SONA COLLEGE OF TECHNOLOGY
Learning is a Celebration!
| An Autonomous Institution |

Department of Electronics and Communication Engineering

TRANSFORMS AND ALGORITHM IN SIGNAL AND IMAGE PROCESSING

2021 - 22

Junction Main Road
Suramangalam (PO)
Salem-636 005. TN. India
www.sonatech.ac.in

Editorial Head			
Dr.R.S.Sabeenian, Professor &Head, Dept of ECE, Head R&D Sona SIPRO			
Editorial Members			
Dr.M.E.Paramasivam Associate Professor		Dr. T.Shanthi Assistant (Sr.G) Professor	
Dr.P.M.Dinesh Assistant Professor		Prof.R.Anand Assistant Professor	
Prof.Eldho paul Assistant Professor		Magazine co-ordinator Dr.K.Manju Assistant Professor	

PREFACE

The field of signal and image processing encompasses the theory and practice of algorithms and hardware that convert signals produced by artificial or natural means into a form useful for a specific purpose. The signals might be speech, audio, images, video, sensor data, telemetry, electrocardiograms, or seismic data, among others; possible purposes include transmission, display, storage, interpretation, classification, segmentation, or diagnosis.

Current research in digital signal processing includes robust and low complexity filter design, signal reconstruction, filter bank theory, and wavelets. In statistical signal processing, the areas of research include adaptive filtering, learning algorithms for neural networks, spectrum estimation and modeling, and sensor array processing with applications in sonar and radar. Image processing work is in restoration, compression, quality evaluation, computer vision, and medical imaging. Speech processing research includes modeling, compression, and recognition. Video compression, analysis, and processing projects include error concealment technique for 3D compressed video, automated and distributed crowd analytics, stereo-to-auto stereoscopic 3D video conversion, virtual and augmented reality.

SIGN LANGUAGE CONVERSION AND RECOGNITION

SUVETHA.S, SHAHIN BANU.A

Human Computer Interaction moves forward in the field of sign language interpretation. Vision based hand gesture recognition system have been discussed as hand plays vital communication mode. Vision based system have challenges over traditional hardware based approach; by efficient use of computer vision and pattern recognition, itis possible to work on such system which will be natural and accepted, in general. The objective of this project is to convert the hand signs into the voice command. This is done with the help of MATLAB and embedded technology. Hand movement detect is done using MATLAB. So, in every hand movement is the edges of the fingers are detected and counted. This counting is given to the Arduino. Here we use flash type re programmable Arduino. This controller is already programmed for our process. Then it gives this pulse to the speaker through voice IC.

Present industry is increasingly shifting towards automation. Two principles components of today’s industrial automations are programmable controllers.

Here we use that embedded technology to translate the hand signs into the equivalent letter and voice. Arduino is the heart of the device which handles all the sub devices connected across it. It has flash type re programmable memory. It has some peripheral devices to play this project perform.

It also provides sufficient power to inbuilt peripheral devices. We need not give individually to all devices. The peripheral devices also activate as low power operation mode. These are the advantages are appearing here.

We have separate instructions for separate signs and that has been fed into the MATLAB and Arduino programs. The outputs will be in two different modes,

1.They will be displayed in the output screen on the PC

2.The instructions or the voice command for the sign will be heard over speakers

ARDUINO UNO:

Arduino Uno is a micro controller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

BLOCK DIAGRAM:

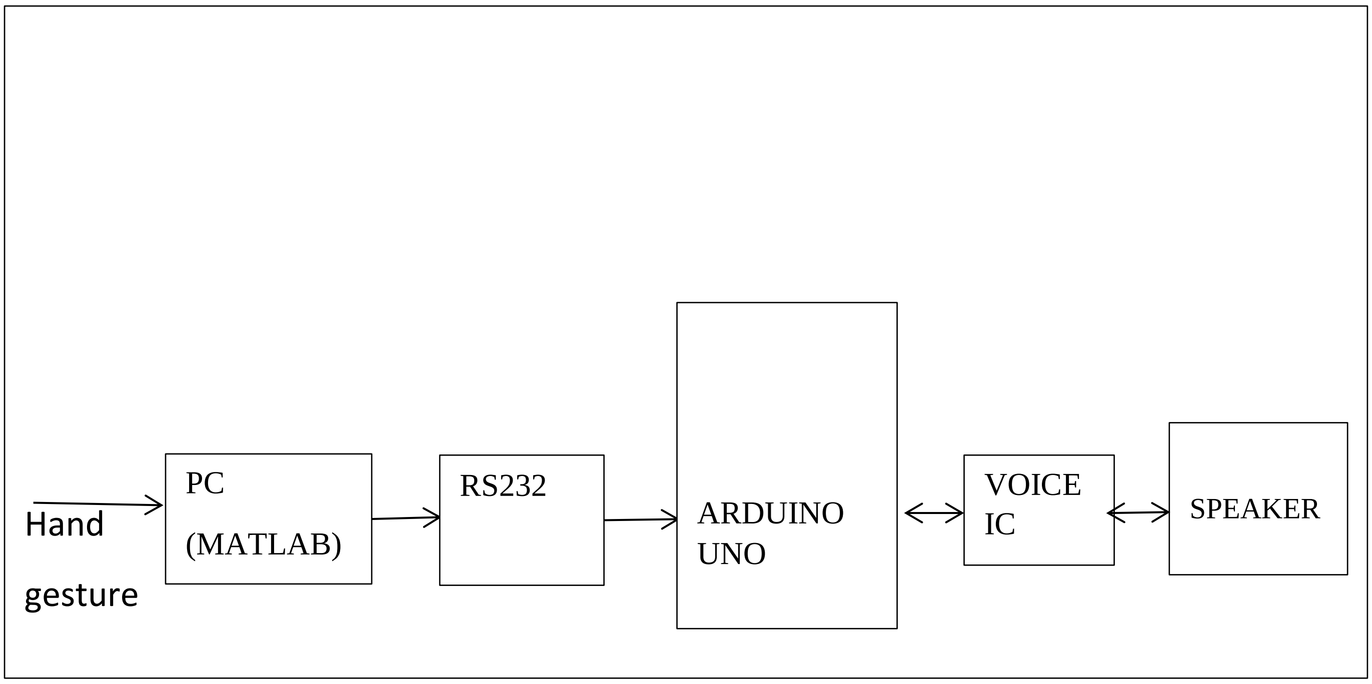


Figure: 1 Arduino uno

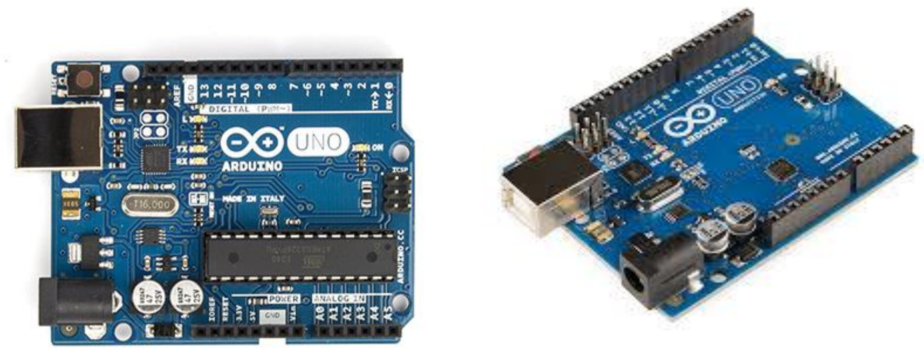


Figure: 2 Arduino uno

The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past, or outdated boards see the Arduino index of boards

POWER:

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the

TECHNICAL FACTORS

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6

Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm

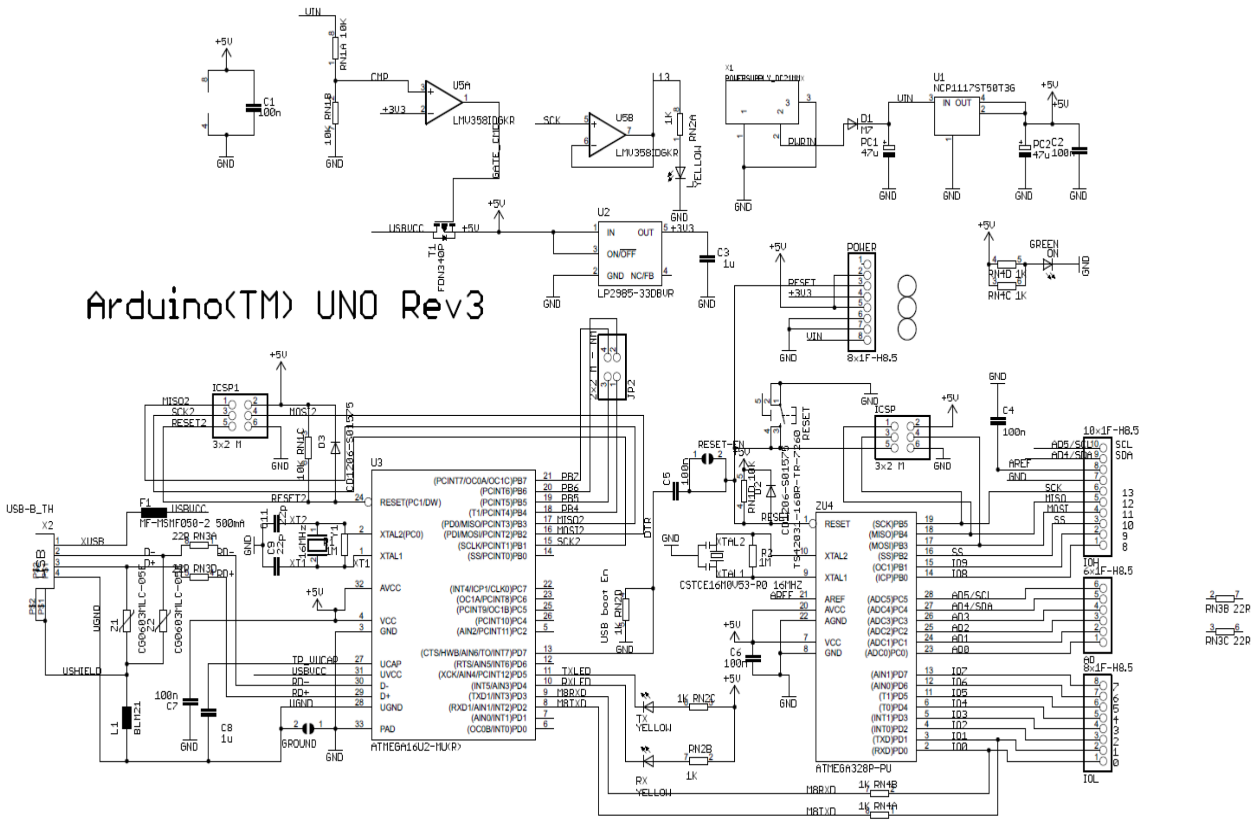


Figure:3 circuit diagram of Arduino uno

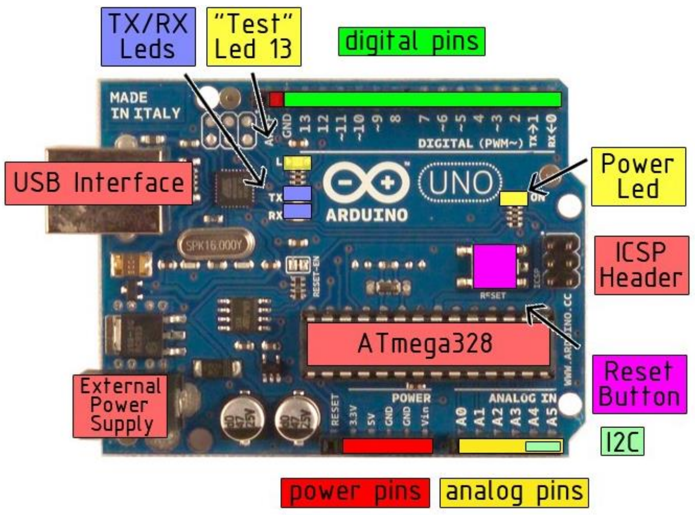


Figure: 4 Arduino uno board description

board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V).
- 3V3. A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino board provides the voltage reference with which the micro controller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

MEMORY:

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

In addition, some of the pins have specialised functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt () function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e., 1024 different values).

There are a couple of other pins on board:

- AREF. Reference voltage for the analog inputs. Used with analog Reference ().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

COMMUNICATION:

Arduino Uno has several facilities for communicating with a computer, another Arduino board, or other micro controllers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed.

The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

AUTOMATIC (SOFTWARE) RESET:

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nano farad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the interface toolbar. This means that the boot loader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

REVISIONS:

Revision 3 of the board has the following new features:
1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

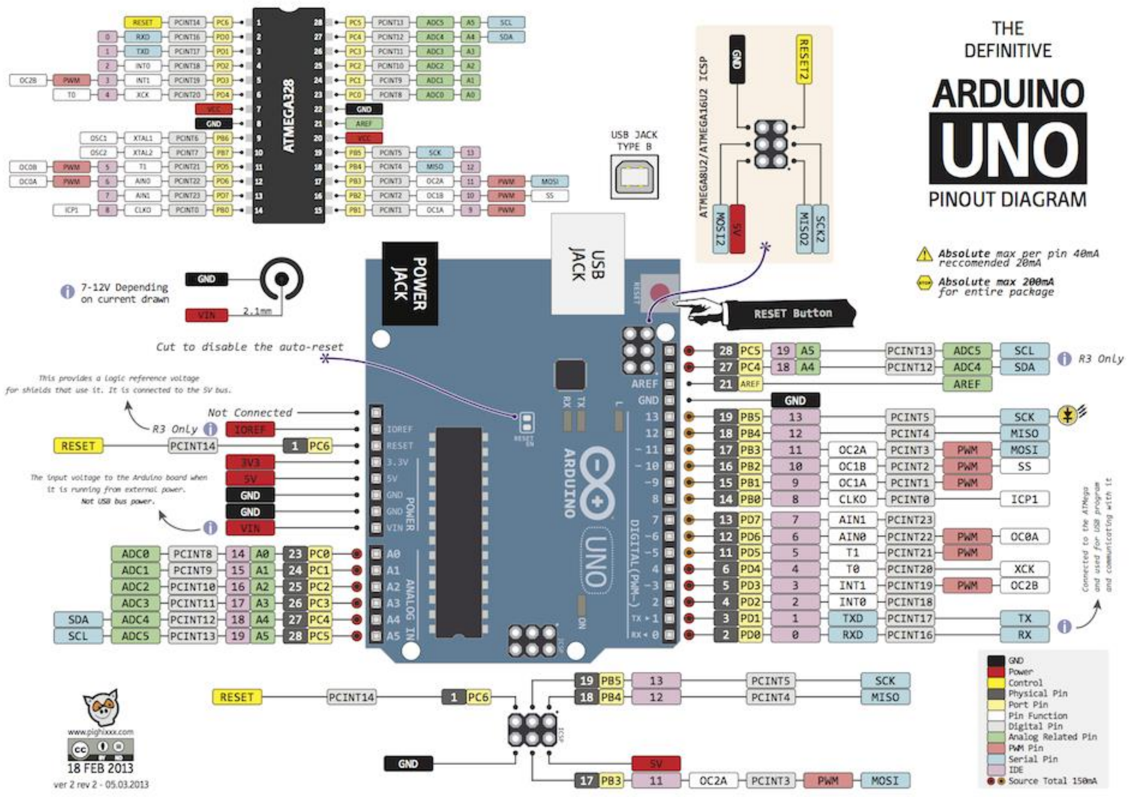

THE DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK TYPE B
POWER JACK
RESET Button
Cut to disable the auto-reset
7-12V Depending on current drawn
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
ATMEGA8U2/ATMEGA16U2 ICSP
Not Connected
R3 Only
18 FEB 2013
ver 2 rev 2 - 05.03.2013

Figure:5 Arduino Uno Pin Diagram

PERSONAL COMPUTER

A **personal computer** (**PC**) is any general-purpose computer whose size, capabilities, and original sales price make it useful for individuals, and which is intended to be operated directly by an end user with no intervening computer operator.

RS232:

In telecommunications, **RS-232** is a standard for serial binary data interconnection between a *DTE* (Data terminal equipment) and a *DCE* (Data Circuit-terminating Equipment). It is commonly used in computer serial ports.

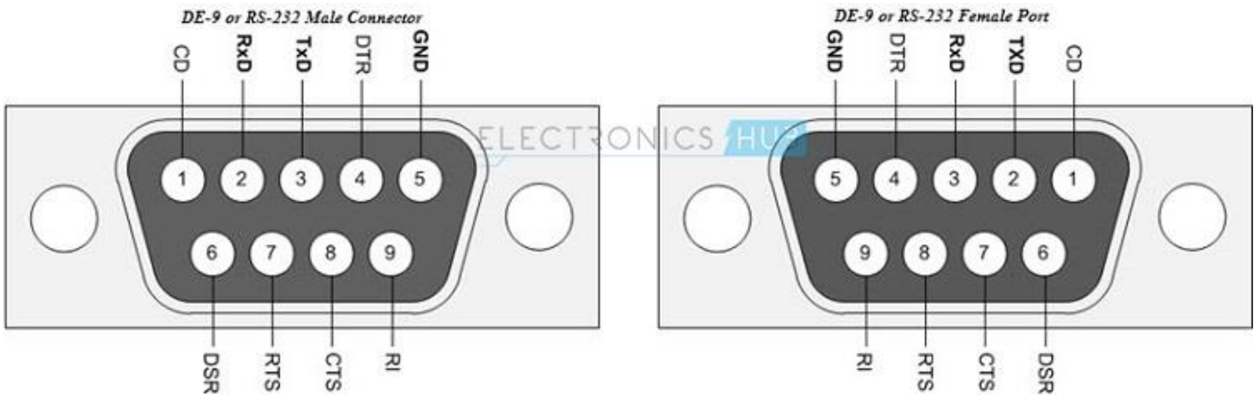


Figure:7 RS-232 Female port

VOICE IC

Features of voice IC:

Up to 7 kinds of operating modes: MP3 mode, one to one key mode, parallel mode, one record one play key mode, Audio-book mode, two-wire serial mode and three -wire serial mode

1. Support MIC and LINE -IN recording.
2. Support plug-in 64M bit SPI-FLASH, recording time up to 1600 seconds
3. Support upload and download voice via USB;
4. Support playback the high -quality voice which downloaded from computer
5. Can record up to 252 segments voice (including fixed voice);
6. With function of power-down data protection
7. Support audio recording at 10KHz or 14KHZ sample rate
8. Adopt a separate document management system, recording without debris, more reasonable distribution of SPI-FLASH space;
9. Support key and MCU control;
10. 8-level controllable volume;

11. Operating voltage: DC2.6 ~ 3.6V, sleep current :10uA (typical).

12. Module package: DIP28, can be defined, reference value: 18.30mm * 36.00mm

13. WTV-SR using SPI-FLASH store voice messages, the current version supports the largest of 64M. Specific as following table, the table data is from audio recording at 10KHz sample rate (unit: sec)

SPEAKER:

A **loudspeaker** (or "speaker") is an electro acoustic transducer that converts an electrical signal into sound. The speaker moves in accordance with the variations of an electrical signal and causes sound waves to propagate through a medium such as air or water.

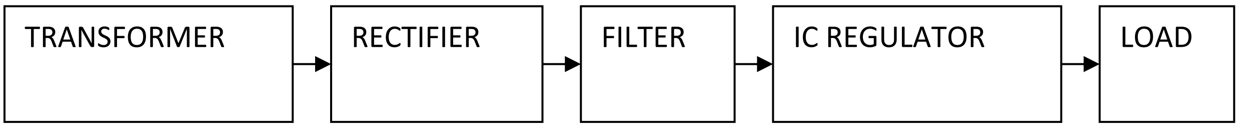
After the acoustics of the listening space, loudspeakers (and other electroacoustic transducers) are the most variable elements in a modern audio system and are usually responsible for most distortion and audible differences when comparing sound systems

OVERALL, POWER SUPPLY

Block diagram

The ac voltage, typically 220V rms, is connected to a transformer, which steps that ac voltage down to the level of the desired dc output. A diode rectifier then provides a full wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage. This resulting dc voltage usually has some ripple or ac voltage variation.

A regulator circuit removes the ripples and also remains the same dc value even if the input dc voltage varies, or the load connected to the output dc voltage changes. This voltage regulation is usually obtained using one of the popular voltage regulator IC units.



F*igure :8 Arduino uno*

TRANSFORMER

The potential transformer will step down the power supply voltage (0-230V) to (0-6V) level. Then the secondary of the potential transformer will be connected to the precision rectifier, which is constructed with the help of op–amp. The advantages of using precision rectifier are it will give peak voltage output as DC, rest of the circuits will give only RMS output.

BRIDGE RECTIFIER

When four diodes are connected, the circuit is called as bridge rectifier. The input to the circuit is applied to the diagonally opposite corners of the network, and the output is taken from the remaining two corners.

Let us assume that the transformer is working properly and there is a positive potential, at point A and a negative potential at point B. the positive potential at point A will forward bias D3 and reverse bias D4.

The negative potential at point B will forward bias D1 and reverse D2. At this time D3 and D1 are forward biased and will allow current flow to pass through them; D4 and D2 are reverse biased and will block current flow.

The path for current flow is from point B through D1, up through RL, through D3, through the secondary of the transformer back to point B. this path is indicated by the solid arrows. Wave forms (1) and (2) can be observed across D1 and D3.

IC VOLTAGE REGULATORS

Voltage regulators comprise a class of widely used ICs. Regulator IC units contain the circuitry for reference source, comparator amplifier, control device, and overload protection all in a single IC.

IC units provide regulation of either a fixed positive voltage, a fixed negative voltage, or an adjustably set voltage. The regulators can be selected for operation with load currents from hundreds of milli amperes to tens of amperes, corresponding to power ratings from milli watts to tens of watts.

A fixed three-terminal voltage regulator has an unregulated dc input voltage, Vi, applied to one input terminal, a regulated dc output voltage, Vo, from a second terminal, with the third terminal connected to ground.

The series 78 regulators provide fixed positive regulated voltages from 5 to 24 volts. Similarly, the series 79 regulators provide fixed negative regulated voltages from 5 to 24 volts.

EAR IMAGE DENOISING AND CLASSIFICATION USING DEEP LEARNING

ABISHEK.A, MEIALAGAN.K, MOKESH.A, NAGARJUN.S

Deep learning has improved over the decades on image analysis tasks. It created a lot of opportunities in various fields. Producing results from the image is a complex task especially in dealing with medicine-related fields, precision is more important. Ear disease identification using image analysis is yet to be explored. The middle ear and mastoid diseases can be easily cured by early detection, which prevents hearing impairment with appropriate medical care. The main reason for hearing impairment is the infections that occur in the middle ear. They are painful because of the inflammation and excess fluid build-up in the middle ear which causes pressure in the eardrum. There is a short of specialist and low diagnostic accuracy calls for the new way of diagnostic strategy, in which deep learning can play a significant role. In this work, two major middle ear diseases have been focused namely normal, myringosclerosis, and chronic otitis media. Otoendoscopy is a device used by ENT specialist to take images of the inner ear. The images obtained by the device may contain noises which leads to degradation in the accuracy of the model. Denoising of the images before passing them to the model is a very crucial task. To get high precision in the identification of the disease. A denoising algorithm is also built to remove the standard noise that could occur in the image namely mixed noises, salt and pepper noise, gaussian noise, image blur. A modified DNCNN architecture with median layers in between is used. Denoised images are then feed into a modified VGG19 architecture. VGG19 has 14 convolution layer, 5 max pooling layers, 3 fully connected layers with 143 M parameters which can be trained.

The CNN architecture includes several building blocks, such as convolution layers, pooling layers, and fully connected layers. A typical architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers. The step where input data are transformed into output through these layers is called forward propagation. Although convolution and pooling operations described in this section are for 2D-CNN, similar operations can also be performed for three-dimensional (3D)-CNN. The images dataset used in different CNN architectures for ear image classification may contain noise, which will degrade the ear classification models performance. A denoising model is used to remove the noise in the images. First, the image will be passed to the denoising model, the output is denoised image. Second, the denoised image will be passed to the classification model, the output is identified disease as shown in the Fig 1. For the denoising model, we have used convolutional neural network (CNN) with residual learning strategy and batch normalization. There has been a constant development of denoising algorithms over the decades using deep learning. We have mentioned some of the existing methods in this section.

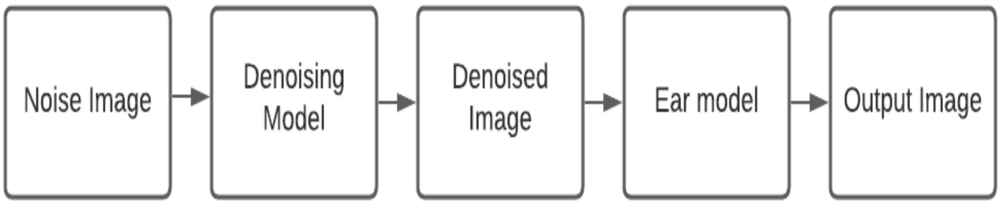


Fig 1 Block diagram of the image classification algorithm

CONVOLUTION LAYER

A convolution layer is a key component of the CNN architecture that performs the extraction function, which usually consists of a mixture of linear and non-linear operations, i.e., convolution and activation. Convolution is a specialized type of linear operation used for the extraction of a feature, where a smaller array of numbers, called the kernel, is applied across input, which is just an array of numbers called a tensor. The element-wise product between every element of the kernel and the input tensor is calculated for each position of the tensioner and totalled up to obtain the output in the corresponding point of the output tensor, called the feature map. This process is replicated by adding several kernels to an infinite number of feature maps describing the various features of the input tensors; different kernels can thus be viewed as different feature extractors. Two primary hyperparameters defining the convolution operation are the size and number of kernels. The former is normally 3×3, but often 5×5 or 7×7.The latter is arbitrary and determines the depth of output feature maps. Shown in Fig 2.

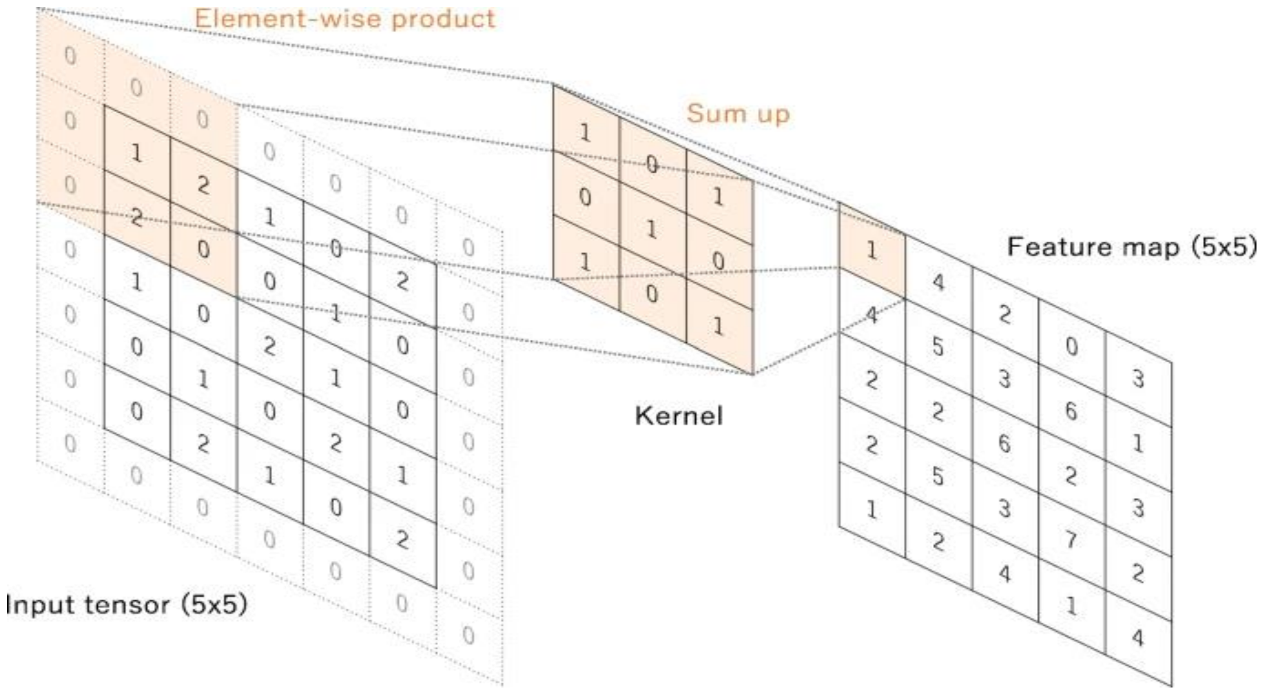


Fig. 2 Convolution Process

The convolution operation mentioned above does not cause the centre of each kernel to overlap with the outermost part of the input tensor and decreases the height and width of the output feature map relative to the input tensor. Padding, usually zero padding, is a technique used to resolve this problem, where rows and columns of zeros are applied to either side of the input tensor to match the middle of the kernel to the outer portion and to preserve the same in-plane dimension via the convolution process. Current CNN architectures typically use zero padding to maintain in-plane dimension to apply more layers. Excluding zero padding, each subsequent feature map will be relatively small just after convolution process.

NONLINEAR ACTIVATION FUNCTION

Linear function outputs like convolution are then transferred into a nonlinear activation function. While smoother nonlinear functions, like sigmoid and perhaps hyperbolic tangent (tanh) function, have been used frequently since they are mathematical representations of biological neuron behaviour, the much more common nonlinear activation function currently used is rectified linear unit (ReLU) which simply calculates the function: f(x) = max (0, x) Shown in Fig 3.

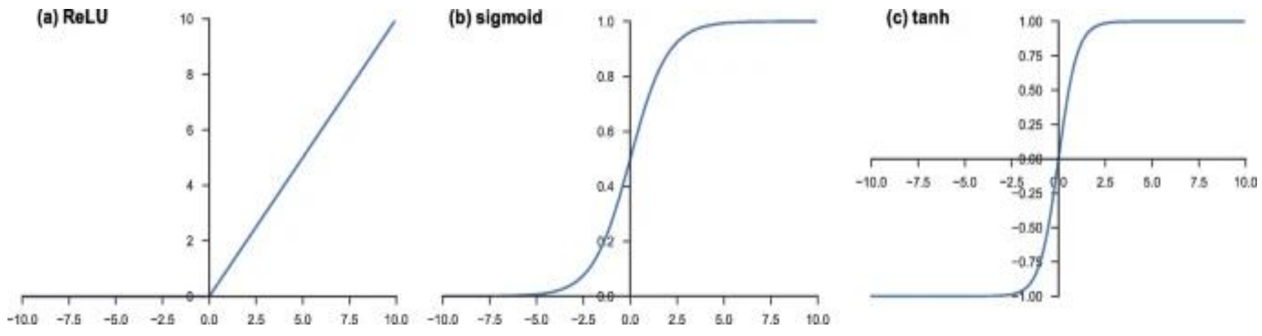


Fig. 3 Non-Linear Activations

POOLING LAYER

The pooling layer provides a standard down sampling operation that decreases the in-plane dimensions of the feature maps by adding a translation invariance of minor shifts and distortions and by decreasing the number of corresponding learning parameters. It should be remembered that there can be no learnable parameter in any one of the pooling layers, while filter size, padding and stride are hyperparameters in pooling operations, close to convolution operations.

MAX POOLING

The most common method of pooling operation is max pooling, which captures patches from the input function maps, generates the highest value for every patch, and rejects all the other values. Max-pooling with such a filter of size 2 × 2 with a stride of 2 is widely used in practice. It reduces the in-plane dimension of its feature maps by a ratio of 2. In relation to height and distance, the depth dimensions of the function maps stay intact.

FULLY CONNECTED LAYER

The output features of the final convolution or pooling layer are usually flattened, i.e., converted into a one-dimensional (1D) sequence of numbers (or vectors) and attached to one or more fully connected, often regarded as dense layers, in which each input is linked to another output via a learnable weight. When the features extracted from the convolution layers and sampled from the pooling layers are formed, a subset of completely connected layers is mapped to final outputs of the network, like the probabilities in each class throughout the classification tasks. The final fully connected layer normally will have the same no of output node as number of classes. Each completely connected layer is followed by a non-linear function, such as ReLU, as mentioned above.

LOSS FUNCTION

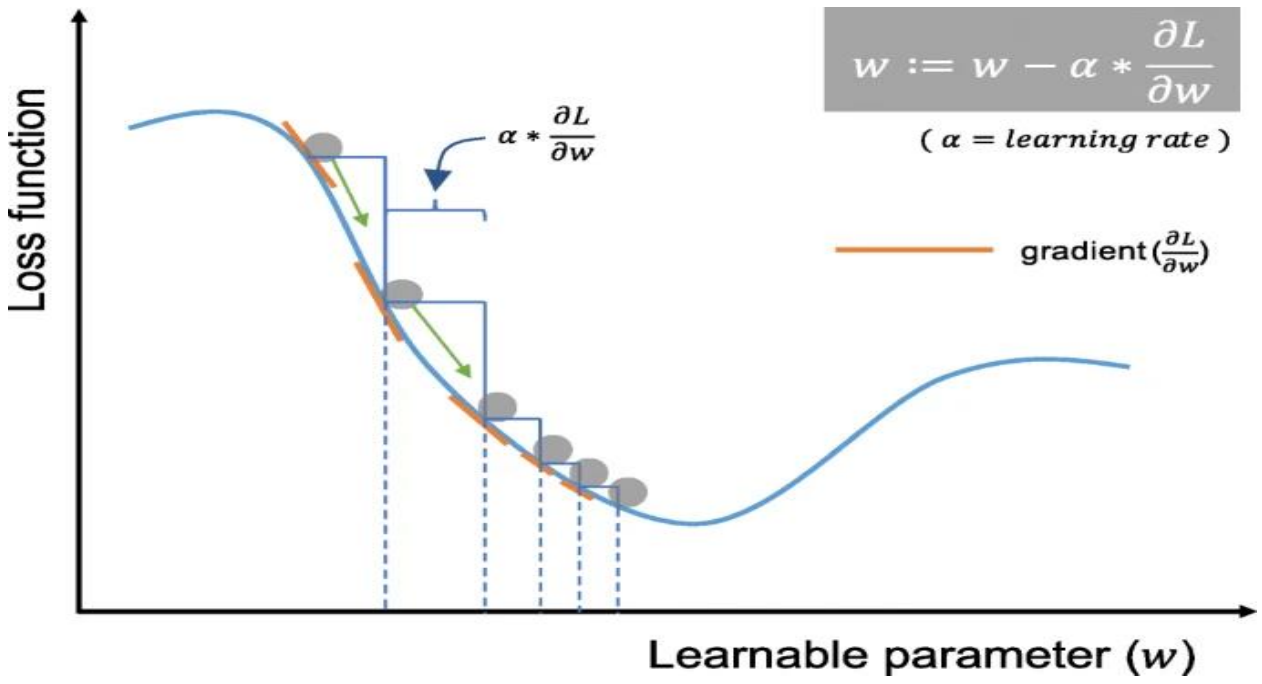
The loss function, often alluded to as the cost function, tests the compatibility between the network's output predictions through forward propagation and the given true labels. Cross-entropy is the widely used loss function for multiclass classification, while mean squared error is usually used for regression for continuous values. The form of loss function is among the hyperparameters and needs to be calculated as per the tasks.

GRADIENT DESCENT

Gradient descent is widely used as an optimization algorithm that recursively updates the learning parameters, i.e., kernels and weights, of the network to minimize losses. The gradient of the loss function gives us the path for which the function has the steepest rate of rise, so each learnable parameter is modified in the opposite direction of the gradient with such an arbitrary step size defined by the hyperparameter named the learning rate. The gradient is, computationally, a partial loss derivative for each learnable parameter, and a single parameter update is formulated. as follows:

$$w{:=}w-\alpha*\partial L\partial w \qquad (1)$$

From shown in equation (1) where w stands for each learnable parameter, α stands for a learning rate, and L stands for a loss function. Shown in Fig 4.



$w := w - \alpha * \frac{\partial L}{\partial w}$
$(\alpha = learning\ rate)$
$\alpha * \frac{\partial L}{\partial w}$
gradient $(\frac{\partial L}{\partial w})$
Loss function
Learnable parameter (w)

Fig.4. Gradient Descent

It should be remembered that, in fact, the learning rate is one of the most significant hyperparameters to be set before training begins. In reality, for purposes such as memory constraints, the loss gradients of the parameters are calculated using a training dataset subset called a mini-batch and added to changes of the parameters. This approach is called mini-batch gradient descent, often generally referred to as stochastic gradient descent (SGD), and mini-batch size is indeed a hyperparameter. In addition, several modifications to the gradient descent algorithm have been suggested and are commonly used, such as SGD with momentum, Adam, RMSprop.

IMAGE DENOISING USING NEURAL NETWORKS

Deep learning has been improved over the past few decades, especially CNN has gained more popularity on the image analysis tasks. Using CNN's for denoising is one of the tasks which researcher trying to implement. We have mentioned some of the deep learning methods used for denoising and their drawbacks. Some of them are Multi-layer perceptron (MLP), Trainable nonlinear reaction diffusion (TNRD), denoising convolutional neural networks (DnCNNs), fast and flexible denoising convolutional neural (FFDNet) are some of them. Kingma and Ba had proposed the use of Convolutional Neural Networks (CNNs) for image denoising. They have found that CNN has performed better than MRF model. Multi-layer perceptron (MLP) has been applied for the task of image denoising and produced good results. Auto encoders have been used for process of removing gaussian noise from the image and produced comparable results than K-SVD. Trainable nonlinear reaction diffusion (TNRD) is one of the best models proposed for denoising, it is a feed-forward network learning the noise removal strategy using gradient descent, by iterating over the input image. Multi-layer perceptron (MLP) and Trainable nonlinear reaction diffusion (TNRD), has achieved better performance than BM3D model. These above models have been trained for a certain low noise level. Denoising convolutional neural networks (DnCNN) uses image patches to remove the gaussian noise present in the image which has produced good results for gaussian noise. DnCNN is not generalized for high noises, which resulted in loss of quality in the input image and it has been trained on predefine noise level. Fast and flexible denoising convolutional neural (FFDNet) uses the approach of blindly finding the noise in the input image without knowing the noise, but it results in high computational power and separate models for each noise level. Many methods have been suggested over the last few decades to solve RVIN denoising issues. Switching-based algorithms were used, and the denoising method was highly reliant on the detection data. These algorithms are iteratively applied to achieve higher detection accuracy and improved denoising efficiency, resulting in long execution times. ANN, for example, is a training-based denoising algorithm with a high execution efficiency and good results. However, their detection accuracy is still lacking, which could result in image loss during the denoising phase.

IMAGE DENOISING USING FILTERS

The Standard Median Filter (SMF) is a fundamental nonlinear filter for Salt and Pepper Noise removal, but it is only efficient at low noise densities and fails entirely at high noise densities. A solution was formulated in the form of the Center-Weighted Median Filter (CWMF) and the Weighted Median Filter. By giving more weight to certain selected pixels in the filtering window, the median filter's output can be increased. This filter, on the other hand, does not differentiate between noise-free pixels and noisy pixels that cause artefacts. By adaptively adjusting the size of the local window, the Adaptive Median Filter (AMF) works well at low and medium noise densities. The downside is that the image suffers from a loss of desirable information as well as increased computational complexity. The Decision Based Algorithm (DBA) was proposed to solve these pitfalls. A set 3x3 window processes an image in this technique. If the pixel being processed is neither 0 nor 255, it is left alone; otherwise, it is replaced with the local window's median value. The median value would be noisy at high noise densities. In this scenario, the substitution pixel is a neighbouring previously processed middle pixel. A streaking effect can result from the continuous replacement of these neighbouring pixel values. To solve this artifact, the Modified Decision-Based Unsymmetrical Trimmed Median Filter (MDBUTMF) was proposed. Unsymmetrical trimming of 0s and 255s is done on both sides of the ordered 1D array of the local window in this technique. The trimmed array's median value is used as the substitution value. As a special case, at high noise densities, the chosen window may only contain 0s and 255s. In this case, the trimmed array's median cannot be found, so the pixel in question is replaced with the local window's mean value. For windows of all 0s or all 255s, the above algorithm fails since the mean value is either 0 or 255. To exclude high-density SPN, a new adaptive weighted mean filter (AWMF) has been suggested. It employs a filtering window with a full size of 39 X 39 pixels. It produces better qualitative and quantitative results, particularly when there is a lot of noise. To extract high-density SPN, an iterative adaptive fuzzy filter (IAFF) is proposed. It can handle up to 97 percent of noise density, with improved performance coming from a higher number of iterations. However, these models take a long time to compute. To exclude high-density SPN, an optimal weighted average filter (EWAF) has recently been proposed. By adaptively adjusting the filtering window duration, it achieves improved noise reduction and edge protection in high-noise settings. Rather than filtering pixels by pixels, it calls the entire noisy image.

RESIDUAL LEARNING AND BATCH NORMALIZATION FOR DENOISING

Convolutional neural networks have recently demonstrated tremendous progress in performing different vision challenges, owing to quick access to large-scale datasets and developments in deep learning techniques. Rectified Linear Unit (ReLU), tradeoff between depth and distance, parameter initialization, gradient-based optimization algorithms, batch normalisation, and residual learning are examples of notable accomplishments in training CNN models. Other factors that contribute to CNN's performance include the successful deployment on modern GPUs. It is simple to produce enough training data for Gaussian and salt-pepper denoising from a collection of high-quality images.

RESIDUAL LEARNING

Deep neural networks in integrate low, middle, and high-level features by using multiple stacked layers. Increasing network depth enables deep networks to learn complex nonlinear functions with an obstacle of vanishing/exploding gradients. Hence adding more layers to deep models leads to higher training error. The training accuracy of deep learning models may degrade along with increasing network depth. The residual learning in convolutional neural network helps to reduce training error by addressing vanishing/exploding gradient problems. Most of the discriminative models are trained by original mapping F(y) to obtain latent clean image. i.e.,f(y) ~ x. Residual mapping is much easier to learn than original mapping. Residual strategy helps identify noise distribution. Hence extreme deep networks can be modelled using residual strategy with less training error. The residual strategy helps to map residual image using hidden layers. The residual image is obtained by removing the clean image using non-linear mapping functions in the hidden layers. Highly deep CNN can be easily trained using such a residual learning technique, and increased accuracy for image recognition and object detection has been achieved. The residual learning formulation is also used in the proposed modified DnCNN model. Our DnCNN uses a single residual unit to predict the residual image, unlike the residual network, which uses multiple residual units. Prior to the residual network, the technique of predicting the residual image has already been used to solve certain low-level vision problems like single image super resolution and colour image denoising. Let h(x) denote desired/ original mapping function of noisy image (x). we can also formulate a nonlinear function f(x):=h(x)-x; i.e., thus original mapping function can be written as h(x)= f(x)+x. The hypothesis of residual strategy it is easier to optimize residual mapping easier f(x)+x that original mapping h(x). The function f(x)+x can be realized by skip connection/short cut connection. The idea of residual learning can be applied to image denoising. Let x,v and y be clean image, noise vector and noisy image respectively. We can model the noise as y = x +v, many deep learning models uses original mapping h(x) =x to predict latent clean image. In residual strategy we train residual mapping to find N(y) = v; i.e., x= y-N(y). The below Fig 5 shows about residual learning.

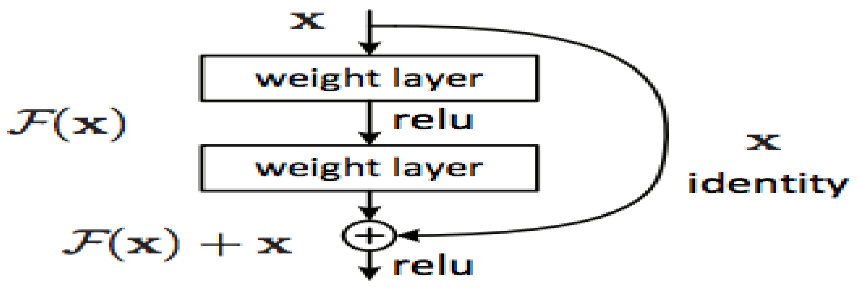
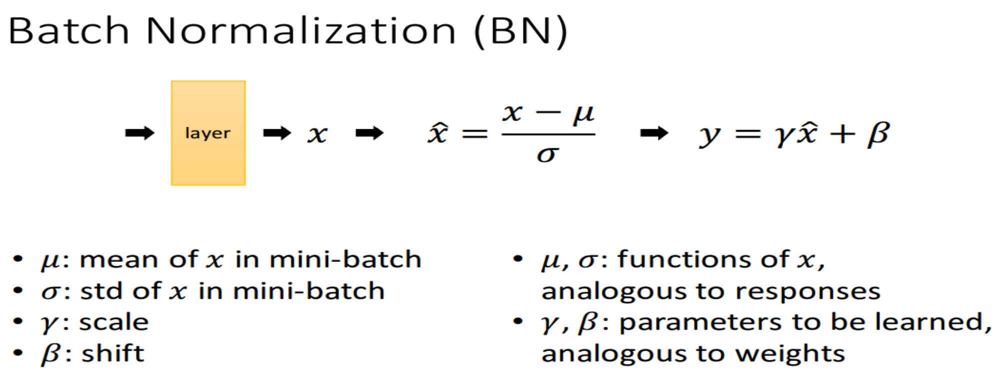


Fig.5 Residual Learning

BATCH NORMALIZATION

For training CNN models, Mini-batch stochastic gradient descent (SGD) is commonly used. Internal covariate shift, i.e. shifts in the distributions of internal nonlinearity inputs during training, reduces the training performance of mini-batch SGD despite its versatility and efficacy. Batch normalization is a method for reducing internal covariate shift by including a normalization step, as well as a scale and shift step, before each layer's nonlinearity. We have found that after including the residual and batch normalization method in the architecture, the model performed better producing faster results, stable training process, The below Fig 6 shows about batch normalization.



Batch Normalization (BN)
layer
x
$\hat{x} = \frac{x-\mu}{\sigma}$
$y = \gamma\hat{x} + \beta$
• μ: mean of x in mini-batch
• σ: std of x in mini-batch
• γ: scale
• β: shift
• μ, σ: functions of x, analogous to responses
• γ, β: parameters to be learned, analogous to weights

Fig.6 Batch Normalization

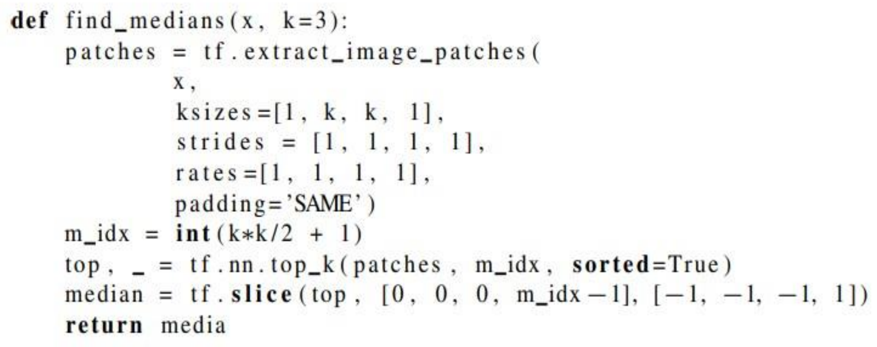
MEDIAN FILTER

The median filter is a nonlinear filter that is particularly good at filtering out impulse noise. It uses the window's median to replace the pixel that is located at the centre window. Applying a median filter to a heavily contaminated picture reduces spikes and increases the signal-to-noise ratio dramatically. Applying a 5×5 median filter twice or more eliminates approximately 50% and 90% of noise, respectively. Repeatedly applying the median filter until all the spikes in the image are replaced using the median value of a fixed-size window is a conventional idea. The noise is removed, but the signal is not recovered. In the first few steps, the Peak Signal to Noise Ratio (PSNR) rises, but then falls as the picture becomes blocky and fuzzy. This phenomenon means that the median filter deviates the signal too much from its original form, which is one of the key reasons why modern researchers no longer use the median filter to denoise S&P noise. The fundamental concept is to keep the standard median filter's ability to remove spikes when attempting to restore the degradations it causes.

Median filtering is implemented as a neural network operation and conducted it on the feature channels instead of explicitly applying median filters to the images. We basically delete spikes in different feature spaces this way, then combine the de-spiked features to predict a better noise-reduced picture. At the one end, median filter applied in between the layers acts as a switch filter like in the traditional methods, on the other end de-spike ability produced by the median filters allows the gradients to pass through the non-noisy pixels.

MEDIAN LAYER DEFINITION

In a moving window fashion, the median filter is added to each part of a feature channel. An RGB-channelled input picture, for example, corresponds to three feature channels; however, a collection of features generated after convolution typically contains many channels. We first extract a series of given size (3 × 3, 5 × 5, ...) patches centred at each pixel for each feature channel. After that we will find the median of every sequence that is formed by the elements in the patch. We have implemented the median layer, which will be inserted in between the model. It is implemented using TensorFlow using python, the input tensor is denoted as ‘x’ and ‘k’ denotes the kernel size for the median filter. The code can be seen in Fig 7.



```
def find_medians(x, k=3):
    patches = tf.extract_image_patches(
            x,
            ksizes=[1, k, k, 1],
            strides = [1, 1, 1, 1],
            rates=[1, 1, 1, 1],
            padding='SAME')
    m_idx = int(k*k/2 + 1)
    top, _ = tf.nn.top_k(patches, m_idx, sorted=True)
    median = tf.slice(top, [0, 0, 0, m_idx-1], [-1, -1, -1, 1])
    return media
```

Fig.7 Median Layer

The median layer will be applied on each feature channel that is outputted by the previous convolutional layer and after applying median filter to those feature channels they will concatenated at the end. And new set of feature maps are formed. If there are

64 feature channels outputted by the convolutional layer, then there will be 64 median outputs.

MODIFIED DnCNN ARCHITECTURE:

In this section, a modified DnCNN model is proposed with median layers in between the architecture and some additional layers are added at the end. This modified DnCNN model is used for removing the salt and pepper noise in the input image. We have used residual learning and batch normalization for faster training and stability of the model. The receptive field size of denoising neural networks has been shown to align with the efficient patch size of denoising methods. Furthermore, a high level of noise necessitates a greater successful patch size to collect more background details for reconstruction. As a result, we study the efficient patch size of many leading denoising methods to direct the depth architecture of our DnCNN by fixing the noise level. The non-local equivalent patches are adaptively scanned for two times in a local widow of size 25×25 in BM3D, resulting in a final successful patch size of 49×49. WNNM, like BM3D, uses a wider scanning window and iteratively performs non-local searching, which results in a larger patch size (361×361). MLP generates the predicted patch with a patch of size 39×39, then averages the output patches with a filter of size 9×9, resulting in an effective patch size of 47×47. With five phases, the CSF and TNRD have a total of ten convolutional layers with a filter size of 7×7 and an efficient patch size of 61×61.Thus patch size varies for different methods according to the noise level they use. Thus, for salt and pepper denoising, we have used a patch size of 40×40 with a depth of 20 layers.

The input to the modified DnCNN model will be the patches that has been produced by a image size of 256×256. The input will be a noisy observation $y = x + v$ where 'v' is the noise in the input image. Methods like MLP and CSF aim to learn the clean image using neural network $F(y) = x$. But for DnCNN we will try to find the residual mapping $R(y) \approx v$ using residual learning formulation, and then we will get the clean image $x = y - R(y)$. From equation (2). The averaged mean squared error between the desired residual images and the approximate ones from noisy data, in formal terms.

$$\ell(\Theta) = \frac{1}{2N}\sum_{i=1}^{N} \|\mathcal{R}(\mathbf{y}_i;\Theta) - (\mathbf{y}_i - \mathbf{x}_i)\|_F^2 \tag{2}$$

This can be used as a loss function to learn the trainable parameters in DnCNN. In the above equation y and x are the noisy and clean image patches. In this modified DnCNN we have four different layers, namely, convolution layer with ReLU, convolution layer with batch normalization and with ReLU, median filter applied on the tensors, only convolutional layer. The convolution layer with ReLU will be the first layer which has 64 filters of 3 × 3 × c is used to output 64 feature maps, and for non-linearity rectified linear units (ReLU, max(0,·)) is used. In this 'c' represents the number of channels in the input image, where c = 1 denotes gray image and c = 3 denotes color image. Second layer will be the median layer which will apply median filter for the image patch after the first convolutional layer. The output of the median layer will be of tensor format. The kernel size of median layer is 3. Two consecutive median layers are used. Third layer will be a stack of convolutional layers with batch normalization for normalizing the parameters and ReLU for activation. For the convolution process kernel size of 3 × 3 is used with the stride size of 1 × 1. The kernels are initialized using orthogonal initialization and same padding is used. Batch normalization is used for each convolutional layer in this stack of layer with a epsilon value of 0.0001. ReLU activation is used after the convolutional layer. This layer has 64 filters which will output a size of 3 × 3 × 64. Our modified has used median filter, residual learning for learning the residual noise and batch normalization to boost the training process and denoising task. DnCNN will increasingly distinguish image structure from noisy observation by hidden layers by integrating convolution with ReLU. In certain low-level vision systems, the output image size must match the input image size. This could result in boundary artifacts. In MLP, the noisy input image's boundary is symmetrically padded in the pre-processing stage, while in CSF and TNRD, the same padding technique is used before each stage. Unlike the previous approaches, we pad zeros immediately before convolution to ensure that each feature map in the middle layers is the same size as the input image. We discovered that using a basic zero padding technique produces no boundary objects. Either the initial mapping F(y) or the residual mapping R(y) can be used to train the network to predict x or v. The residual mapping would be much easier to optimise if the original mapping is more of an identity mapping. It's worth noting that the noisy observation y resembles the latent clean image x even better than the residual image v. (especially when the noise level is low). As a result, F(y) is more like an identity mapping than R(y), and the residual learning formulation is better for image denoising. Batch normalisation benefits residual learning on the one hand.

This is simple since batch normalisation has certain advantages for CNNs, such as removing the issue of internal covariate transfer. Batch normalisation, on the other hand, results from residual learning. Batch normalisation can be used for residual learning to speed up the training and improve efficiency. The input intensity and convolutional function are associated with their neighbours without residual learning, and the distribution of the layer inputs is also dependent on the content of the images in each training mini-batch. DnCNN indirectly eliminates the latent clean picture with the operations in the hidden layers with residual learning. As a result, residual learning will aid batch normalisation in minimising internal covariate transition. To summarise, combining residual learning and batch normalisation will improve denoising efficiency while both speeding up and stabilising the training process. In the training process we have used different noise level and trained the model for 30 %, 60% and 90%. Our method can remove the S&P using median layers. This method can be used for various noise strategies. By giving the images as patches the training time has been reduced. From results obtained, it is learned that our DnCNN model has produced better results for salt and pepper noise. The below Fig 8 shows the DnCNN architecture.

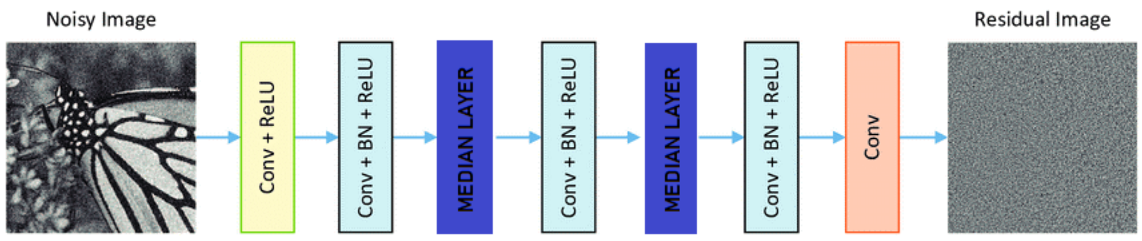


Fig 8 Architecture of modified DnCNN architecture

The above fig shows the architecture of modified DnCNN. Median layers are added between the DnCNN model. It has been trained to find the residual image from the noise image. The architecture has a total parameter count of 557,513, with trainable parameter count 555,593 and non-trainable parameter count 1,920.

CNN FOR CLASSIFICATION

In a particular time, deep learning differs in the capacity to assess the correctness of the inference. This is the need for every study of the image processing since they have to determine on their own if the classification is right. Like classifying and distinguishing between a signal post and a pedestrian when in a self-driving vehicle. The key reason why CNN is used for the classification of images rather than any other accessible machine learning methodology is that it automatically extracts the essential features required for a better classification. CNN has experienced numerous developments in recent years, which have contributed to the creation of state-of-the-art architectures. Often used CNN architectures include AlexNet, VGG Net, Inception, ResNet. In this article, we chose VGG 19 architecture over other architectures because it achieves better results with a limited number of images.

The consistency of the model depends on its complexity, which also depends on the availability of the dataset. Less datasets used in a dynamic model yield less accuracy. Normally, other CNN models use more datasets on a complex model to produce reliable performance. VGG 19 produces greater accuracy with a considerable volume of data set. Shown in Fig 9

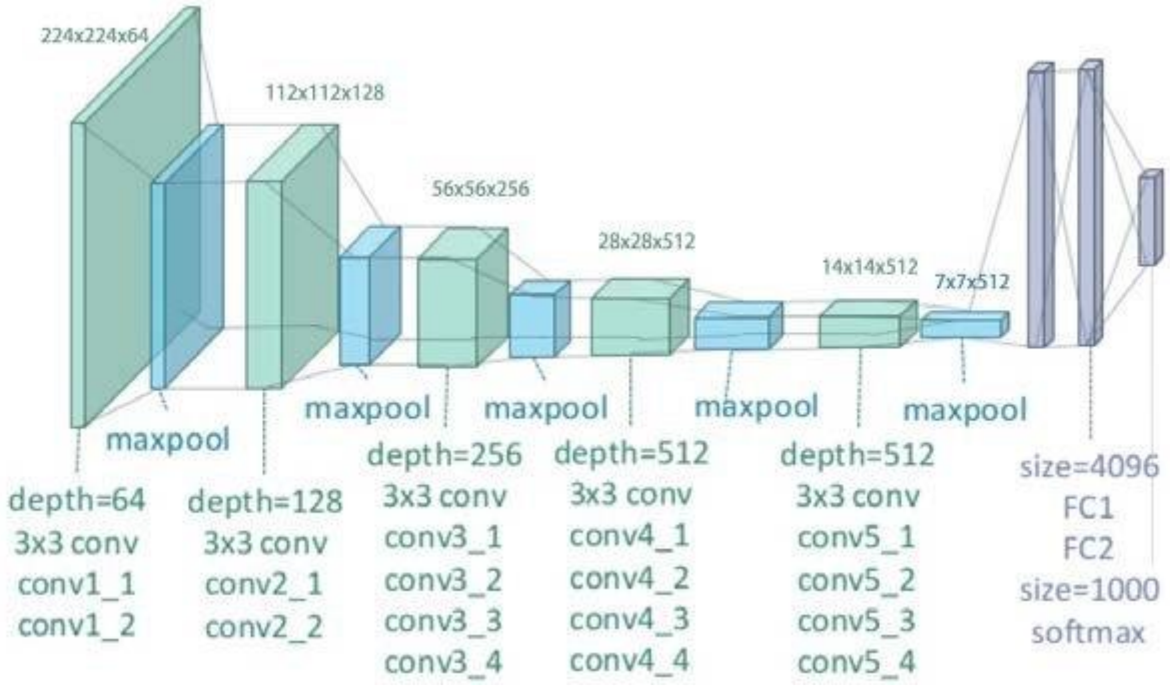


Fig 9. VGG-19 Architecture layout

VGG19 ARCHITECTURE

VGG19 architecture has 14 convolutional layers, 5 max pooling layer and 3 fully connected layer, which results in 143 M parameters. Input to the architecture is a 224 * 224 * 3 image. 224 is the number of pixels and 3 is the RGB layers of the image. The first two convolutional layers uses 3* 3 size filter with 64 filters, so it would result in 224 * 224 * 64 volume. Since we are using same convolution the height and width are Table 1. Parameters and Activations of the VGG19 architecture the same. Stride 1 is used for the convolution layer. Next is the pooling layer size is 2 * 2 with stride 2.

It will reduce the size of the input image; the resulting dimension will be 112 * 112 * 64. It is followed by 2 more conv layers with 128 filters each. Since we have the same convolution, the new dimension is 112 * 112 * 128. Then the pooling layer is used to reduce the dimensions of the conv layer resulting in the dimension of 56 * 56 *128. It is followed by 2nd conv layer with 256 filters each resulting in dimension of 56 * 56 * 256. Then a pooling layer is used of the same size as before to reduce the height and width. The new dimension obtained is 28 * 28 * 256. It is then passed through 3rd conv layers of filter size of 3 * 3 with 512 filters. The dimension of 28 * 28 * 512 image is then passed through a pooling layer to reduce the size into half. Then, 14 * 14 * 512 is passed through 4th conv layer of 3 * 3 filter size and stride 1 of same convolution with 512 filters. It is further followed by a pooling layer of size 2 * 2 with stride 2. The dimension obtained after pooling layer is 7 * 7 * 512. At the end it passed into fully connected layer (FC). FC layer has 4096 units. Which is followed by 2 fully connected layers of 128 and 64 units. Then passed through the last layer, which is softmax layer with 4 classes. The parameters and activations of the VGG19 architecture is shown below in the table 1.

TRANSFER LEARNING

Public deep learning algorithms with an ImageNet database (http://www.image-net.org) capable of classifying 1000 natural objects have been used to train an otoendoscopic image model. Among several publicly accessible deep learning models, Alexnet, GoogLeNet, ResNet(ResNet18, ResNet50, ResNet101), Inception-V3, Inception-ResNet-V2, SqueezeNet, and MobileNet-V2 have been used and compared because these network models are considered to demonstrate higher accuracy results compared to any other network with comparable prediction times. To see the output for online computing, smaller-scale network models were also used. Shown in Table 1. When transferring layers in existing models to new models, we swapped the very last fully connected layer of each model with a new fully connected layer with four output nodes, with softmax activation function at the end.

Training was performed using the Adaptive Moment Estimation (ADAM) with a batch size of 50, a maximum epoch of 20 and an initial learning rate of 0·0001. The initial learning rate of 0·0001, which might sound poor, was chosen based on our knowledge that the usage of the traditional learning rate of 0·01 to 0·001 did not converge in the current application.

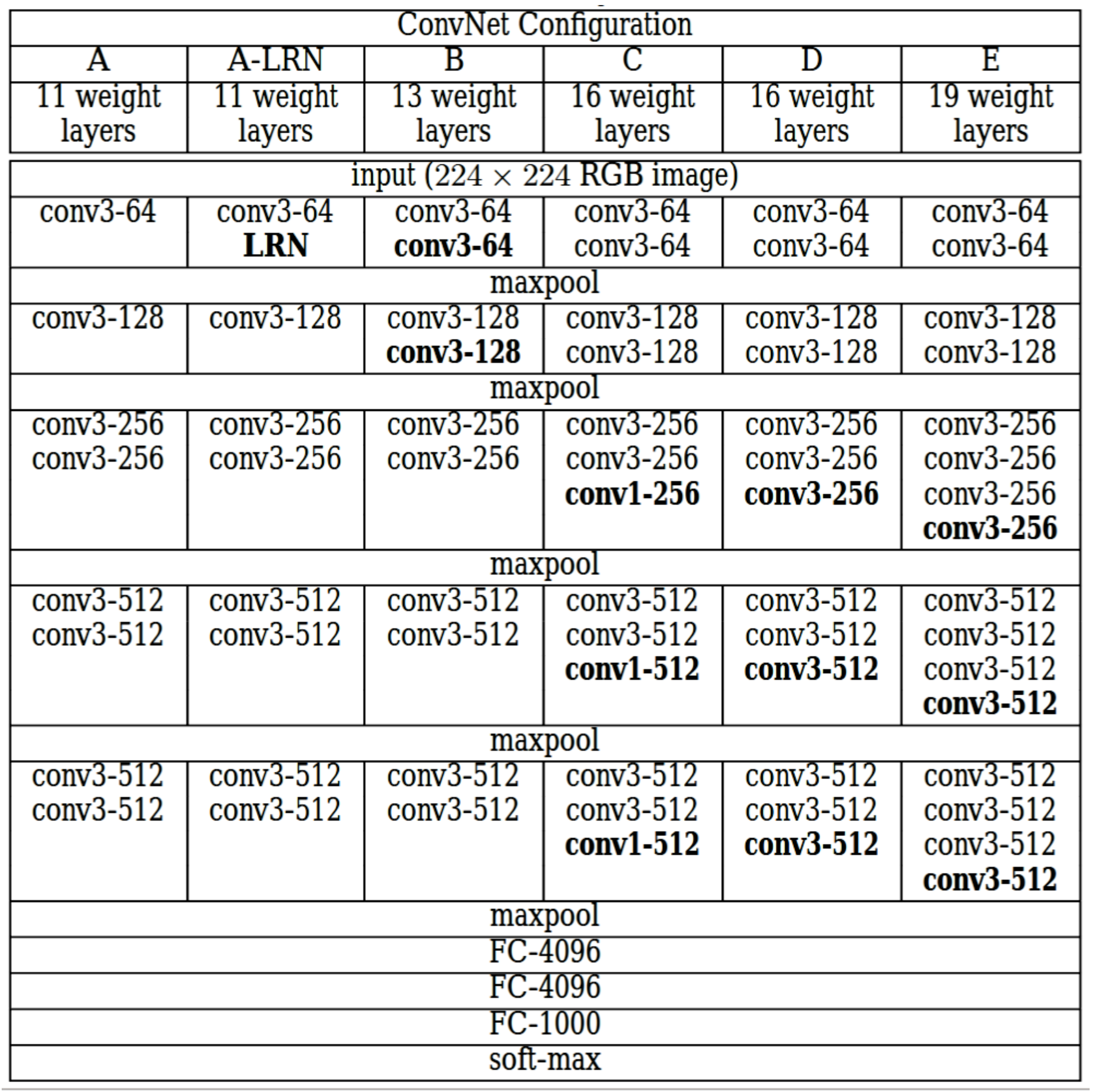
Layer type		**#Filters**	**# Parameters**	**# Activations**

Throughout computer vision, transfer learning is typically expressed using pre-trained models. A pre-trained model that is trained on a broad benchmark dataset to solve a problem close to the one that we want to solve. Consequently, due to the computational expense of training such models, it is standard practice to import and use models from existing literature (e.g., VGG, Inception, MobileNet). Canziani et al. are providing a thorough analysis of the success of pre-trained models on computer vision issues using evidence from the ImageNet challenge (Deng et al. 2009). (2016). Several pre-trained models used for transfer learning are based on a broad CNN (Voulodimos et al. 2018). CNN has been found to succeed in a wide variety of computer vision tasks (Bengio 2009). Its high performance and ease of training are two of the key reasons driving CNN's success in recent years.

	Input			150K
Conv1_1		64	1.7K	3.2M
Conv1_2		64	36K	3.2M
	Max pooling			802K
Conv2_1		128	73K	1.6M
Conv2_2		128	147K	1.6M
	Max pooling			401K
Conv3_1		256	300K	802K
Conv3_2		256	600K	802K
Conv3_3		256	600K	802K
Conv3_4		256	600K	802K
	Max pooling			200K
Conv4_1		512	1.1M	401K
Conv4_2		512	2.3M	401K
Conv4_3		512	2.3M	401K
Conv4_4		512	2.3M	401K
	Max pooling			100K
Conv5_1		512	2.3M	100K
Conv5_2		512	2.3M	100K
Conv5_3		512	2.3M	100K
Conv5_4		512	2.3M	100K
	Max pooling			25K
	FC6		103M	4K
	FC7		17M	4K
	OUTPUT		4M	1K

Table.1 VGG Parameters

The dataset consists of 1880 RGB images among which each category consists of 470 images. The CNN architecture classifies the images into 4 categories namely Normal, Myringosclerosis, Chronic otitis media, Earwax plug. The preprocessing step involves basic contrast adjustments and image denoising techniques to make the image readable. the green channel of the RGB image is segregating separately for further processing. The input image is of 224 * 224 of size. The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv (receptive field size)-(number of channels)”. Shown in Table 2.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table.2 ConvNet Description

EFFECTIVE PREDICTION OF PLANT DISEASE USING DEEP LEARNING ALGORITHM

S.P. AARTHI, M. ABIRAMI, S. NANTHINI

Agricultural productivity is a key component of Indian economy. Therefore, the contribution of food crops and cash crops is highly important for both the environment and human beings. More number of research has been done based on detecting disease of plant by processing its leaf. However, detecting disease after occurrence of it and identifying solution and source for it presence is a delay and useless process. The occurrence of disease in plants is due to absence of nutrient content in it. In our work, identification of plant leaf disease has been implemented using Deep learning technique. A plant grows in healthy way when it contains basic nutrients such as Nitrogen, phosphorus, potassium etc. It provides benefits in monitoring the large crop fields and helps in detecting the symptoms of the disease when they are found on the leaves. In this paper, the primarily focus on finding the plant diseases and which will reduce the crop loss and hence increases the production efficiency. Our proposed work detects the symptoms of plant diseases at the very initial stage and classifies plant disease based on the symptoms using a Deep Learning (DL) technique. The proposed approach recognizes the diseases using a CNN.

In recent years, agriculture has become much more important than it used to be some years back where plants were only used to feed humans as well as animals. This is since plants are now used to generate electricity and other sources of energy to improve upon the living conditions of mankind. However, there are so many diseases that affect plants that can cause great harm to various economies and societies. It can even lead to great ecological losses. For this reason, it is better to diagnose diseases accurately and timely to avoid such loses. Plant diseases can be detected through several means including manual and computer-based systems. Most plant diseases appear as spots on the leaves which are more visible to human eye. On the other hand, there are some diseases that do not appear on the leaves and others appear in the later stages when they have already caused great harm to the plants.

In such instances, it is recommended that computerized systems would be the only option to detect the situation timely using complex algorithms and analytical tools, preferably using powerful microscopes and other machines. In some other instances, the signs can only be detected through the electromagnetic means which produces more images that are not visible to the human eye. India is a cultivated country and about 70% of the population depends on agriculture. Farmers have large range of diversity for selecting various suitable crops and finding the suitable pesticides for plant. Disease on plant leads to the significant reduction in both the quality and quantity of agricultural products. The studies of plant disease refer to the studies of visually observable patterns on the plants. Monitoring of health and disease on plant plays an important role in successful cultivation of crops in the farm. In early days, the monitoring and analysis of plant diseases were done manually by the expertise person in that field. This requires tremendous amount of work and requires excessive processing time. The image processing techniques can be used in the plant disease detection. In most of the cases disease symptoms are seen on the leaves, stem, and fruit. The plant leaf for the detection of disease is considered which shows the disease symptoms.

PROPOSED SYSTEM

Plants are susceptible to several disorders and attacks caused by diseases. There are several reasons that can be characterizable to the effects on the plants, disorders due to the environmental conditions, such as temperature, humidity, nutritional excess or losses, light and the most common diseases that include bacterial, virus, and fungal diseases. Those diseases along with the plants may shows different physical characteristics on the leaves, such as a change in shapes, colours etc. Due to similar patterns, those above changes are difficult to be distinguished, which makes their recognition a challenge, and an earlier detection and treatment can avoid several losses in the whole plant. The proposed plant disease prediction method takes input from the plant's leaves images. Initially pre-processing is implemented for a given dataset where size and noise in images are removed. Feature extraction is the next step this extracts the required features from the image and undergone the process classification. Here CNN is the deep learning algorithm used for accurate prediction of leaf disease prediction. Our system achieves better solution compared to existing methods.

INCEPTION V3

It is an extended network of the popular GoogLeNet which has achieved good classification performance in several biomedical applications using transfer learning. Following GoogLeNet, inception-v3 proposed an inception model which concatenates multiple different sized convolutional filters into a new filter. Such design decreases the number of parameters to be trained and thereby reduces the computational complexity. The basic inception-v3 is illustrated in Figure.1.

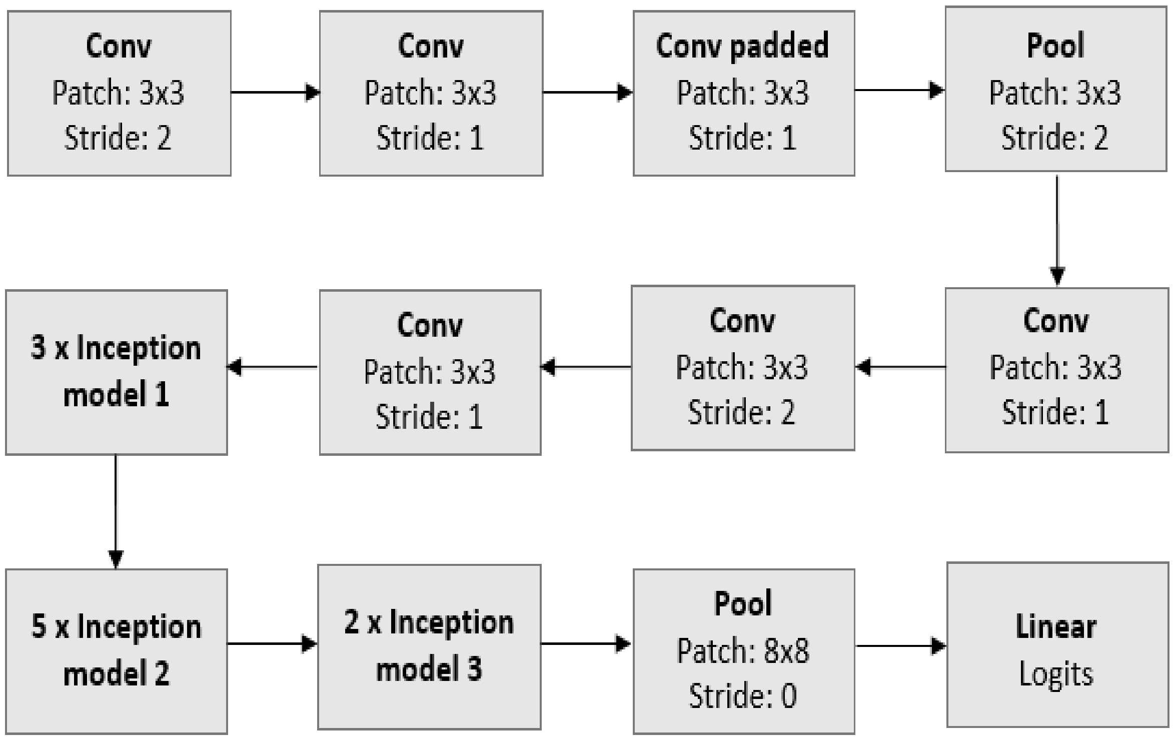


Figure 1: Block Diagram of System Architecture

Inception networks proved to be computationally efficient in terms of computing resource utilization as well as the number of parameters. However, the downside of the original inception network was its limited application adaptability in new use cases. the original inception network model by introducing factorized convolutions with large filter size, factorization into smaller convolutions, and asymmetric convolutions.

As a pre-processing step, we used histogram equalization to increase the contrast. In addition, the input images are resized to match the requirements of the individual network

It is a convolutional neural network that is 42 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database.

ADVANTAGES

- Accurate detection of leaf disease using deep learning algorithm.
- Increases performance of the system.
- Reduces cost and computation power

SEQUENTIAL DIAGRAM

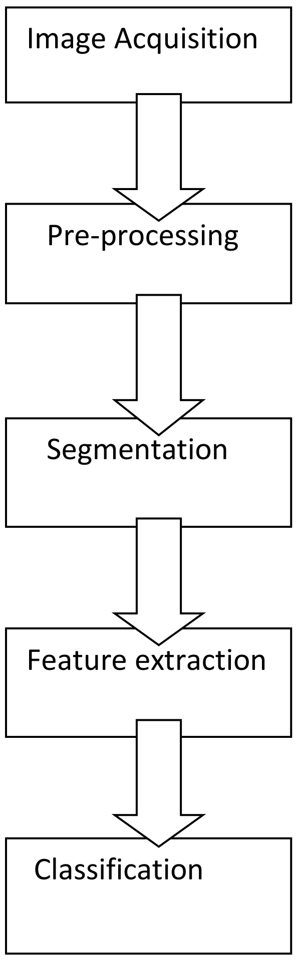


Figure 2: Sequential Diagram of Implementation Process

IMAGE ACQUISITION

Input image samples are containing leaf images are collected and stored. Collection of the leaf images are done using digital camera and internet sources. Collected leaf sample contains the infected plant leaves with Black rot and Black measle diseases and Quick Wilt diseased leaves of pepper, Rice blast disease of paddy and the healthy leaves of above spices. Appropriate datasets are required at all stages of object recognition research, starting from training phase to evaluating the performance of recognition algorithms. All the images collected for the dataset were downloaded from the Internet, searched by disease and plant name on various sources. To distinguish healthy leaves from diseased ones, one more class was added in the dataset. It contains only images of healthy leaves. An extra class in the dataset with background images was beneficial to get more accurate classification. Thus, deep neural network could be trained to differentiate the leaves from the surrounding.

PRE-PROCESSING

To improve the image quality, suppress the undesired distortion and for noise removal, pre-processing is done on the gathered images. The input images will be of different sizes which are initially resized to 256x256 pixels to maintain uniformity. Images downloaded from the Internet were in various formats along with different resolutions and quality. In order to get better feature extraction, final images intended to be used as dataset for deep neural network classifier were pre-processed in order to gain consistency. Furthermore, procedure of image pre-processing involved cropping of all the images manually, making the square around the leaves, in order to highlight the region of interest (plant leaves). During the phase of collecting the images for the dataset, images with smaller resolution and dimension less than 500 px were not considered as valid images for the dataset. In addition, only the images where the region of interest was in higher resolution were marked as eligible candidates for the dataset. In that way, it was ensured that images contain all the needed information for feature learning. Images used for the dataset were image resized to 256×256 to reduce the time of training.

SEGMENTATION

Segmentation is performed after doing some pre-processing which works on basis for classification and also verified object detection. In CNN image segmentation is performed by applying filters. These filters are also used in feature extraction. This also involves giving or feeding input to the CNN model, where labelling of pixels takes place. It scans the image based on the input stride given, and this process continues until it has covered the whole image.

FEATURE EXTRACTION

After segmentation the features are extracted by applying the filters and also various features are extracted using various techniques of feature extraction. It specifically defines the diseased area on the basis of colour, form and textural features. Various extraction techniques, such as colour co- occurrence, skewness, comparison, similarity, etc., are used to obtain the desired set of features. It is a type of dimensionality reduction technique and as a compact feature vector the interesting parts of an image are represented efficiently. From the segmented image shape, texture and color features are extracted. Color features like color moments: Mean, Standard Deviation is extracted. Using Grey Level Co-occurrence matrix (GLCM) texture features including Homogeneity, Mean, Standard deviation, Variance and Entropy are extracted.

CLASSIFICATION

Classification is used to identify the class of the new observation using training data whose class label is known. It is performed by using three different machine learning techniques. For classification convolution Neural Network is used. CNN takes a varied approach towards regularization. It is less complicated than conventional models of regularization. The layers are described below.

Convolutional Neural Network, which is known as CNN, consists of three layers which is the Reference layer, the Middle layers, and the Output layer. The first layer which is the input layer is the one that recognizes the features as input, in other words, the images are provided as input through this layer. The second layer which is the middle layer consists of the necessary number of nodes on the basis of the program. The output layer produces the data. Convolutional Layer performs a convolutionary operation over the elements of the image matrix which is the pixel values in it along with the kernel matrix. The kernel matrix will move over the image matrix which is the pixel matrix and the value will be calculated. This is used to find out the final size of the filter map provided as output. The basic functional unit of a neural network is “neurons or perceptrons”.

There are basically four layers

(i) Convolution layer
(ii) Pooling Layer
(iii) Fully connected layer
(iv) SoftMax Layer

Convolution Layer: In convolution Layer the input image is multiplied with a filter Where basic feature extraction takes place.

Pooling Layer: It is used to reduce the spatial size representation and also it used for reduction in number of framework and calculation in the network.

Fully Connected Layer: Fully Connected layer works as multi-layer in forward direction where the output of one neuron is traversed over another as input to another neuron.

Softmax Layer: This layer consists of N x 1 matrix where the output data is interpreted by probability distribution function.

Department of Electronics and Communication Engineering

Vision

To be recognized by the society at large as a full- fledged department, offering quality higher education in the Electronics and Communication Engineering field with research focus catering to the needs of the stakeholders and staying in tune with the advancing technological revolution and cultural changes.

Mission

To achieve the vision, the department will

- Establish a unique learning environment to enable the students to face the challenges in Electronics and Communication Engineering field.
- Promote the establishment of centres of excellence in niche technology areas to nurture the spirit of innovation and creativity among faculty and students.
- Provide ethical and value-based education by promoting activities addressing the societal needs.
- Enable students to develop skills to solve complex technological problems and provide a framework for promoting collaborative and multidisciplinary activities.