IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

INTEGUMENTARY SYSTEM MALIGNANCY CLASSIFICATION USING DEEP LEARNING

Muthukumarasamy.S
Associate Professor
S.A. Engineering College
Chennai

Kokila. A
B.E.(CSE) Student
S.A. Engineering college
Chennai

Srija. R
B.E.(CSE) Student
S.A. Engineering College
Chennai

Indhuprabha. C
B.E.(CSE) Student
S.A. Engineering College
Chennai

Abstract -- Skin cancer is an abnormal growth of skin cells. It generally develops in areas that are exposed to the sun, but it can also form in places that don't normally get sun exposure. Skin cancers aren't all identical, and they may not cause many symptoms. Still, unusual changes to your skin can be a warning sign for the different types of cancer. Being alert for changes in your skin may help you get a diagnosis earlier. Accurate and precise diagnosis of diseases has been a significant challenge and the recent advances in computer vision made possible by deep learning have paved the way for disease diagnosis of skin cancer. It described the innovative solution that provides efficient disease detection and deep learning with convolutional neural networks (CNNs) has achieved great success in the classification of various skin cancer diseases. A variety of neuron-wise and layer-wise visualization methods were applied using a CNN, trained with a publicly available skin cancer disease given image dataset. So, it was observed that neural networks can capture the colors and textures of lesions specific to respective diseases upon diagnosis, which resembles human decision-making. And this model deploys the Django web framework .The procedure which are presented is very difficult and challenges must be addressed in the future.

Keywords – Disease Detection, Deep Learning, TensorFlow,CNN,Artifical Intelligence.

I. INTRODUCTION

Skin cancer is one of the most active types of cancer in the present decade. As the skin is the body's largest organ, the point of considering skin cancer as the most common type of cancer among humans is understandable. Skin cancer is cancer that emanates from the skin. Skin cancers are generally developed on the upper layer of the epithet dermis. This legion canto grows in size and shape which might invade other parts of the body.

Melanoma is one type of skin cancer responsible for creating malignant tumors on the skin. Skin cancer is detected by using dermatological photographs. CNN-based skin cancer detection where the feature is extracted from dermoscopic images using feature extracting techniques. They achieved an accuracy of detection of 89.5% in the testing phase. However, the accuracy of detection was not sufficient which was needed to improve. Skin cancer is the most common form of cancer, globally accounting for at least 40% of cancer cases. The most common type is nonmelanoma skin cancer, which occurs in at least 2–3 million people per year. However, this is a rough estimate as good statistics are not kept. Of nonmelanoma skin cancers, about 80% are basal-cell cancers, and 20% are squamous-cell skin cancers. Basal-cell and squamous-cell skin cancers rarely result in death. In the United States, they were the cause of less than 0.1% of all cancer deaths. Globally in 2012, melanoma occurred in 232,000 people and resulted in 55,000 deaths. White people in Australia, New Zealand, and South Africa have the highest rates of melanoma in the world. The three main types of skin cancer have become more common in the last 20 to 40 years, especially in regions where the population is predominantly white.

II. FEASIBILITY STUDY

A. Splitting the dataset:

The data used is usually split into training data and test data. The training set contains a known output and the model learns on this data to be generalized to other data later on. It has the test dataset (or subset) to model and it will do this using the Tensor flow library in Python using the Keras method.

B. Construction of a Detecting Model:

Deep learning needs data gathering and has a lot of past image data. Training and testing this model working and predicting correctly.

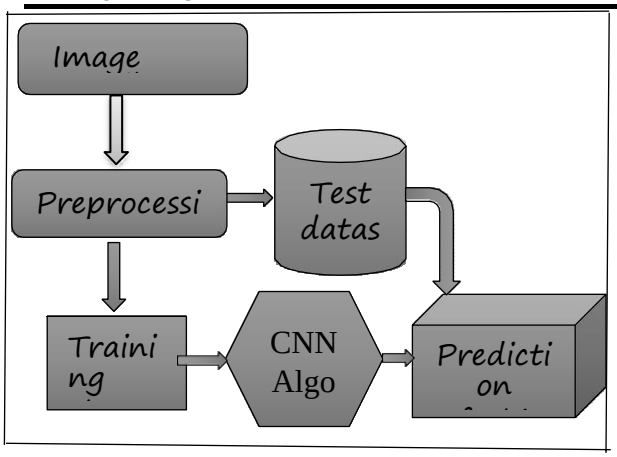


Fig1: Process of a dataflow diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or parallel, unlike a traditionally structured flowchart that focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top-down approach to Systems Design. Symbols and Notations Used in DFDs Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system, or a business system. They are also known as terminators, sources, and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, sort data based on logic, or direct the data flow based on business rules.

Datastore: files or repositories that hold information for later use, such as a database table or a membership form.

Data flow is the route that data takes between the external entities, processes, and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like "Billing details."

DFD levels and layers A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1, or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. A wide audience should easily understand it, including stakeholders, business analysts, data analysts, and developers.

C. Workflow diagram:

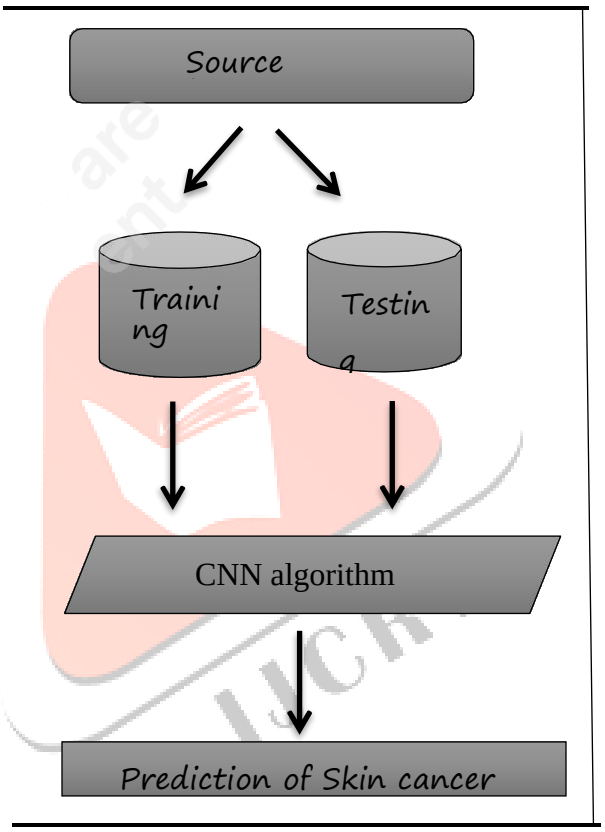


Fig2: Workflow Diagram

III. METHODOLOGY

Preprocessing and training the model (CNN): The dataset is pre-processed such as Image reshaping, resizing, and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 4 different retinal diseases is obtained, out of which any image can be used as a test image for the software.

The training dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the disease if the retinal image is contained in the dataset. After successful training and pre-processing, a comparison of the test image and trained model takes place to predict the disease.

IV. CNN Model steps:

A. Conv2d:

The 2D convolution is a fairly simple operation at heart: you start with a kernel, which is simply a small matrix of weights. This kernel "slides" over the 2D input data, performing an element-wise multiplication with the part of the input it is currently on, and then summing up the results into a single output pixel.

The kernel repeats this process for every location it slides over, converting a 2D matrix of features into yet another 2D matrix of features. The output features are essential, the weighted sums (with the weights being the values of the kernel itself) of the input features are located roughly in the same location as the output pixel on the input layer.

Whether or not an input feature falls within this "roughly same location", gets determined directly by whether it's in the area of the kernel that produced the output or not. This means the size of the kernel directly determines how many (or few) input features get combined in the production of a new output feature.

This is all in pretty stark contrast to a fully connected layer. In the above example, we have 5×5=25 input features and 3×3=9 output features. If this were a standard fully connected layer, you'd have a weight matrix of 25×9 = 225 parameters, with every output feature being the weighted sum of every single input feature. Convolutions allow us to do this transformation with only 9 parameters, with each output feature, instead of "looking at" every input feature, only getting to "look" at input features coming from roughly the same location. Do take note of this, as it'll be critical to our later discussion.

B. MaxPooling2D layer

Down samples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension.

The resulting output, when using the "valid" padding option, has a spatial shape (number of rows or columns) of: output shape = math. Floor ((input_shape - pool_size) / strides) + 1 (when input_shape >= pool_size)

The resulting output shape when using the "same" padding option is: output shape = math. Floor ((input_shape - 1) / strides) + 1

C. Input shape

If data_format='channels_last': 4D tensor with shape (batch_size, rows, cols, channels).

If data_format='channels_first': 4D tensor with shape (batch_size, channels, rows, cols).

D. Output shape

If data_format='channels_last': 4D tensor with shape (batch_size, pooled rows, pooled_cols, channels).

If data_format='channels_first': 4D tensor with shape (batch_size, channels, pooled rows, pooled_cols).

E. Flatten layer

Flatten the dimensions of the image obtained after convolving it. Dense: It is used to make this a fully connected model and is the hidden layer. Dropout: It is used to avoid overfitting on the dataset and dense is the output layer contains only one neuron which decides to which category image belongs.

Flatten is used to flatten the input. For example, if flatten is applied to a layer having input shape as (batch_size, 2,2), then the output shape of the layer will be (batch_size, 4)

Flatten has one argument as follows

(Keras. layers. Flatten (data_format = None))

The data format is an optional argument and it is used to preserve weight ordering when switching from one data format to another data format. It accepts either channels_last or channels_first as value. channels_last is the default one and it identifies the input shape as (batch_size, ..., channels) whereas channels_first identifies the input shape as (batch_size, channels, ...)

F. Dense layer

Dense implements the operation: output = activation (dot (input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use bias is True). These are all attributes of Dense.

For example, if the input has dimensions (batch_size, d0, d1), then we create a kernel with shape (d1, units), and the kernel operates along axis

2 of the input, on every sub-tensor of shape (1, 1, d1) (there are batch_size * d0 such sub-tensors). The output in this case will have shape (batch_size, d0, units).

Besides, layer attributes cannot be modified after the layer has been called once (except for the trainable attribute). When a popular Kwarg input_shape is passed, then Keras will create an input layer to insert before the current layer. This can be treated as equivalent to explicitly defining an Input Layer.

G. Input shape

N-D tensor with shape: (batch_size, ..., input_dim). The most common situation would be a 2D input with shape (batch_size, input_dim).

H. Output shape

N-D tensor with shape: (batch size, ..., units). For instance, for a 2D input with shape (batch size, input_dim), the output would have shape (batch size, units).

I. Dropout layer

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/ (1 - rate) such that the sum over all inputs is unchanged.

Note that the Dropout layer only applies when training is set to True such tha lues are dropped during inference. When odel. Fit, training will be appro ue automatically, and in other co he Kwarg explicitly to True when ca .

Dropout layer. trainable does not affect the layer's behavior, as Dropout does not have any variables/weights that can be frozen during training.)

J. Image Data Generator:

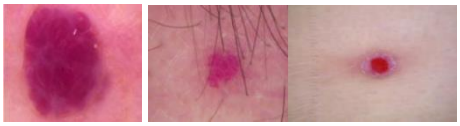
Rescales the image, applies shear in some range, zooms the image, and does horizontal flipping with the image. This Image Data Generator includes all possible orientations of the image.

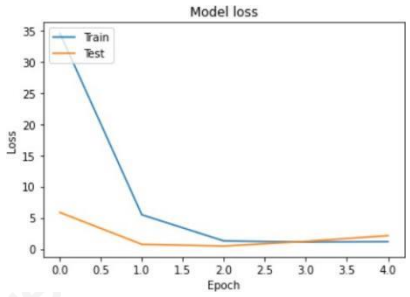
K. Training Process:

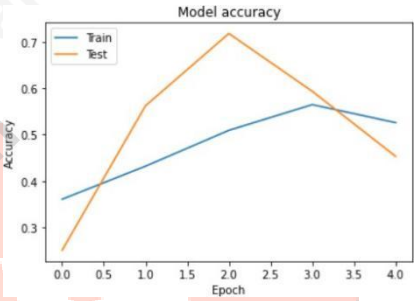
Train_datagen. flow_from_directory is the function that is used to prepare data from the train dataset directory Target size specifies the target size of the image. Test_datagen. flow_from_directory is used to prepare test data for the model and all is similar as above. The fit generator is used to fit the data into the model made above, other factors used are steps_per_epochs tell us about the number of times the model will execute for the training data.

L. Validation process:

Validation data is used to feed the validation/test data into the model. Validation steps denote the number of validation/test samples.







V. Features

Keras leverages various optimization techniques to make high-level neural network API easier and more performant. It supports the following features

Consistent, simple, and extensible API.Minimal structure - easy to achieve the result without any frills. It supports multiple platforms and backends. It is a user-friendly framework that runs on both CPU and GPU. Highly scalability of computation.

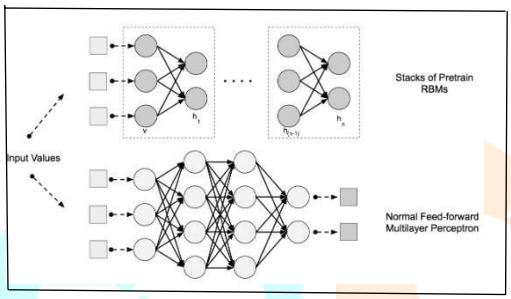
VI. Artificial Neural Network:

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets that look like the figure below. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

Artificial Neural Network (ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.

A. Key Points related to the architecture:

The network architecture has an input layer, a hidden layer (there can be more than 1), and an output layer. It is0 also called MLP (Multi-Layer Perceptron) because of the multiple layers. The hidden layer can be seen as a "distillation layer" that distills some of the important patterns from the inputs and passes them onto the next layer to see. It makes the network faster and more efficient by identifying only the important information from the inputs leaving out the redundant information The activation function serves two notable purposes: It captures non-linear relationships between the inputs. It helps convert the input into a more useful output.



B. Key advantages of Neural Networks:

ANNs have some key adva[illegible]t make them most suitable for certa[illegible]and situations:

1. ANNs have the abil1ity to lea[illegible] non-linear and complex relationships, wh[illegible]mportant because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.

2. ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

3. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e., data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g., stock prices) where data volatility is very high.

VII. Conclusion

Cancer is an abnormal growth of cells. By using a deep learning algorithm, we can be sure that the application which is going to be used in the coming future will be helpful for the betterment of human life. Where the disease can be known at an early stage or beforehand what type of disease the patient is affected with and what needs to be done before it's too late for the person. The survival rate will be more if the Melanoma is detected in time. The accuracy of detecting disease is excellent. Since machine learning plays an important role in skin cancer detection it can be a helpful factor in the medical field.

VIII. Future Work

The medical department wants to automate the detection of Skin Cancer from the eligibility process (real-time).

To automate this process by showing the prediction results in a web application or desktop application. To optimize the work to implement in an Artificial Intelligence environment.

IX. Reference

[1]. Suganya R. 'An automated computer-aided diagnosisof skin lesions detection and classification for dermoscopy images.' (2016) International Conference on Recent Trends in Information Technology (ICRTIT).

[2]. B. Harangi. 'Skin lesion classification with ensembles of deep convolutional neural networks.' (2018) J. Biomed. Information.

[3]. Barata, C., & Marques, J. S. 'Deep Learning for Skin Cancer Diagnosis with Hierarchical Architectures.' (2019) IEEE 16th International Symposium on Biomedical Imaging (ISBI).

[4]. Hosny K. M, Kassem M. A., &Foaud M. M. 'Skin Cancer Classification using Deep Learning and Transfer Learning.' (2018) 9th Cairo International Biomedical Engineering Conference (CIBEC).

[5]. Nida, Nudrat&Irtaza, Aun&Javed, Ali &Yousaf, Muhammad Haroon& Mahmood, Muhammad. 'Melanoma lesion detection and segmentation using deep region-based convolutional neural network and fuzzy C-means clustering.' (2019) International Journal of Medical Informatics.

[6]. Maglogiannis and C. Doukas, Overview of advanced computer vision systems for skin lesions

characterization IEEE Trans. on Information Technology in Biomedicine, vol. 13, no. 5, pp. 721–733, 2009

[7]. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv preprint arXiv:1512.03385, 2015.

[8]. J. Kawahara, A. BenTaieb, and G. Hamarneh, "Deep features to classify skin lesions," IEEE International Symposium on Biomedical Imaging (IEEE ISBI), pp. 1397–1400.

[9]. C. Barata, M. Ruela, M. Francisco, T. Mendonc¸a, and J. S. Marques, "Two systems for the detection of melanomas in dermoscopy images using texture and color features," IEEE Systems Journal, vol. 8, no. 3, pp. 965–979, 2014.

[10]. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[11]. N. Ahmed, Chaya "Segmentation and Classification of Skin Cancer Images," International Journal of Advanced Research in Computer Science and Software Engineering Research vol. 4, no. 5, pp. 1349–1353, 2014.

[12]. S. Achakanalli and G. Sadashivappa, "Skin cancer detection and diagnosis using image processing and implementation using neural networks and ABCD parameters," International Journal of Electronics, Communication & Instrumentation Engineering Research and Development., vol. 4, no. 3, pp. 85–96, 2014.

[13]. J. Pauline, S. Abraham, and B. Janney, "Detection of skin cancer by image processing techniques," Journal of Chemical and Pharmaceutical Research, vol. 7, no. 2, pp. 148–153, 2015.

[14]. Korotkov, K & Garcia, R. (2012), "Computerized Analysis of Pigmented Skin Lesions:", A review Artificial Intelligence in Medicine, Vol. 56, pp. 69-90.

[15]. R. Sumithra, M. Suhail, and D. S. Guru, "Segmentation and Classification of Skin Lesions for Disease Diagnosis," Procedia - Procedia Comput. Sci., vol. 45, pp. 76–85, 2015.

[16]. Z. She, Y. Liu, and A. Damato, Combination of features from skin pattern and ABCD analysis for lesion classification Skin Research and Technology, vol. 13, pp. 25–33, 2007.

MUTHUKUMARASAMY.S is a Ph.D. scholar in the department of Computer Science and Engineering at Anna University. He received his M.E. degree in 2011 at Sri Krishna Engineering College and B.E. degree in 2005 at E.G.S. Pillay Engineering College. His area of interest includes Mobile computing, Wireless Sensor Networks, Cloud Computing and Information Security.

KOKILA.A is a B.E. scholar in the department of Computer Science and Engineering at S.A. Engineering College. Her area of interest includes CNN and Artificial Intelligence. Subject knowledge in cloud computing, Database Management System, and python

SRIJA.R is a B.E. scholar in the department of Computer Science and Engineering at S.A. Engineering College. Her area of interest includes Web Development and Artificial Intelligence. Subject knowledge in Cloud Computing, Database Management System, and python

INDHUPRABHA.C is a B.E. scholar in the department of Computer Science and Engineering at S.A. Engineering College. Her area of interest includes ANN and Artificial Intelligence. Subject knowledge in Cloud Computing, Database Management System, and python