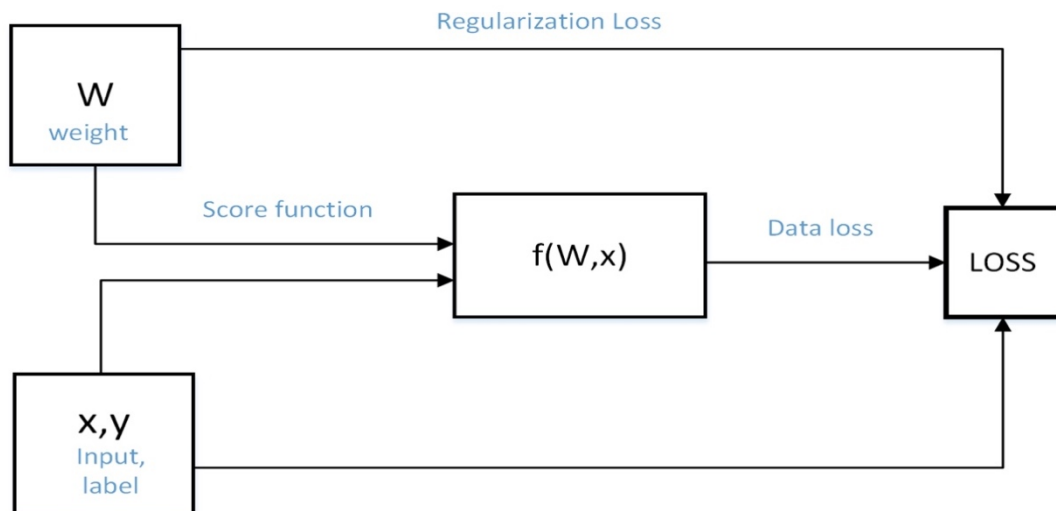


Fig 3.4: Data flow diagram of supervised learning

Here  $W$  is some weight that would be multiplied by some input  $x$  which produces some score function which is then straightened up by an activation function  $f$ , given by  $f = Wx$ .  $f$  can be any non-linearity as shown in fig 3.3.  $f$  tries to match the predicted value from  $x$  to the corresponding label  $y$ . The Data loss is the different between the predicted label value to the actual label value. Note that the regularization loss is only a function of the weight. The Regularization loss is then added to the Data which gives the total loss to evaluate how well our model is doing in fitting the data. Then total loss shows us how well our model is doing. This is the main objective of this thesis, that is, to classify images of smoke by supervised means using convolutional networks. Which means we have our datasets of smoke images and their respective labels and we try to fit the input images to their corresponding labels and then we test how well our model have learnt to correctly generalized on new dataset (synthetic and real-world datasets). Our model would be built on existing convolutional neural network as GoogleNet[4] and ResNet[5] and our result would be analyzed to show the model with the best classification error rate.

So, after calculating the total loss and our model is not doing so well. Mostly we need to adjust our parameters. Because there are millions of parameters that can be chosen from to minimize the loss. This is not an easy task especially in neural network training. Therefore, the process of finding the right Weight( $W$ ) or parameter is called **optimization**.

### 3.5 Optimization:

Optimization in deep learning enables us to minimize our objective function or error function to the barest minimum. Which means we are trying to optimize the internal learnable parameter e.g. weight( $W$ ) and Bias. Parameters are used in computing the output values and are learned and updated in the direction of the optimal solution (minimal loss). It is quite imperative to optimize this learnable parameter in order to achieve accuracy in our result. There are basically various types of optimization algorithm in neural network but for the purpose of this research we can just focus more on back propagation and gradient descent. Optimization can be categorized into numeric gradient and analytic gradient. Optimization have 3 basic strategies namely:

#### 3.5.1 Numeric gradient:

This is approximately slow and easy to write. Strategy 1 and strategy 2 falls under numeric gradient.

Strategy 1: random search - here in order to choose the best weight,  $W$ . we can do a random search and take random  $w$  and observe the behavior and our loss. This is not really a good idea because it can get very complex and time costly.

Strategy 2: using the slope estimation method - here we follow a slow pattern by adding small value of height and recomputing the loss  $w + h$ . We assume optimization is like a terrain whereby we have to move to the end. For every step we estimate the slope and add the small value to the existing weight. This is a very bad method and also slow to achieve optimum.

### 3.5.2 Analytic gradient:

This method is exact, fast and error prone but it is often used because it is better as proven by researchers and practitioners. Strategy 3 falls under this category.

Strategy 3: this entails the use of calculus to calculate the change in weight  $\Delta w$  from each step and so on. Once we know how to compute our gradient, we can now apply **gradient descent algorithm**.

### 3.5.3 Vanilla gradient descent:

Gradient descent is the sequence of steps taken in computing the gradient of the loss function and updating the parameters. Gradient descent is very commonly used[38],[39] in neural network optimization. Here we multiply a `step_size` by the weight gradient and add it to the original weight which then gives us the new weight. Below is a simple algorithm of vanilla gradient descent.

#### # Vanilla Gradient Descent Algorithm:

while True:

```
weights_grad = evaluate_gradient(loss_fun, data, weights)
weights += - step_size * weights_grad           # update parameter
```

`Step_size` is a hyper-parameter called learning rate which may vary between 0.1 to 0.0001 depending on the scenario. It is a critical parameter, if it is too high the optimization may not converge or even diverge. If it is set too low, learning will take too long. Hyper-parameters are usually set prior to training, they can be occasionally considered as manual parameters because it is specified manually. The total loss for training with gradient descent algorithm is given as

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(x_i, y_i, W) + \lambda R(W) \quad (1.8)$$

The above is the combination of data loss and regularization loss for N number of datasets. N can be too large depending on the dataset which would take long time to update. This becomes a problem until a better type gradient descent called **stochastic gradient descent** was introduced.

### 3.5.4 Stochastic gradient descent (SDG):

This is also an optimization technique that instead of calculating the gradient over an entire dataset, does a stochastic approximation of gradient by calculating for randomly selected subset of the dataset thereby reducing the complexity of N. Stochastic gradient is definitely better than gradient descent because here N is a minibatch of maybe 32,64,128. When dealing with large datasets like ImageNet datasets, SGD is advisable because it is wasteful to calculate the full loss function over the training set in order to perform a single parameter update as done in vanilla gradient descent rather the losses are computed in batches of training data. This batch is then used to perform a parameter update.

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_W L_i(x_i, y_1, W) + \lambda \nabla_W R(W) \quad (1.9)$$

The algorithm for stochastic gradient descent is shown below:

*#Vanilla Minibatch Stochastic Gradient Descent:*

while True:

```

    data_batch = sample_training_data(data,256)           # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad.               # update parameter

```

Note that some other techniques can be used to improve the gradient descent algorithm. For instance, one of the techniques is Momentum update, which tries to ensure that the gradient makes small consistent steps in right direction. Others are Adagrad[40], Adam optimizer[41], RMSProp[42] and Nesterov Momentum[43] etc. can also be used with gradient descent.

### 3.6 Backpropagation:

From our previous discussions we have seen that the best way to get the perfect weight(W) is by performing optimization and we also know that we can evaluate the gradient of network and use it to update the parameter of our model, we also saw we can use SGD to minimize our loss function. Now the question is how can we obtain our gradient?

Back propagation is a term used to refer to calculating the gradients in a network backwards to the beginning. The gradients are calculated with respect to the corresponding inputs and the weights are then updated. Note that a single pass of data through a model is called an epoch. Then the loss is computed as discussed earlier and the gradient which is now the recursive application of chain rule from calculus. This

gradient is then multiplied by the learning rate and then the weights is updated. Therefore, we can conclude that one forward pass of data through a feed forward neural network gives the loss( $L$ ) and one backward pass through the same network gives the gradient  $\frac{\partial L}{\partial z}$  where  $z$  is the total loss. The backward pass is what we refer to as backpropagation and for a network to learn it need to go through the forward and backward pass recursively. Back propagation algorithm was first introduced 1986 by[44], where they discovered that neural network can optimize itself through finding the gradient at each node and replacing it with the previous weight. Update is what we mean by the network is learning i.e. the value to assign to each weight to make it closer to the output label making the loss function smaller. Below is the representation of back propagation using a computational graph. A computational graph is a graph that show the mathematical flow of information from the start to the finish in a neural network.

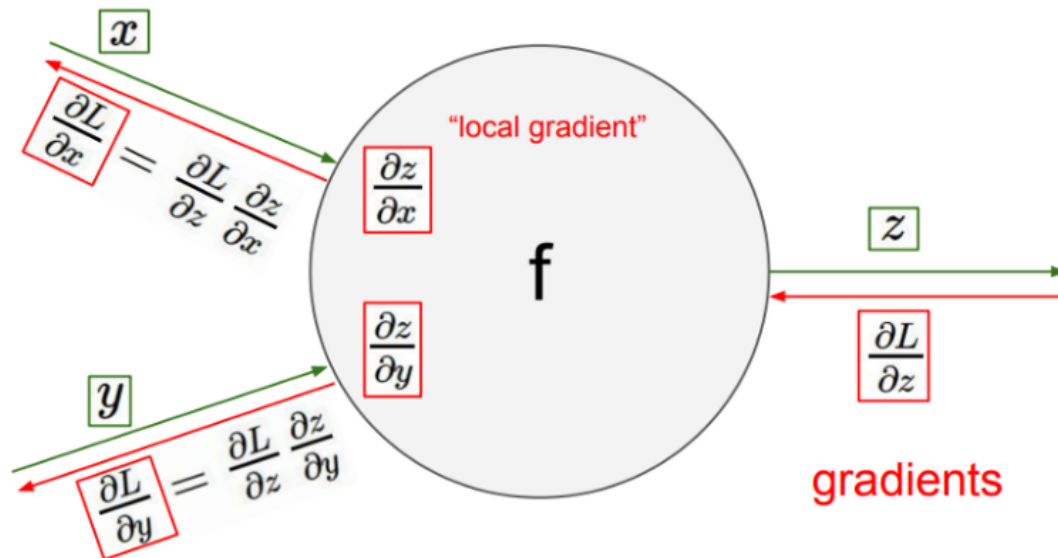


Fig 3.5: Simple back propagation for a single neuron[45].

The above representation is what happens for a single neuron. The output  $Z$  is the forward pass total loss,  $\frac{\partial L}{\partial z}$  is the gradient with respect to the total loss as going back towards the opposite direction.  $\frac{\partial z}{\partial x}$  and  $\frac{\partial z}{\partial y}$  is the local gradient computed and stored in memory for later calculation. It is the gradient with respect to the activation function. This local gradient is then used to calculate the gradient at each weight  $x$  and  $y$  by chain rule of the local gradient multiplied by the gradient with respect to output  $Z$ . This gradient value outcome is then multiplied by the learning rate and used to replace the initial weight and it repeats itself until the least loss is achieved. The above is for scalar valued operation but for vectorized operation like matrices, use Jacobian matrix to calculate the gradient. Below is a simple example of scalar valued backpropagation process:

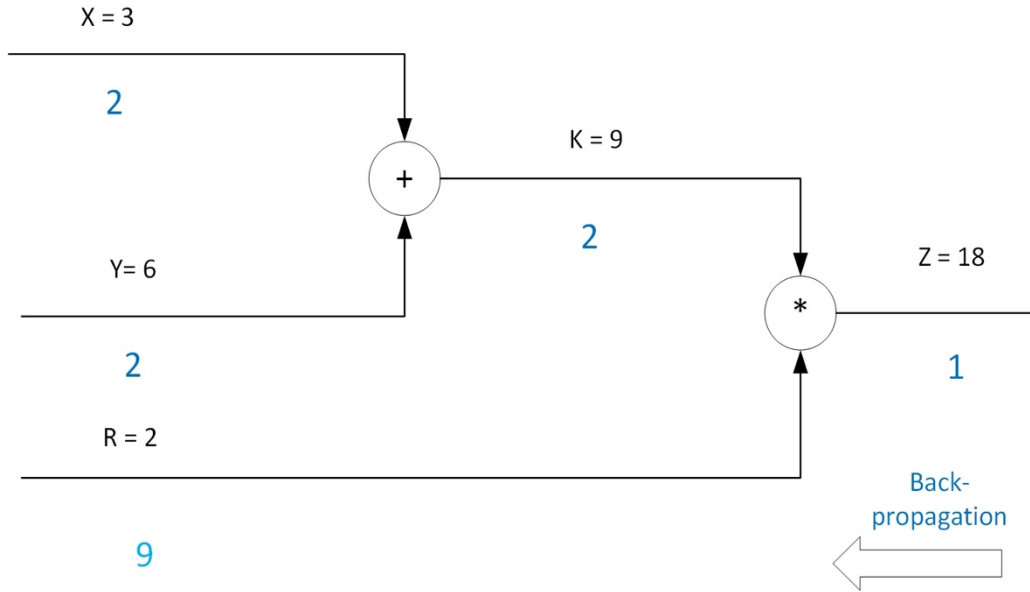


Fig 3.6: A simple Computational circuit of backpropagation

Note that the local gradients are the inputs with respect to its output i.e. we have multiplicative gate and also additive gate from the above computational circuit. The local gradient for the additive gate is +1 because of its additive property but the local gradient of the multiplicative gate is 2 and 9. The forward pass gives a loss of 18. From that point the gradient with respect to the loss  $z$  is 1 i.e.  $\frac{\partial z}{\partial z} = 1$ . Then the gradient with respect to  $k$  is 2. i.e.

$$[\text{local gradient}] * [\text{upstream gradient}] \quad (2.0)$$

Where the local gradient is 2 and the upstream gradient is 1. It can also be expressed as  $\frac{\partial z}{\partial k} = 2$  for  $k$  and  $\frac{\partial z}{\partial R} = 9$  for  $R$ . For  $y$  and  $x$ . the local gradient is 1 because of the additive property. Therefore, using chain rule:

$$\begin{aligned} \text{for } y, \quad \frac{\partial z}{\partial y} &= \frac{\partial z}{\partial k} \frac{\partial k}{\partial y} = 2 * 1 \\ &= 2 \\ \text{and for } x, \quad \frac{\partial z}{\partial x} &= \frac{\partial z}{\partial k} \frac{\partial k}{\partial x} = 2 * 1 \\ &= 2 \end{aligned}$$

**Summary:**

Earlier, we discussed image classification as a form supervised learning. We drew the conclusion that supervised learning is assigning or fitting training input to the corresponding label and trying to use that learnt model to generalize on new data. In the case of image classification, the input becomes the image with its corresponding label.

We explain neural network regression and classification where we introduced the concept of loss function which is a function to know for sure how well our model is doing and also activation function for neural network classification where  $f$  can be non-linearity that can squashes our input to the required output probably 0 or 1 depending on which non-linearity was chosen.

We also discussed regularization  $R(W)$  as a medium to help us penalize our weight which we then add to our equation to give the total loss. We discussed the various optimization process like gradient descent as an iterative process and the very popular stochastic gradient descent which divides N number of examples into minibatch for easier computation and efficiency. So, for a network to fully learn we need a perfect weight that can make our loss very low and to do so we introduced a process called back propagation.

We have seen that backpropagation is a recursive process. Before backpropagation algorithm can occur, there must be a forward pass to get the loss. Then the gradient backwards to the origin is called backpropagation. This way we can get the perfect weight that reduces the loss to the barest minimum.

### 3.7 Artificial Neural network:

The human nervous system contains approximately 86 billion neurons. The brain contains lots for neurons used for basic and advance computations. Artificial neural network is modeling the biological neural network of the brain in a way that it can perform similar computation as the brain. The neuron of the brain is connected by synapses. Each neuron receives input signals from its dendrites and output a signal that passes through its single axon and then the axon carries the information to the dendrites of another neuron through synaptic connection. When we model the similarities of the biological neuron mathematically. We observe that the signal is the input  $x_i$  and the interaction between the input to another dendrite is based on the synaptic strength  $w_i$  and it is multiplicative i.e.  $w_i x_i$ . The synaptic strength ( $w_i$ ) are learnable, the dendrite carries the signal to the cell body where they are summed. So, the summing up happens in the cell body and if the final sum is above a certain threshold the neuron can fire a spike along its axon. What determines the firing rate of the neuron is the activation function  $f$ . We have discussed activation function earlier as non-linearity that can lead to firing of neuron based on the frequency rate. Below is a diagram of a biological like neuron and an artificial neuron.



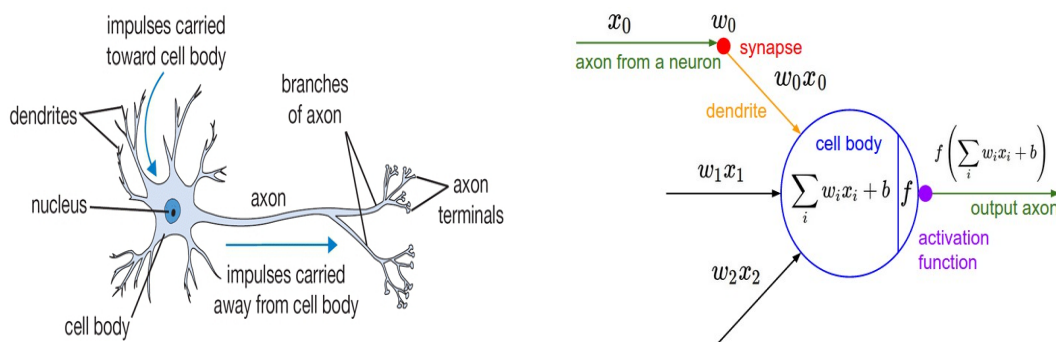


Fig 3.7 biological neuron and its mathematical model[46]

The above represent the movement of signals with synaptic strength to the cell body where it is summed up and multiplied by an activation function. The output from the axon becomes  $f(\sum_i w_i x_i + b)$ .

Artificial neural network can be arranged from a single layer neural network to a multiple deep neural network with one or more hidden layer. When a neural network contains multiple hidden layers, it becomes deeper which introduces the concept of deep learning. Deep learning is a subset of machine learning whereby we require a deeper layer and larger datasets for learning to take place. Below is a diagram of a 3 -layer neural network with the one input containing 3 inputs, 2 hidden layers containing 4 nodes each and 1 output layer;

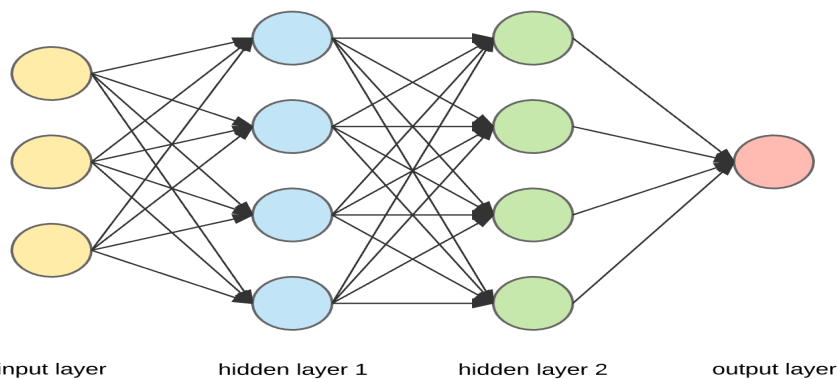


Fig 3.8: A 3-layer neural network or a 2 hidden layer neural network[47].

Assuming from the above diagram, we choose a rectifier linear unit (Relu) activation function which characteristics behavior is similar to biological neuron and is zero piecewise linear i.e.  $f(x) = \max(0, x)$  as shown in fig. 3.3. then our output (a) for a 3-layer neural network will look like thus:

$$a = w_3 \max(0, w_2 \max(0, w_1 x)) \quad (2.1)$$

### 3.7.1 Convolutional Neural Networks (CNNs):

Convolutional neural networks was first used in [48] for digit recognition. Convnets are a class of deep neural network architectures that are commonly used for visual recognition like image classification, text recognition, object detection, image segmentation, face recognition, video classification, medical image, galaxy classification, street sign recognition etc. For the purpose of this research convolutional neural network will be used for smoke image classification. Instead of performing the archaic linear classification here will use convolutional filters to extraction activation maps which are termed feature extraction. This extraction of filters helps the network to learn to classify images based on the learnt features. The way convolutional neural networks are connected resembles the way neurons are connected in the visual cortex of our brain. The term convolution is a mathematical operation that the network performs between 2 signals. Therefore, we can conclude that convnets is a sequence of convolutional layers interspaced with activation functions. It consists of the input, hidden and the output layer. When the input (tensor) with shape (number of image)\* (image width)\* (image height)\*(image depth) is passed through a convolutional layer, the images is transformed to a feature map (activation map) with shape (number of images) \* (feature map width) \* (feature map height)\* (feature map channels). This feature map is then moved to the next layer and the same process of convolution takes place. In training convolutional networks with reference datasets, it still follows the feedforward and backpropagation process to help reduce the problem of vanishing gradient and exploding gradient. Vanishing gradient can occur when weight  $w$  becomes smaller and smaller until it vanishes and exploding gradient is when weight  $w$  becomes bigger and bigger until it explodes. A lot of factors can lead to vanishing or exploding gradient which might be discuss later as we progress. The convolutional layer within a neural network should have the following attributes:

1. Convolutional kernels defined by a width and height (hyperparameters).
2. The number of input channels and the output channels (hyper parameter).
3. The depth of the convolutional filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Below is 2-layer convolutional neural network:

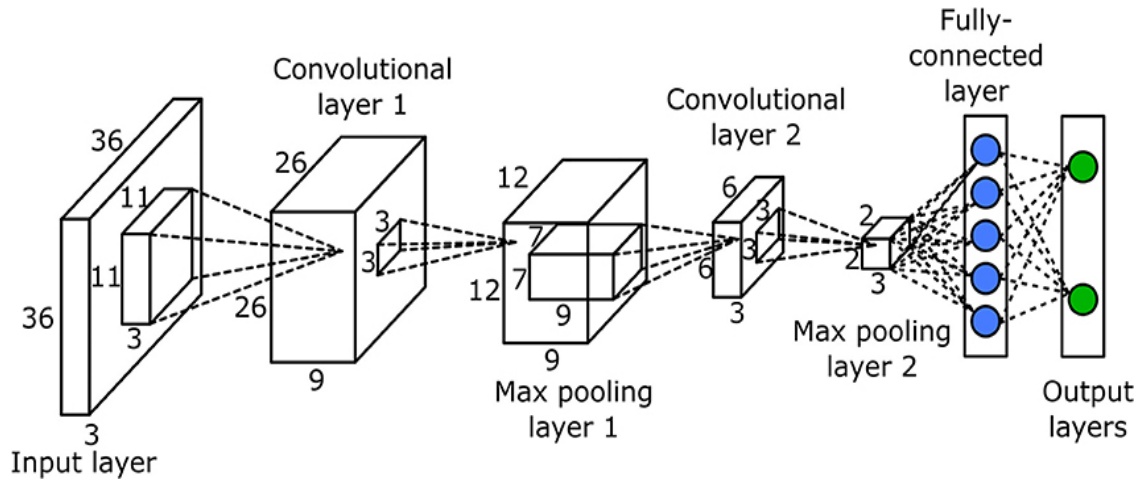


Fig 3.9: A simple Convolutional neural network architecture

From the above we noticed that the input layer contains an image input of dimension  $W * H * D$  ( $36*36*3$ ) and the convolutional filter or kernel of  $W * H * D$  ( $11*11*3$ ) slides through the input. The number of filters in the conv1 (K) layer that slides on the input becomes the depth of the output which is ( $26*26*9$ ) and so on. By sliding the filters around the input means performing a convolution operation and the output is called an activation map or feature map which becomes input to another layer. The max pool layer is more of a down sampling layer. It partitions the input image into a set of non-overlapping rectangles and for each sub region outputs its maximum. Pooling layers can be max pooling or average pooling. Max pooling takes the maximum while average pooling takes the average of the convolved space. It should be noted that the depth of the tensors after undergoing pooling is not affected. Below is a pictorial view of  $2*2$  pooling on an input  $4*4$  tensor.

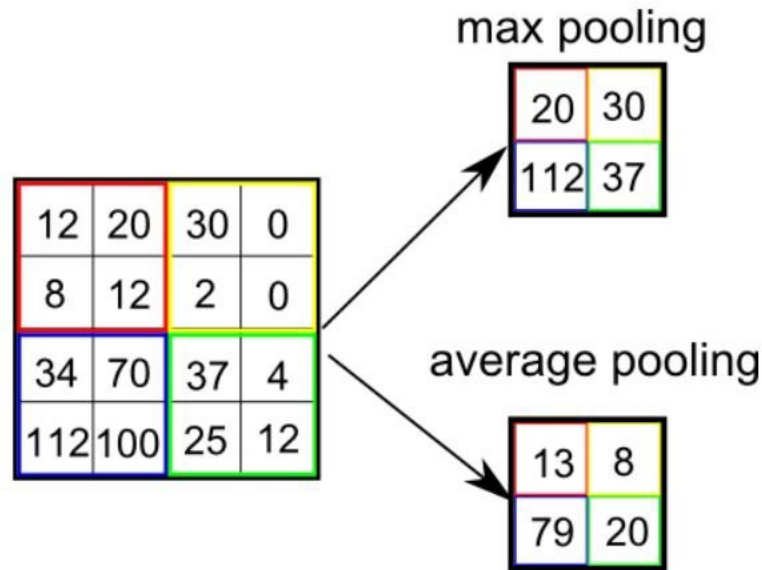


Fig 3.10: Pictorial representation of max pooling and average pooling[49].

Padding adds zeros around the borders of the input in case so we can retain the dimension of the output. Now the output becomes smaller and smaller once a convolutional filter slide around it. This might not be too good for us as some feature maps are lost around the edges. Zero padding solves this by allowing us preserve the original input size by adding zero along the borders of the input. The good thing is that most neural network API can automatically specify the size of the border for us. We just have to specify whether we want padding or not.



Fig 3.11: When Zero padding =1.

The above diagram shows how zero padding is represented as blocks of zeros around the borders of the rows and columns of the input tensor. These extra pixels help the output retain its spatial features after convolution.

Mathematically, the output of each layer in a convolutional neural network can be obtained using the formula;

$$OUTPUT\ SIZE = \frac{N-F}{S} + 1 \quad (2.2)$$

Where  $N$  is the width or height size,  $F$  is filter spatial extent,  $S$  is the number of strides which was applied.

Generally, a convolutional layer contains 4 hyper parameters: The number of filters  $K$ , their spatial extent  $F$ , the stride with which they are applied  $S$ , and the amount of zero padding of the input  $P$ .

The number of output volume size is  $W_2 * H_2 * D_2$  where  $W_2 * H_2 * D_2$  are outputs when the filter convolves the input.

The number of parameters in each filter is  $(F * F * D_1) * K$  weight and  $K$  bias.

We can illustrate the above by using the example in fig 3.9. Consider a  $36*36*3$  input that 9 convolutional filter of size  $11*11*3$  with stride 1 pad 0 convolves around, the output volume would be  $26*26*9 = 6084$  and the number of parameters(weight and biases) for the filter would be  $11*11*3*9 + 9 = 3276$  parameter output for the first layer.

### 3.7.2 Convolutional Neural Network Architectures:

A convolutional neural network architecture is built by stacking convolutional layers and introducing pooling layers, Relu non linearity, Batch normalization[50] to control the computational complexity of the network. There are several convolutional neural network architectures used in deep learning like [1],[2],[3],[4],[6],[5] especially in image classification but for the purpose of these research we are going to restrict our attention to two (2) architecture for our smoke image classification namely:

1. GoogleNet
2. ResNet

### 3.8 GoogleNet:

GoogleNet was developed by Google and it draws inspirations from the older network called LeNET[48]. It was the ILSVRC 2014 winner where it showed relatively lower error rate (top-5 error is 6.67%) compared with VggNet which was the 1st runner-up in 2014 (top-5 error is 7.4%). Its main contribution was development of an inception module that reduced the number of parameters in the network from 60million in AlexNet to 4million using inception module ( $V_1, V_2, V_3, V_4$ ). The name inception came to light because the author felt there was a need to go deeper in building deep neural network for deep learning task. It also uses average pooling instead of fully connected layers at the end of the convNet to reduce number of parameters. GoogleNet has deeper layers (22-layers) with computational efficiency. GoogleNet draws its inspiration from Network in Network[51], where global average pooling and  $1*1$  convolution kernel was first used in their network.

### 3.8.1 Vanilla Inception module:

The naïve inception module comprises of four convolutional filters which comprises of  $1 \times 1$  conv filter,  $3 \times 3$  conv filter,  $5 \times 5$  conv filter and  $3 \times 3$  max pool. The output from the previous layer passes through this filter and the information or features extracted or activation maps are then concatenated. It is noticed that this naïve inception module can be computationally complex as the depth becomes 672 as shown in fig 3.93. So, the depth is blown up as shown below:

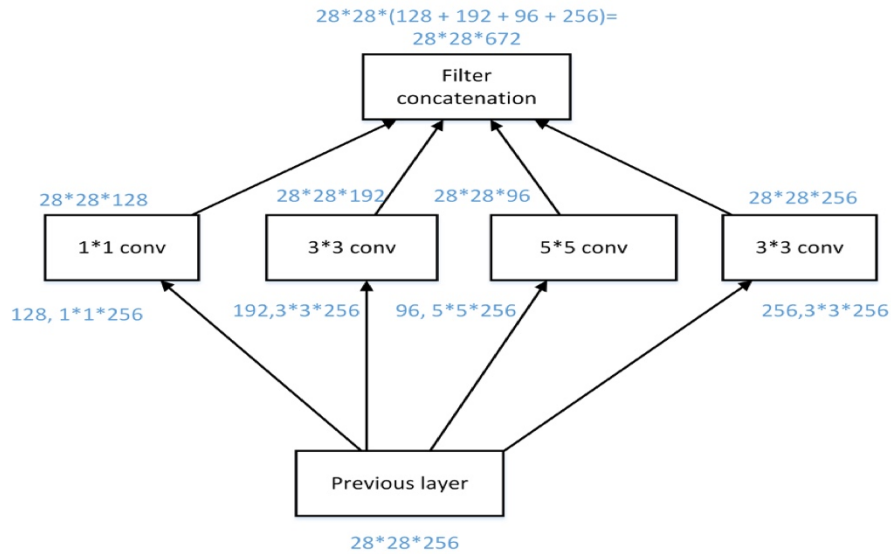


Fig 3.12: Vanilla inception module.

### 3.8.2 GoogleNet Inception Module:

To solve the depth blown up problem in the vanilla inception module, a bottleneck was introduced by Google as a solution by using  $1 \times 1$  convolutional filter before any higher dimensional filter. So,  $1 \times 1$  conv filter was applied before  $3 \times 3$  conv filter,  $5 \times 5$  conv filter and after  $3 \times 3$  Max pooling for depth reduction and then concatenating depth wise dimensions. We noticed a reduce from 672 to 480 when the channel is concatenated and similar patterns was used around other inception modules in the 22-layer network.  $1 \times 1$  convolution kernel is used as a dimension reduction module to reduce the computation and the feature depth.  $1 \times 1$  convolution can help to reduce model size which can also somehow help to reduce the overfitting problem which is the problem of the naïve inception module and increases efficiency. A GoogleNet inception module is shown in the figure below:

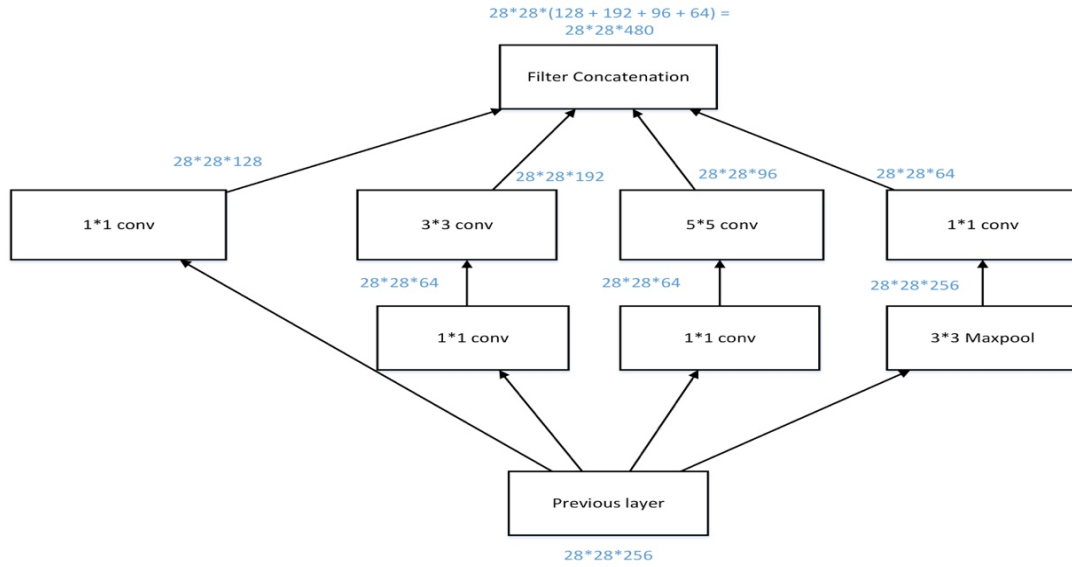


Fig 3.13: GoogleNet inception module.

The Max pooling layer is added to summarize the content of the previous layer. All the results are concatenated one after the other and given to the next layer. From the above the 3\*3 Max pool will produce the same 28\*28\*256 because it was given a padding of 1(padding=same) and stride 1 and then 1\*1 conv is also added as a bottleneck to reduce the depth from the output from the Max pooling. The role of the 1\*1 conv can be illustrated in the example where 1\*1 conv is added and where it is not added below;

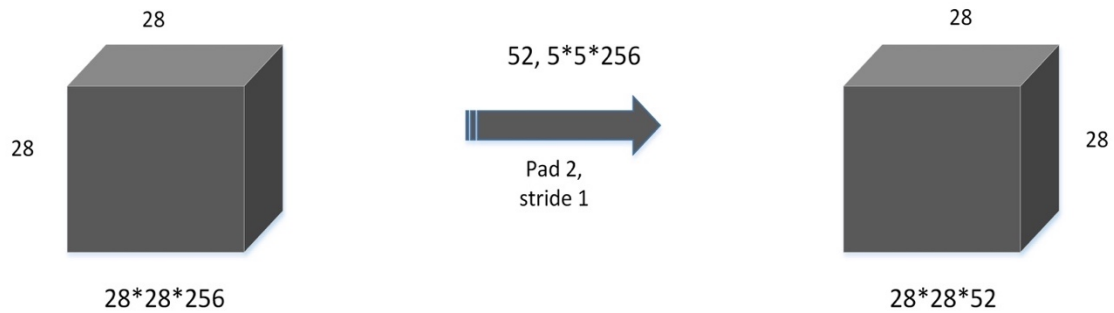


Fig 3.14: Naïve convolution without 1\*1 conv bottleneck.

Here when 52 filters of 5\*5 conv filter with pad 2 and stride 1 convolves with an input of 28\*28\*256, the number of operations can be written as;

$$\begin{aligned} \text{Number of operations} &= (28*28*52) *(5*5*256) \\ &=260\text{M} \end{aligned}$$

When we add 1\*1 conv bottleneck, we get

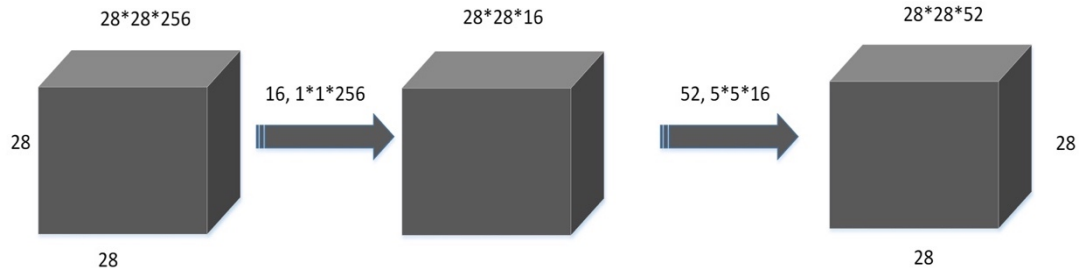


Fig 3.15: Introducing a bottle neck by adding 1\*1 conv filter.

When 16 conv filters of 1\*1 convolves round an input of 28\*28\*256, it results to similarly dimension for the width and height but a reduction in the depth. Then the output becomes input to 52 conv filters of 5\*5 dimension. The total number of operations here becomes the number of operations from the first part and that of the second part which is given as;

$$\text{Number of operations for } 1*1 = (28*28*16) * (1*1*256) = 3.2\text{M}$$

$$\text{Number of operations for } 5*5 = (5*5*16) * (28*28*52) = 16\text{M}$$

$$\text{Total number of operations} = 3.2\text{M} + 16\text{M} = 19.2\text{M}$$

When we compare fig 3.96 and fig 3.95, we would notice that there is a great reduction of operations from 260 million to 19.2 million just by introducing 1\*1 conv filter. There reduces overfitting by reduces the model size which means that the lesser the number of parameters the lesser it is prone to overfit so the role of inception layers is to avoid parameter explosion. A 22- layer GoogleNet has 9 inception layers.

### 3.8.3 Global average pooling:

This is used in GoogleNet architecture at the end of the network rather fully connected layers. Global average pooling was first used in Network in Network [51] where it was shown to be more robust to spatial translation of the input. It was also shown in [4] that there is no parameter to optimize by using global average pooling so overfitting is avoided. Global average pooling takes the average of each feature map and the resulting vector is fed directly to the SoftMax layer. The idea is to generate on feature map for each corresponding category of the classification in the last inception layer i.e. pooling the average value of each single feature map thereby drastically reduces the dimension of the output (thereby reducing overfitting because the more the parameters the more our network is prone to over fit). So, there is no parameter to train. It was found in [4] that moving from fully connected layer to global average pooling improved the top-1 accuracy



by about 0.6%. So, in the network global average pooling averages each feature map from 7\*7 to 1\*1 as show below:

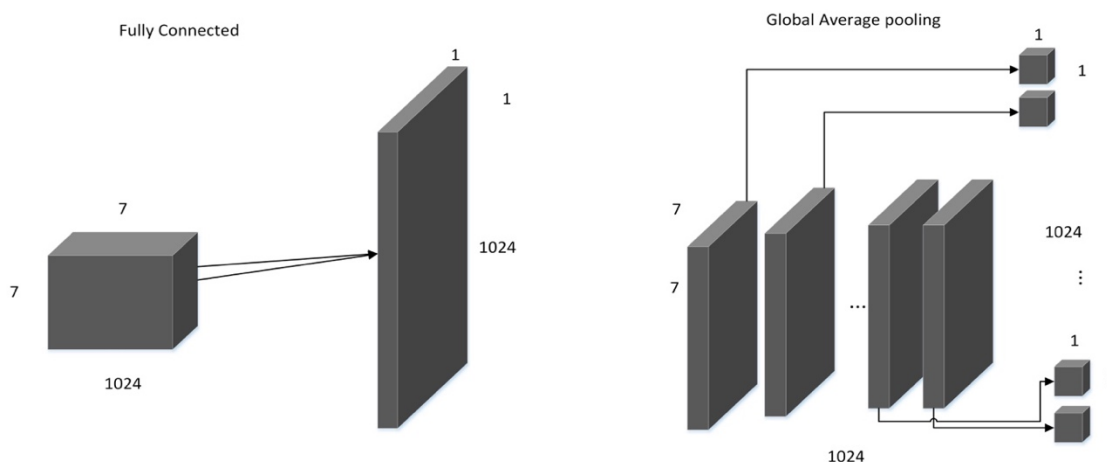


Fig 3.16: Fully Connected layer (FC) vs Global Average Pooling.

Number of weights for the fully connected network =  $7*7*1024*1024 = 51.3M$

Number of weights for the global average pooling = 0

From the above we noticed that there is no parameter to be optimized and this avoids overfitting. Also, a fully connected layer with 1024 units and rectified linear activation and dropout layer[52] of 70% ratio is necessary before the SoftMax activation for prediction.

### 3.8.4 The GoogleNet-v1 Architecture:

The overall architecture contains 22-layers with 9 inception layers and 2 auxiliary layers. There are 4, 3\*3 Max pool before and after the normalization layer and after depth concatenation to change the height and width. The auxiliary layer also known as side branches takes some hidden layer and tries to use it to make some predictions (predicting output label). This layer helps to ensure that the features computed even at the intermediate layers are not too bad for predicting the output course of the image. Another advantage is that it has a regularizing effect on the network inception thereby preventing overfitting and also prevent vanishing gradient effect. It takes output from the depth concatenated area and apply average pooling, then use 1\*1 conv, the FC layer and the SoftMax for classification, and it is only used at training time and not at test time. Below is a diagram of the overall architecture:

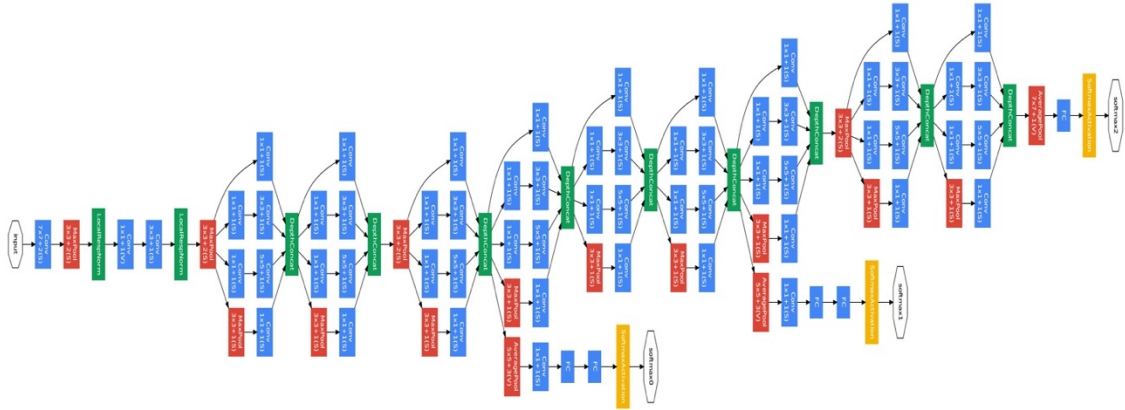


Fig 3.17: GoogleNet Architecture[4].

The first GoogleNet architecture was called GoogleNet Inception V1, and then later GoogleNet Inception V2, V3 and V4 was released. GoogleNet is deeper than previous Convolutional Neural Networks before it like LeNet, AlexNet, VggNet but is still not so deep compared to later model like Deep Residual Network (ResNet) with 152-layers which was invented in 2015 by Kaiming He and this group at Microsoft.

### 3.9 Residual Neural Network (ResNet):

It was initially assumed that training a very deep neural network is difficult but this is not the case for ResNet and other convNets like Vgg19, GoogleNet. 2015 was a ground breaking period for deep learning society by presenting a network with the deepest layers and also achieve 3.37% top-5 error at ILSVRC which is below the detection and classification error seen in humans which is about 5% top-5 error. The major problem of deep neural network was vanishing gradients and cause of dimensionality. This can be solved by learning a function called identity function. Residual neural network makes it feasible to train deeper layers up to hundreds and still achieve better performance and efficiency which nullifies the initial assumptions. From the 1998 LeNet which had 7 layers to AlexNet which had 8 layers, Vgg19 with 19-layers, GoogleNet had 22-layers. We noticed there is an increasing rate of the depth of convolutional neural network architectures. This increasing depth does not work by stacking layers together alone because by doing so alone can lead to overfitting. There are certain tricks that was applied for example for GoogleNet as discussed earlier where inception layers with bottlenecks were implemented. ResNet achieved the great feat by introducing a skip or short connections between the layers and this greatly improved image classification,

localization and object detection in computer vision. This skip connection helped to reduce or combat the vanishing gradient problem as seen in deeper neural networks. Vanishing gradient is a term used to refer to a problem where by the gradient as the network backpropagates tends toward zero and then vanishes, so at that point no learning can occur and the network starts degrading. Below is the visualization of what happens when we keep stacking deeper layers on a plain convolutional neural network.

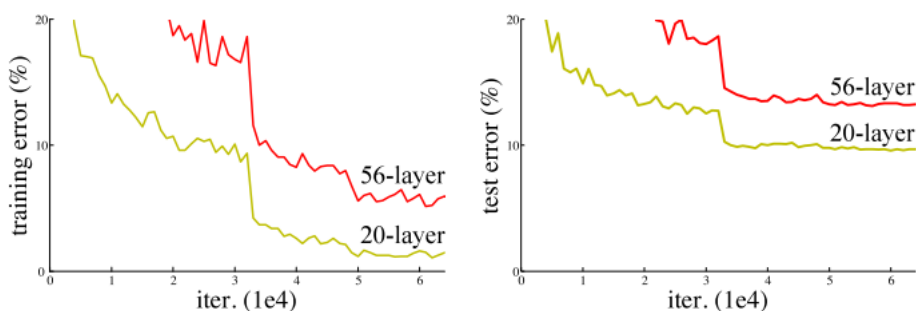


Fig 3.18: Stacking layers on a plain network[5].

From the above we noticed that for a 56-layer network, there is higher training error and also higher test errors when compared to smaller 20-layer network. This higher error is not due to overfitting because overfitting only occurs when the training error is low and the test error is high but in this case both the training and the test error are both higher. This problem is called degradation problem. So, there is a need to optimize a deeper network to achieve greater accuracy than a smaller network. The intuition behind the popular ResNet is that deeper model should be as good as shallow model and even better. So, the solution to the above problem as shown in fig 3.99 is copying learned layers from shallower layers and setting additional layers to identity mapping. This idea means introducing skip connections. Below is a simple diagram illustrating skip connection for ResNet:

Assume:

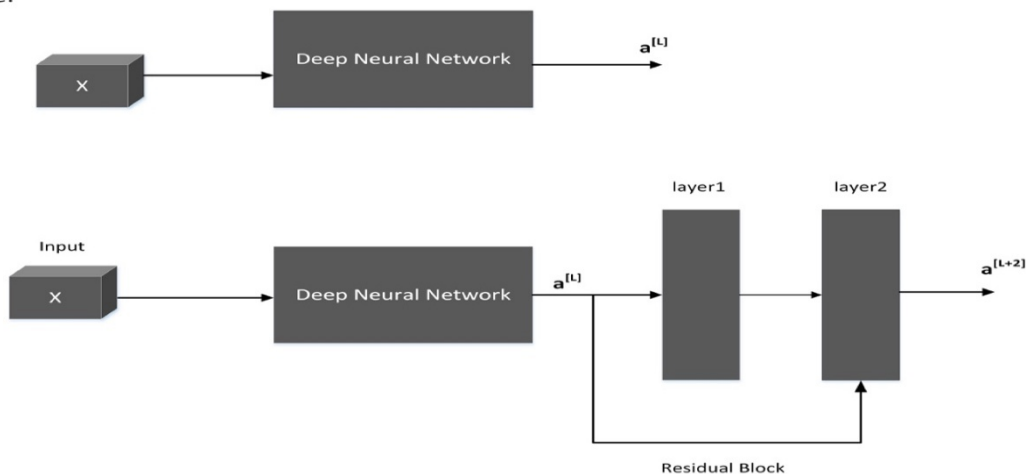


Fig 3.19: A simple illustration of skip connection in a deep network.

From the above diagram, it can deduce that

$$a^{[L+2]} = f(Z^{[L+2]} + a^{[L]}) \quad (2.3)$$

where  $a^{[L]}$  is the skip or short connections from stacked convolutional filters,  $a^{[L+2]}$  is the output of the network and  $f$  is an activation function (Relu). Note that for  $(Z^{[L+2]} + a^{[L]})$  to be a valid addition it must be of the same dimension i.e. it must be of the same convolutions which will make  $a^{[L]}$  and  $a^{[L+2]}$  to be of the same dimension. From equation 2.3 we can say that;

$$a^{[L+2]} = f(W^{[L+2]}a^{[L+1]} + b^{L+2} + a^{[L]}) \quad (2.4)$$

Where  $Z^{[L+2]}$  is  $W^{[L+2]}a^{[L+1]}$  which is the weight times the input plus bias term. Assuming weight decay or  $L_2$  Regularization is applied to  $W^{[L+2]}$  and  $b^{L+2}$  and it becomes zero. The equation becomes

$$a^{[L+2]} = f(a^{[L]}) \quad (2.5)$$

We know Relu function cannot be negative, so the final equation becomes

$$a^{[L+2]} = a^{[L]} \quad (2.6)$$

From equation 2.6 we can conclude that the output of the shallower layer is become equals the output of the deeper layer because of the skip connections. The identity function is easy for the residual block to learn which makes it easy to make  $a^{[L+2]} = a^{[L]}$  because of the skip connections. What this means is that adding extra layers do not affect the network and it is observe that  $a^{[L+2]}$  can do as well as  $a^{[L]}$ . The above is just a simple illustration of what skip connections means so it can be applied to deeper layer in ResNet. Therefore, it would be easy to just learn the identity function or just copy  $a^{[L]}$  to  $a^{[L+2]}$ . So, this is why adding extra layers or block do not hurt the network performance. If we use ordinary ResNet without skip connection it would be difficult to choose parameters that can learn. ResNet works because it is easy for extra layer to learn the identity function and during back propagation gradient flow easily. Skipping during training is important because we don't know the optimal layers necessary for the network which might depend on the complexity of the dataset. So instead of treating number of layers as a hyperparameter, adding skip connection enables our network skip training for the layers that are not important and do not add value to the overall accuracy.

ResNet can go as deep as 34 layers to 152 layers. There are others like ResNet-12, ResNET-18, ResNet-34, ResNet-50, ResNet-101 etc.

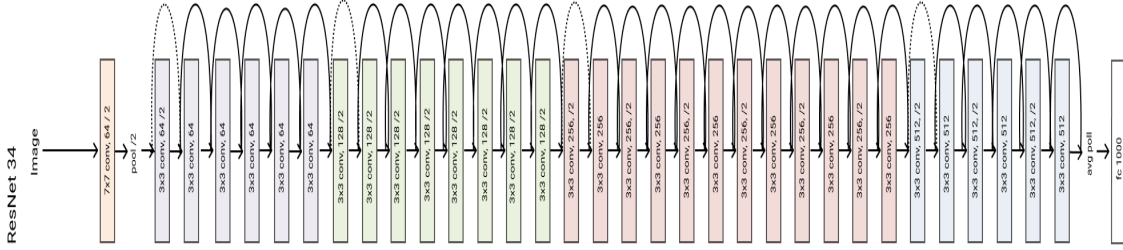


Fig 3.20: The ResNet- 34[5] Architecture.

From the above ResNet-34, when an input image is passed into the network, it pass through a 7\*7 conv filter, then to a pooling layer to make adjustment to the dimensions  $W_5$ . Notice that there is a pooling layer in-between the transition of one conv layer dimension which are just for dimensional adjustments. For instance,

Assume  $a^{[L+2]} = f(Z^{L+2} + a^{[L]})$ , if  $a^{[L+2]}$  is of depth 256 and  $a^{[L]}$  is of depth 128. it means that a fixed matrix of dimension  $W_5 = \mathbb{R}^{256 \times 128}$  must be multiplied to  $a^{[L]}$  to make the equation equal to  $a^{[L+2]} = f(Z^{L+2} + W_5 a^{[L]})$  or a matrix parameter learnt must be multiplied to  $a^{[L]}$ . The penultimate layer contains the average pooling which performs the role as seen in equation 3.39. and then a fully connected layer which might be a SoftMax function for classification. So, form the above, it can be seen that ResNet stack 3\*3 convolutional layers to form a very deep network. Each convolutional layer contains a batch normalization[50] to prevent internal covariant shift, Relu activation function and then the 3\*3 convolution simultaneously as shown below:

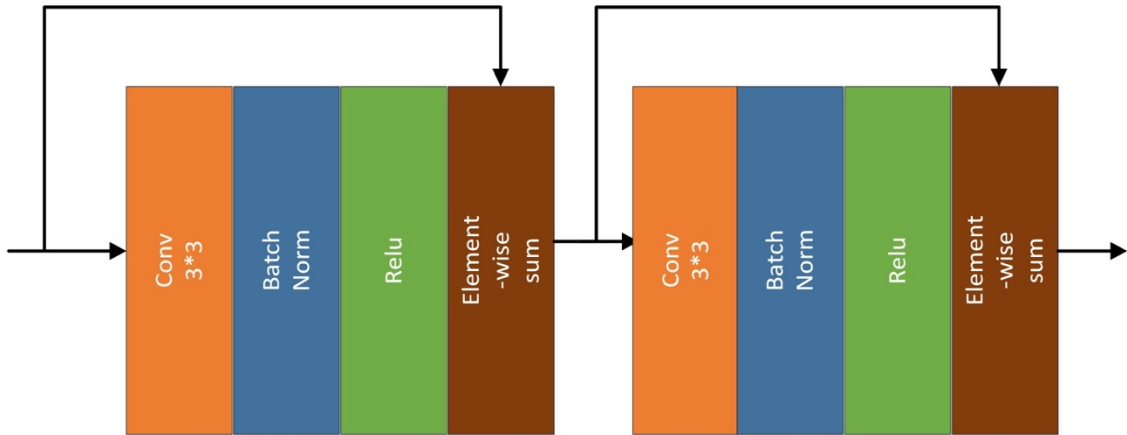


Fig 3.21: Batch Normalization and Relu activation in Residual block.

The above layers are stacked to form a deeper network with residual connections until the global pooling layers and then the SoftMax layer. Batch normalization is usually connected after the convolution layer and before the non-linearity, this makes the output to have a unit gaussian thereby normalizing the output of the convolutional layer. As given in [50], in as much as we want to control the output to be a unit Gaussian by eradicating

saturation, we want a little bit saturation. By implication, instead of totally constraining the output by batch normalization, we want to be able to control the saturation as shown in [53] [50]. So, batch normalization has several advantages like:

1. helps to control the saturation from the output of the convolutional layer.
2. It improves the flow of the gradient through the network, allows higher learning rate.
3. Reduces dependency on weight initialization and also
4. Acts as a form of regularization and slightly reducing the need for drop-out.

Some ResNet versions like ResNet-18 and ResNet-34 the residual block are straight forward stacking convolutional layers while other versions like ResNet-50, ResNet-152 introduces a bottleneck in between. This can be seen in the diagram below:

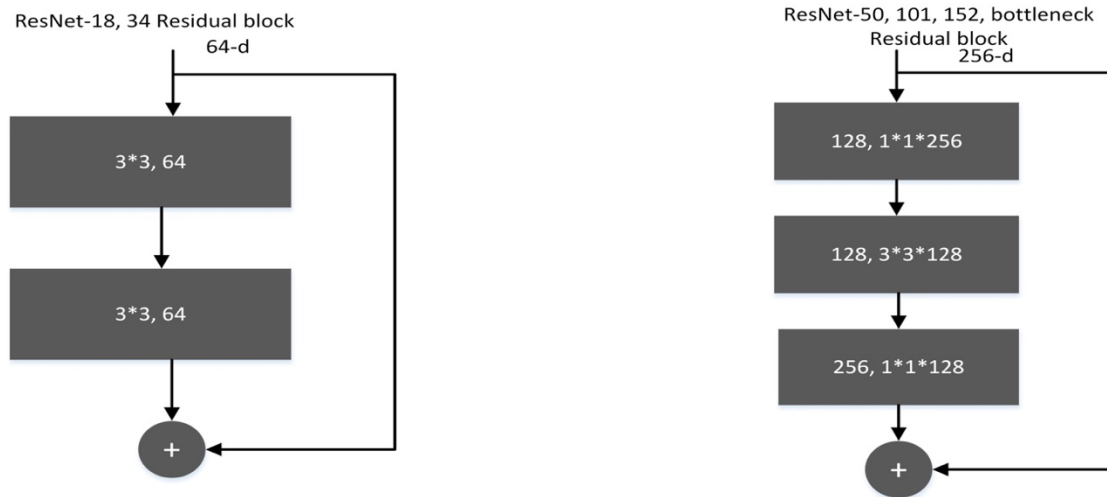


Fig 3.22: Variants of Residual Blocks in lower and higher layers of ResNet.

### Summary:

We have discussed the two convolutional neural network architectures that we used in this research above which are GoogleNet 22-layer and ResNet 34-layer. GoogleNet works by stacking inception layer together to produce a deeper layer network and reduce the number of parameter and also ResNet works by introducing skip connection or identity connection or residual layer which copies information from the shallower layer to the deeper layer with less parameter count. From the above discussion, we can conclude that the two convnets are deeper convnets with lesser parameters and more efficiency.

### 3.9.0 OUR WORK FLOW:

We used these convnet architectures for classifying smoke images. Below is the work flow diagram of our method.

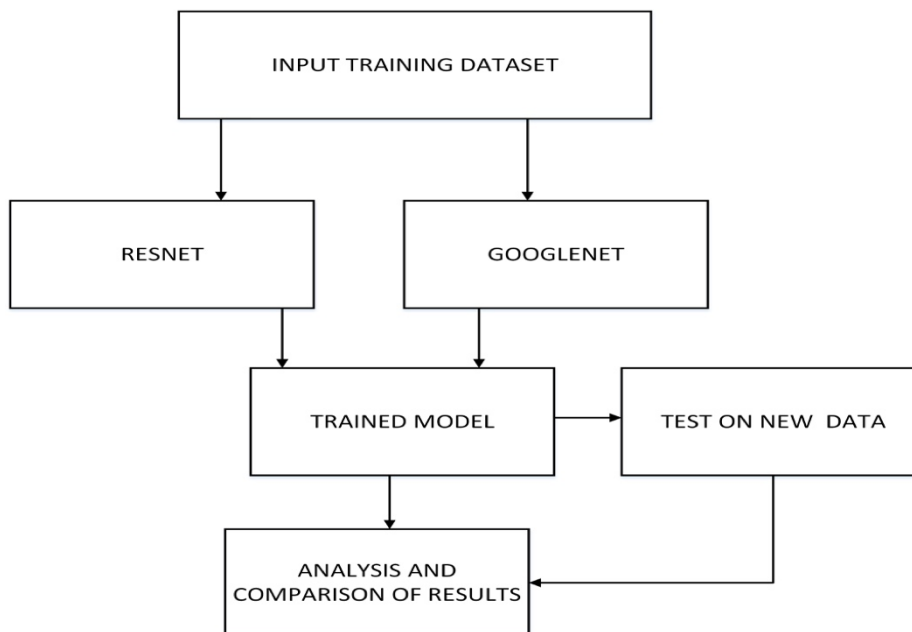


Fig 3.23: Our Work flow.

The input smoke training dataset is used to train GoogleNet-v1 with 22-layers and ResNet-34 with 34 layers and the trained model is used for testing for generalization. One of the challenges experienced in the field of deep learning is generalization. Generalization is the goal of any deep learning model. For a model to be good at image recognition and classification the model must be able to predict accurately on model it was not trained on. It can be called test dataset or unseen dataset.

After our trained model have been tested for generalization. Then we compare our result from both architectures. The input to ResNet-34 and GoogleNet-v1 is a combination of smoke images and background images.

### 3.9.1 Transfer learning:

In our method we use transfer learning. Transfer learning was important because of the unavailability of enough dataset. To train a convolutional neural network, large dataset is required. When the dataset is limited, our network will overfit and perform poorly because it is running on a powerful model. We discussed earlier that overfitting can be combated by regularization. Another way to reduce or eradicate overfitting is by use of transfer learning. Transfer learning uses the intuition that our dataset is small, so we use a pretrained convolutional neural network that was already trained on a very large dataset e.g. ImageNet dataset. We use it to then trained our small dataset by changing the output of the fully connected layer to 2 classes which might be smoke or no smoke. How this works is we take our network architecture i.e. ResNet and GoogleNet. Trained it on ImageNet and then apply the features or weights to our smoke datasets. So instead of classifying 1000 images as in  $1024 \times 1000$  we classify only smoke and no smoke which is

1024\*2. So, we freeze the weights and only do linear classification and let it on our dataset. This means that our pretrained GoogleNet and ResNet which must have been trained on ImageNet is frozen leaving other layers which include the input and the fully connected layer. We then fix our datasets as the input and change our Fully connected layer to 1024\*2 for prediction.

### 3.9.2 Dimension Calculation:

Table 2.0: Dimension Calculation for GoogleNet-v1.

Kind	Filter/ Stride, Padding	Output Size	Calculations: $\frac{N+2P-F+1}{s}$
Convolution	7*7/2 Pad=3	112*112*64	$\frac{224+2*3-7+1}{2}=112$
Max pool	3*3/2 Pad=1	56*56*64	$\frac{112+2*1-3+1}{2} = 56$
convolution	3*3/1 Pad=1	56*56*192	$\frac{56+2*1-3+1}{1} = 56$
Max pool	3*3/2 Pad=1	28*28*192	$\frac{56+2*1-3+1}{2} = 28$
Inception-1		28*28*256	$28 * 28 * (64 + 128 + 32 + 32) = 28 * 28 * 256$
Inception-2		28*28*480	$28 * 28 * (128 + 192 + 96 + 64) = 28 * 28 * 480$
Max pool	3*3/2 Pad= 1	14*14*480	$\frac{28+2*1-3+1}{2} = 14$
Inception-3		14*14*512	$14 * 14 * (192 + 208 + 48 + 64) = 14 * 14 * 512$
Inception-4		14*14*512	$14 * 14 * (160 + 224 + 64 + 64) = 14 * 14 * 512$
Inception-5		14*14*512	$14 * 14 * (128 * 256 * 64 * 64) = 14 * 14 * 512$
Inception-6		14*14*528	$14 * 14 * (112 + 288 + 64 + 64) = 14 * 14 * 528$
Inception-7		14*14*832	$14 * 14 * (256 + 320 + 128 + 128)$ $= 14 * 14 * 832$
Max pool	3*3/2 Pad=1	7*7*832	$\frac{14+2*1-3+1}{2} = 7$
Inception-8		7*7*832	$7 * 7 * (256 + 320 + 128 + 128) = 7 * 7 * 832$
Inception-9		7*7*1024	$7 * 7 * (384 + 384 + 128 + 128) = 7 * 7 * 1024$
Average Pooling	7*7/1	1*1*1024	$7 + 2 * 0 - 7 + 1 = 1$
Drop out (40%)		1*1*1024	
linear		1*1*2	
SoftMax		1*1*2	

For the inception layers above the filter sizes used can be seen in[4] and the process of calculation for the GoogleNet inception is shown in fig 3.94 and fig 3.96. The SoftMax layer is performing a binary classification while average pooling before the classifier layer performs no operations.

Because of the problem of transporting gradients backwards in large layer network (vanishing gradient), an auxiliary layer was added to solve this vanishing gradient problem at intermediate layers and also regularize the network. The auxiliary layers are added at inception-4 and inception-7 layer and it contains average pooling of 5\*5 filter size, stride 3 which gives 4\*4\*512 for inception-4 and 4\*4\*528 for inception-7, a 1\*1 convolution with 128 filters to reduce dimension with Relu, a fully connected layer with 1024 units and Relu, a drop out of 70% and a linear layer with SoftMax loss as a classifier



to predict the 2 output (smoke or no smoke) as the main classifier but these is removed at test time. Here all the layers are counted include the auxiliary layer and the Max pool layers, normalization, inception layer and the linear layer to give a total of 22-layers Network.

Table 3.0: Dimension calculation for ResNet-34.

Kind	Filter/Stride, Padding	Output Size/ skip connection	Calculations: $\frac{N+2P-F+1}{s}$
Convolution_1	7*7/2 Pad=3	112*112*64	$\frac{224+2 \times 3-7+1}{2}=112$
Max pool	3*3/2 Pad=1	56*56*64	$\frac{112+2 \times 1-3+1}{2}=56$
Convolution_2	3*3/1 Pad=1	56*56*64 / 6 skip connection	$\frac{56+2 \times 1-3+1}{1}=56$
Convolution_3	3*3/2 Pad=1	28*28*128 / 8 skip connection	$\frac{56+2 \times 1-3+1}{2}=28$
Convolution_4	3*3/2 Pad= 1	14*14*256/ 12 skip connection	$\frac{28+2 \times 1-3+1}{2}=14$
Convolution_5	3*3/2 Pad=1	7*7*512 / 6 skip connection	$\frac{14+2 \times 1-3+1}{2}=7$
Average Pooling	7*7/1	1*1*1024	$7+2 * 0 - 7 + 1 = 1$
Linear(fc)		1*1*2	
SoftMax		1*1*2	

From the above table for ResNet-34, it is only the convolutional layer and fully connected layers make up the name 34 layers. Other activation layer or pooling layers are not counted. That is: the first convolutional layer is one layer and the other convolutional layers with skip connection are:  $6 + 8 + 12 + 6 = 32$  layers and the fully connected layer is also counted as one layer. Then the total number of layers for ResNet-34 =  $1 + 32 + 1 = 34$  layers.

**Summary:**

Remember that in this research we are focused on comparative analyses between two convolutional networks as discussed in our method above. It also important to also to include certain input points as show in our dimensional calculation to help in understanding what each layer does. The input layer reads the data samples we fed it while the convolutional layers extract the features. The first convolutional layer looks for edges and lines of smoke while the second layer focus more on the colors and the rest convolutional layers have specific properties it learns. The pooling layers learns no parameter hence no weight is need as it is just for smoke image dimension reduction. Drop-out[52] is also important in dropping nodes to enable the weight spread out properly by shrinking it for regularization effect (reduce overfitting) at training time but we don't use drop out at test time because it adds noise. The fully connected layer and SoftMax classifier which tries to minimize the negative log likelihood of the correct class or simply do the binary classification. Next we start the data samples collection and actual experiments and analysis in chapter 4.

## 4. Experiment, Results and Analysis

### 4.1 Datasets collection:

Here, we outline the process of collecting our dataset samples. There are various ways of collecting dataset for our experiments like downloading already made datasets from the internet or creating a dataset. But for the purpose of this research, we are going to use our own dataset for training, validating and testing out our model. One of the key requirements for deep learning is enough data samples. Creating large datasets can be challenging and time consuming but even with small dataset, we can still train our network without experiencing overfitting by using transfer learning or data augmentation procedures. Below is an analysis of how our data sets was collected.

#### 4.1.1 Real smoke dataset:

Real smoke datasets were collected with HK VISION video camera, a smoke generator and a laptop with at least 100mega byte free space to capture the videos. The HKVISION software is downloaded on the laptop and the wired connection is made with the camera and the laptop port (USB port). The camera is fixed to a Tripod stand for convenience and samples was taken outdoor. This data samples contains real world scenarios like trees occlusion, Humans, houses and also weather anomalies. A total of 8 videos where gotten from the camera of different smoke background. The camera resolution was 1280 \* 720p with at least 2 minutes for each smoke video and a file size of approximately 15Megabyte per video. A corresponding static non-smoke background image was also capture for each of the real-world scenario. Reasons for using real world dataset is to observe how well our model does on real smoke data samples which are being affected by advanced weather conditions like wind, sunlight, haze, and other problems from the camera like chromatic aberration which is a common problem in lenses which occurs when colors are incorrectly refracted or bent by the lens, resulting in a mismatch at the focal point where the colors don't combine as they should. The smoke sample videos are then converted into corresponding images by extracting frames per video sample. A total of 880 smoke images where extracted with 110 background images without smoke. Microsoft word software was used to resize both the smoke images and the background images to 256\*256 resolution and save into different folder. Below is a Pictorial view of the camera setup:



Fig 4.1 Setup for smoke data samples capture.

#### 4.1.2 Synthetic dataset:

Here we try to simulate smoke onto non-smoke image using Adobe photoshop software. Different images were downloaded from the internet and also smoke PNG images where downloaded from the internet. The smoke PNG image is then juxtaposed with the non-smoke image to give a synthetic smoke image. This is done for several images to accumulate a total of 1100 synthetic smoke images and corresponding 110 background samples. These images are also resized to  $256 * 256$  using Microsoft word software and placed in a folder.

#### 4.2 Data preparation:

After obtain our real and synthetic dataset, we made sure our dataset is properly labeled into the inputs and the corresponding label to give a set of pairs  $(x, y)$ . Where  $x$  is some inputs example and  $y$  is the corresponding label. We then split the dataset into three folds, commonly a training, validation and test set with a percent of 80:10:10. The training set is for optimization of parameters when getting the loss and gradient during backpropagation while validation set is for hyper parameter optimization and the test set for evaluation our model for generalization.

### 4.3 Training, Validation and Test data:

Training data is the data that the models is trained on. This is different from the testing dataset. The sample dataset for our work is 1980 smoke images and 990 non smoke images for classification. The purpose of training the data samples with our model architecture is for the model to learns to assign smoke images to their corresponding labels. The goal of the training is to fit the model to the training dataset while still generalizing to unseen data. In image classification, only training data is labelled.

Validation dataset is a dataset that helps us to adjust out hyper parameter. Note that validation set have not been seen before by the model and it helps to prevent our model from overfitting. Our validation set is 10 percent of the total dataset. The goal of the validation set is to check how well the model is generalizing. One key difference between validation set and test set is that test set doesn't have label meanwhile validation set have labels.

Testing dataset is the data that is used to predict if the model has learnt enough to accurate generalize. Our test dataset is 10 percent of the total dataset Most times testing data is not as large as training data. Note that testing data is that data that the model has not been trained on. It gives us a metric of how good our algorithm is doing on classifying images or a dataset.

For this experiment we have Real smoke of 880 images, synthetic smoke of 1100 images, background images both captured and downloaded images containing no smoke of 980 images. The total number of dataset samples is 2970 which will be categories into 80percent training data, 10 percent validation data and 10 percent testing data. This is general done in our program but for the purpose of this analysis our training data is 2376 images, while our validation and test set are both 297 images each.

### 4.4 Data preprocessing:

We convert all our dataset to the same resolution dimension of 256\* 256 which will become the input to our network. We also made sure our dataset is appropriately labelled and named in the proper file directory to be called upon from the program when the need arises. We hand pick our data samples to prevent repeating images. Below is the pictorial view of our data samples.

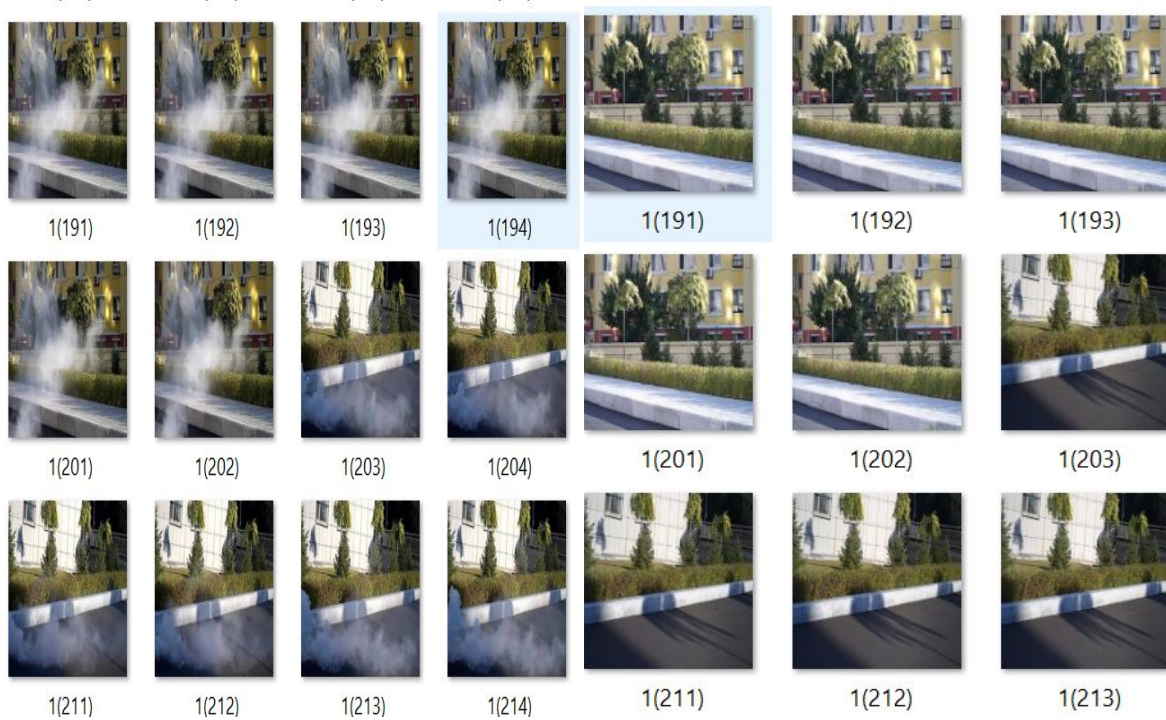


Fig 4.2: Real smoke data and background image samples.

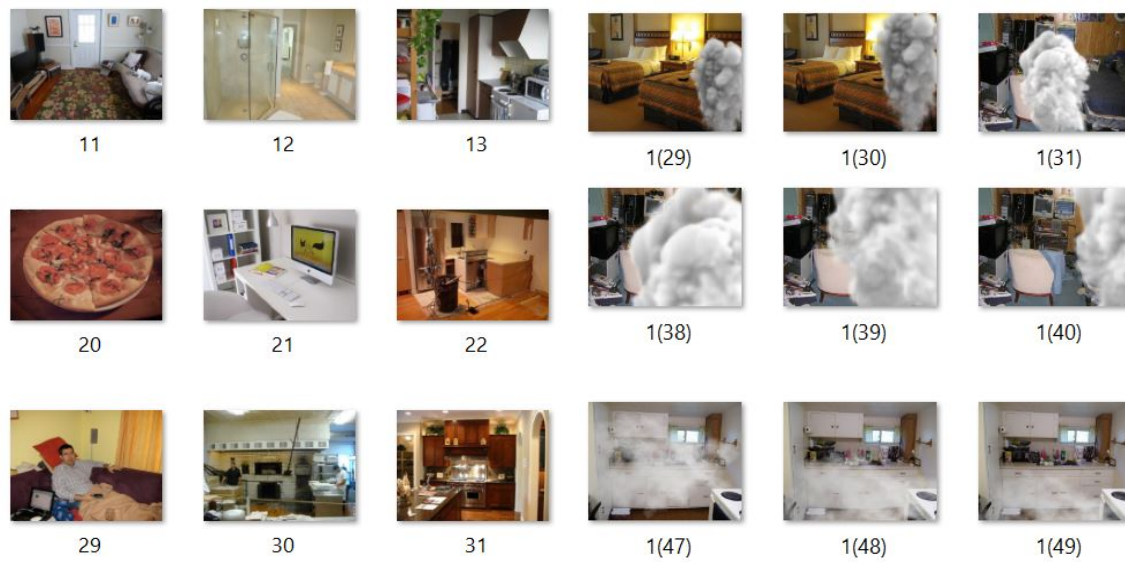


Fig 4.3: Synthetic smoke and background images download from the internet.

## 4.5 Experimental procedures:

Due to the unavailability of enough large dataset for our training we employ the process of transfer learning in our experiments by using a pretrained network (GoogleNet, Vgg-19) already trained on ImageNet and freezing the middle layer containing various convolutional layers, pooling layers, Relu and Batch Norm layers and training only the input layer and output layer for this classification. Firstly, we download the pretrained network weights from GitHub and also our network architecture from GitHub since we don't need to code from scratch, we only have to fine tune. We download ResNet-34 and GoogleNet v1- 22 layers from the network Zoo. This experiment was carried out using python programming language with TensorFlow framework.

## 4.6 Training Requirements and details:

This training requires at least python 3.6 and TensorFlow 2.0 and keras open source library for building our models but since the model have been built, we just modify the model architecture to suits our use. We import various modules like Matplotlib for plotting and visualization, NumPy for multi-dimensional arrays and matrices operations, Pillow for opening, manipulating and saving image file formats. The training is done on GeForce GTX 1080 Ti GPU with 16gb ram.

### 4.6.1 Training and Hyper parameter optimization and Result:

We initialize the network with weights pretrained on ImageNet. We use stochastic gradient descent with momentum 0.9 to train the weight of the convolutional network and Adam[41] to train the other components of the model. We used a learning rate of 0.1 and a decay rate of  $1 \times 10^{-6}$  and Nestorov momentum of friction set  $\beta_1 = 0.9$ ,  $\beta_2 = 0.99$  and we keep finetuning our network and train with batch-size of 2 and epochs of 10. We do this for both network architectures.

Training lasted approximately 5 hours for each architecture (GoogleNet and ResNet) with a training loss of 0.11 and an accuracy of 0.89 and validation loss of 0.15 and accuracy of 0.85 for GoogleNet which is close and good because the validation loss is not far away from the training loss which shows our model is doing well on un seen data. For ResNet-34, our training loss is 0.07 and accuracy is 0.93 and validation loss of 0.12 and accuracy of 0.88.

Table 4.0: Training and Validation results.

<b>Training Details</b>	<b>Training loss</b>	<b>Validation loss</b>	<b>Training Accuracy</b>	<b>Validation Accuracy</b>
GoogleNet-v1	0.11	0.15	0.89	0.85
ResNet-34	0.07	0.14	0.93	0.88

From the table above we noticed that our model training accuracy is 89% for GoogleNet-v1 and 93% for ResNet-34 which is as a result of the skip connections and the deeper layer of ResNet, it was able to learn the samples better than GoogleNet-v1 which is also good because the two-network validation loss was close which also tells us that our network is not overfitting.

But the goal of deep learning is not how well our model do during training but how well our model do during testing. Testing means feeding our model with unseen data and observing how well our model do with classifying smoke images. Our 10% testing data is 297 images which is the images our model never saw. Below the table for the results for our test dataset.

Table 5.0: Testing results.

<b>Testing Details</b>	<b>Test Accuracy</b>	<b>Testing Loss</b>
GoogleNet-v1	0.88	0.12
ResNet-34	0.90	0.10

#### 4.8 SUMMARY:

From the above we notice that our model architectures are both doing well in testing results with GoogleNet-v1 having a testing accuracy of 88% and ResNet having an accuracy of 90%. This means that ResNet does better in classifying smoke images from non- smoke images better than GoogleNet-v1. This is as a result of deep network of skip connections which is easier to learn and therefore layers are copied to the deep layers which reduces computation complexity. GoogleNet-v1 which has 22 layers is not as deep as ResNet with 34 layers but still does well in classification but not as good as ResNet. This justifies the reasons for Microsoft ResNet winning the 2015 ImageNet competition. ResNet-34 and GoogleNet-v1 shows high accuracy for real and synthetic dataset which means our model is robust to real world data samples and will do well in it is deployed to the real world.

## 5. Conclusion

It is noticeable that over the past few years, image classification has achieved tremendous accuracy in classification of various materials. Once you show the model an image it correctly classifies the image to its corresponding label. Generally supervised learning has greatly improved in the field of artificial intelligence and a vast majority of researchers dived into it and great contributions were made to improve the models, algorithms, hyperparameters etc. and a lot of software's, libraries, API's and framework have greatly added to the deep learning community.

The development of convolutional neural network also helps the deep learning field to be noticed bringing forth state of the art convolutional neural network architectures that did very well in visual recognition task e.g. object detection, image classification, localization, image captioning, image segmentation, instant segmentation etc.

Smoke classification using these convolutional neural networks makes classification of smoke image from non-smoke image easy and quite straightforward. Binary classification and Multiclass classification can be done using ImageNet dataset or any dataset. ImageNet dataset and other online datasets also helped to combat the problem of limited dataset and also transfer learning helped too.

In this thesis, we introduced our dataset which lets us observe how well our model did on real world dataset and also gives us a notion of how to prepare datasets for image classification which gives us a very high accuracy of 90% for ResNet-34 and 88% for GoogleNet-v1.

In conclusion, smoke image classification is another way of recognizing smoke image from non-smoke image in other to detection and combat the menace of fire outbreak. Since our model can learn to classify smoke, it can also be trained to detect smoke using various method.



## 6. Reference

- [1]Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998, doi: 10.1109/5.726791.
- [2]A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Neural Information Processing Systems*, vol. 25, 01/01 2012, doi: 10.1145/3065386.
- [3]M. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Neural Networks*. 2013.
- [4]C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7-12 June 2015 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.
- [5]K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 27-30 June 2016 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [6]K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv 1409.1556*, 09/04 2014.
- [7]H.-E. Bouali, M. Zghal, and Z. Lakhdar, "Popularisation of optical phenomena: establishing the first Ibn Al-Haytham workshop on photography," 10/24 2005, doi: 10.1117/12.2207764.
- [8]S. Jianbo and J. Malik, "Normalized cuts and image segmentation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 17-19 June 1997 1997, pp. 731-737, doi: 10.1109/CVPR.1997.609407.
- [9]P. Viola and M. Jones, *Rapid Object Detection using a Boosted Cascade of Simple Features*. 2001, pp. 1-511.
- [10]N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 20-25 June 2005 2005, vol. 1, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.
- [11]P. Felzenszwalb, D. McAllester, and D. Ramanan, *A Discriminatively Trained, Multiscale, Deformable Part Model*. 2008.
- [12]M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303-338, 06/01 2010, doi: 10.1007/s11263-009-0275-4.
- [13]J. Deng, W. Dong, R. Socher, L. Li, L. Kai, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 20-25 June 2009 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.
- [14]M. Z. Alom *et al.*, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, p. 292, 03/05 2019, doi: 10.3390/electronics8030292.

- [15]J. Brownlee, "Introduction to the Imagenet Large Scale Visual Recognition Challenge," May 1 2019. [Online]. Available: [machinelearningmastery.com](http://machinelearningmastery.com).
- [16]A. Karpathy, "Image Classification," 2017. [Online]. Available: [cs231n.github.io](https://cs231n.github.io).
- [17]B. Töreyn, Y. Dedeoglu, U. Gudukbay, and A. Cetin, "Computer vision based method for real-time fire and flame detection," *Pattern Recognition Letters*, vol. 27, pp. 49-58, 01/01 2006, doi: 10.1016/j.patrec.2005.06.015.
- [18]C. Yu, Z. Mei, and X. Zhang, "A Real-time Video Fire Flame and Smoke Detection Algorithm," *Procedia Engineering*, vol. 62, pp. 891-898, 12/31 2013, doi: 10.1016/j.proeng.2013.08.140.
- [19]M. Favorskaya, A. Pyataeva, and A. Popov, "Verification of Smoke Detection in Video Sequences Based on Spatio-temporal Local Binary Patterns," *Procedia Computer Science*, vol. 60, pp. 671-680, 12/31 2015, doi: 10.1016/j.procs.2015.08.205.
- [20]F. Yuan, "A fast accumulative motion orientation model based on integral image for video smoke detection," *Pattern Recognition Letters*, vol. 29, pp. 925-932, 05/01 2008, doi: 10.1016/j.patrec.2008.01.013.
- [21]J. Zeng, Z. Lin, C. Qi, X. Zhao, and F. Wang, *An Improved Object Detection Method Based On Deep Convolution Neural Network For Smoke Detection*. 2018, pp. 184-189.
- [22]Z. Yin, B. Wan, F. Yuan, X. Xia, and J. Shi, "A deep normalization and convolutional neural network for image smoke detection," *IEEE Access*, vol. PP, pp. 1-1, 08/30 2017, doi: 10.1109/ACCESS.2017.2747399.
- [23]S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, *Convolutional neural network for video fire and smoke detection*. 2016, pp. 877-882.
- [24]S. Wu, F. Yuan, Y. Yang, Z. Fang, and Y. Fang, "Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis," *IET Image Processing*, vol. 9, 06/17 2015, doi: 10.1049/iet-ipr.2014.1032.
- [25]Y. Liu, W. Qin, K. Liu, F. Zhang, and Z. Xiao, "A Dual Convolution Network Using Dark Channel Prior for Image Smoke Classification," *IEEE Access*, vol. PP, pp. 1-1, 05/08 2019, doi: 10.1109/ACCESS.2019.2915599.
- [26]H. Tian, W. Li, L. Wang, and P. Ogunbona, *A Novel Video-Based Smoke Detection Method Using Image Separation*. 2012, pp. 532-537.
- [27]H. Tian, W. Li, P. Ogunbona, and L. Wang, "Detection and Separation of Smoke From Single Image Frames," *IEEE Transactions on Image Processing*, vol. PP, pp. 1-1, 11/08 2017, doi: 10.1109/TIP.2017.2771499.
- [28]C. Long *et al.*, *Transmission: A New Feature for Computer Vision Based Smoke Detection*. 2010, pp. 389-396.
- [29]C. Ledig *et al.*, *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*. 2017, pp. 105-114.
- [30]K. He, X. Zhang, S. Ren, and J. Sun, *Identity Mappings in Deep Residual Networks*. 2016, pp. 630-645.
- [31]Z. Zhu, J. Li, L. Zhuo, and J. Zhang, "Extreme Weather Recognition Using a Novel Fine-Tuning Strategy and Optimized GoogLeNet," in *2017 International Conference on Digital*

- Image Computing: Techniques and Applications (DICTA)*, 29 Nov.-1 Dec. 2017 2017, pp. 1-7, doi: 10.1109/DICTA.2017.8227431.
- [32]P. Aswathy, Siddhartha, and D. Mishra, "Deep GoogLeNet Features for Visual Object Tracking," in *2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, 1-2 Dec. 2018 2018, pp. 60-66, doi: 10.1109/ICIINFS.2018.8721317.
- [33]Z. Zhong, L. Jin, and Z. Xie, "High performance offline handwritten Chinese character recognition using GoogLeNet and directional feature maps," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 23-26 Aug. 2015 2015, pp. 846-850, doi: 10.1109/ICDAR.2015.7333881.
- [34]R. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogLeNet models for malware detection," *Journal of Computer Virology and Hacking Techniques*, 08/28 2018, doi: 10.1007/s11416-018-0324-z.
- [35]J. Canny, "A Computational Approach To Edge Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, pp. 679-698, 12/01 1986, doi: 10.1109/TPAMI.1986.4767851.
- [36]S. Sharma, "Activation Functions in Neural Networks," September 6 2017. [Online]. Available: [towardsdatascience.com/activation-functions-in-Neural-Network](https://towardsdatascience.com/activation-functions-in-Neural-Network).
- [37]K. Janocha and W. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," *Schedae Informaticae*, vol. 25, 02/18 2017, doi: 10.4467/20838476SI.16.004.6185.
- [38]S. Hochreiter, A. Younger, and P. Conwell, *Learning To Learn Using Gradient Descent*. 2001, pp. 87-94.
- [39]M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," 06/14 2016.
- [40]N. Zhang, D. Lei, and J. F. Zhao, *An Improved Adagrad Gradient Descent Optimization Algorithm*. 2018, pp. 2359-2362.
- [41]D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12/22 2014.
- [42]Y. Dauphin, H. Vries, J. Chung, and Y. Bengio, "RMSProp and equilibrated adaptive learning rates for non-convex optimization," *arXiv*, vol. 35, 02/15 2015.
- [43]A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 14-19 May 2017 2017, pp. 1899-1903, doi: 10.1109/IJCNN.2017.7966082.
- [44]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back propagating errors. Cogn," *Nature*, vol. 5, 01/01 1986.
- [45]K. S. Sivamani. "Back-propagation-simplified." [towardsdatascience.com/back-propagation-simplified](https://towardsdatascience.com/back-propagation-simplified) (accessed).
- [46]A. Kaparthy, "Neural Network," 2017. [Online]. Available: [cs231n.github.io](https://cs231n.github.io).
- [47]I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

- [48]Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 12/01 1998, doi: 10.1109/5.726791.
- [49]J. Salomon and B. Phelan, "Lung Cancer Detection using Deep Learning," 2018.
- [50]S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 02/10 2015.
- [51]M. Lin, Q. Chen, and S. Yan, "Network In Network," 12/16 2013.
- [52]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 06/01 2014.
- [53]S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, *How Does Batch Normalization Help Optimization? (No, It Is Not About Internal Covariate Shift)*. 2018.

