CONVOLUTIONAL NEURAL NETWORK TO CLASSIFY MEDICAL MNIST DATASET

Abstract: Convolutional Neural Network (CNN) is a classic algorithm in Deep Learning that is used to analyze and classify images. It doesn't compromise with the accuracy and performs better than humans on many of the image classification problems. In this paper, I have implemented a CNN using Python and TensorFlow framework to classify radiological images from the Medical MNIST dataset. This dataset consists of 58954 images with 6 different classes. I managed to train, validate and test my neural network by achieving an accuracy of 99.52% on the test data.

1. Introduction

Over the past years, Neural Networks have shown exceptional results while classifying the image datasets. In this research paper, I am scrutinizing the Convolutional Neural Network (CNN) to determine the most optimal architecture for the classification of the Medical MNIST dataset. This dataset has 58954 images and 6 classes viz. AbdomenCT, BreastMRI, ChestCT, CRX, Hand and HeadCT. The dataset is loaded, analyzed and then preprocessed using the image preprocessing techniques such as rescaling, zooming in and out, sheer ranging images, etc. to make it suitable before feeding into the network. I will be exploring the number and type of layers required by the model to give the best accuracy. Also, I will experiment the number of neurons (nodes) which are necessary in each layer. This paper gives an insight into the basic architecture of Convolutional Neural Network.

In Section 2, the Methodology of neural network is provided. Followed by the Methodology is Section 3, which illustrates the process I have followed to load, preprocess, train, validate and test my dataset. Furthermore, Section 4 comes up with the results of my neural network. Section 5 is the Conclusion of the network and the last section is the References which I have referred throughout the process.

2. Methodology

Deep learning is a subset of machine learning that has algorithms that are structured and function similar to the human brain. It has become one of the hottest technologies that has seen a wild growth over the past few decades [8]. It shows better performance as more data is fed into it unlike other old algorithms that plateau after a certain point. Interestingly, it has also shown a high impact and accuracy on medical imaging field over the past few years, delivering exceptional results that are comparable and even better than expert doctors/humans; see Figure 1.

In Deep Neural Networks, Convolutional Neural Network (CNN) is one of the main categories to do image recognition and classification. It takes an image as input, process and classify it under the different classes (probability). A CNN consists of an input layer, some hidden layers and an output layer. The input to a layer is a tensor with a shape (no. of inputs) x (input height) x (input width) x (input channels). Once the image passes through a convolutional layer, it becomes abstracted to a feature map also known as activation map, with a shape (no. of inputs) x (feature map height) x (feature map width) x (feature map channels). Pooling layers are used after the convolutional layer to reduce the dimensionality of each map but retaining the important information/features.

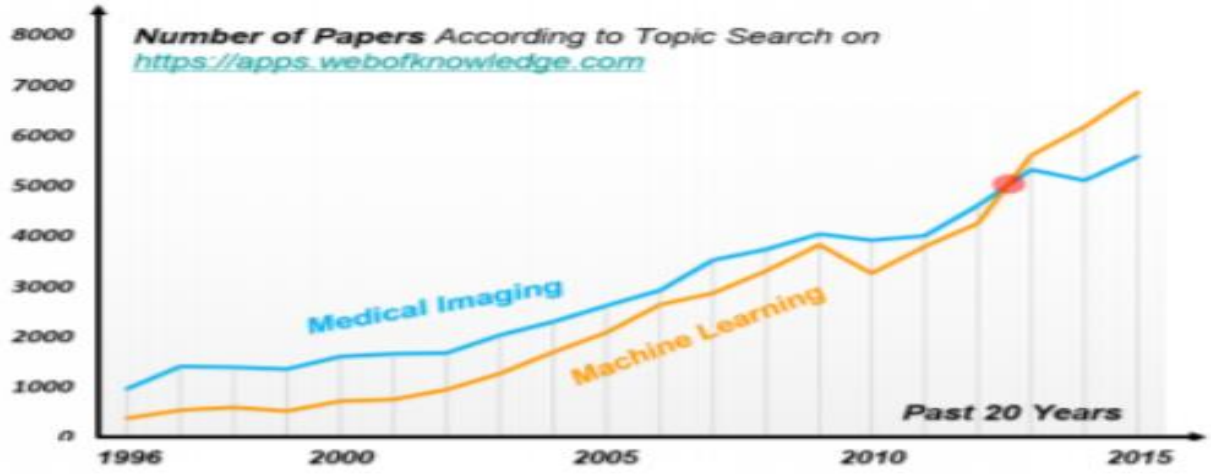


Figure 1. Attention to machine learning and medical imaging has exponentially increased over the past 2 decades. The intersection shows the research efforts became comparable in deep learning and medical imaging to that of machine learning.

There are 3 types of Pooling Layers viz. Max Pooling (takes the largest element from the feature map), Average Pooling (Avg. of the elements in a feature map window) and Sum Pooling (sum of all elements in the feature map). We can add more layers such as Dropout Layer, Dense Layer, Batch Normalization Layer, etc. depending upon the requirements and the accuracy to be achieved by the network. The final layer of CNN is a classification layer that has the number of nodes/neurons equal to the number of classes of the dataset. We can add as many hidden layers, until satisfied with the performance of our network. Based on the activation of the final convolutional layer, the output layer gives a probability that specifies how likely an image will belong to a class. The class which has the maximum value is more likely to be the result of the classification.

Shin [1] in his research paper, discusses the development and improvement that has been made in classifying the image datasets because of the deep convolutional neural network (CNNs). He discusses the various CNN architectures that have high number of parameters which vary in different layers of the deep neural networks. I got some idea of how to use a combination of layers from this paper. The paper [3], dives into the basic

layers which make a CNN. They are convolutional layers, activation layers, regularization layers, pooling layers, dropout layers and batch normalization layers. The overall architecture of a basic CNN is highly dependent on various factors such as number of nodes, type of activation function, size of the kernel, regularization, learning rate of the network, etc. [4]. The deep learning is a great algorithm for feature extraction from images as the papers [5]-[7] suggests. In 2016, [2] issued his paper to demonstrate the perspective of deep learning on medical imaging and how it empowers image analysis. The results were impressive and showed high accuracy. Further to the research in the medical images, I am going to explore the basic architecture.

3. Simulations

3.1 Dataset

This research paper is based on the classification of Medical MNIST dataset. It was taken from the Kaggle. Link to it is https://www.kaggle.com/andrewmvd/medical-mnist. This dataset has 58954 medical images from 6 different classes. The images are a combination of Computed Tomography (CT), X-ray and MRI of various body parts of the human beings. The 6 classes of this dataset are AbdomenCT (Computed Tomography of the Abdominal Cavity), BreastMRI (MRI of the Breast), ChestCT (Computed Tomography of the Chest), HeadCT (Computed Tomography of the Head), Hand (X-ray of the Hand) and CRX (X-ray of the Chest). In this dataset, every class has 10000 images except BreastMRI which has 8954 images. The images are black and white of size 64x64; see Figure 2.

3.2 Processing Data

In order to implement and analyze the data, the first step I did was to download the dataset from the Kaggle. The size of it is 76MB which is not very large. In order to create testing data. I manually created 6 sub directories into a folder with the same name as that of each class. This testing data comprises of approx. 20% images of the dataset and each class in it has almost same number of images. This dataset is quite interesting as some of the classes such as HeadCT and AbdomenCT are quite similar to each other which makes the classification harder. In order to extract the data, I have used the Image Data Generator from the Keras Api of the Tensorflow framework. I have split my training dataset into 20% validation and rest 80% for training before feeding into the model. Then I rescaled the training and validation data images, zoomed 20% in and 120% out i.e., zoom range equals 0.2 and sheered the images by 20% i.e., sheer range equals 0.2 to make the dataset better. The important part was to shuffle the training data before passing into the neural network.

In the case of testing data, I have just rescaled the images. This is because the images which will be fed to the network in the future will not be processed before passing into the model. This way we can determine the true accuracy of our model on unseen raw data.

3.3 Network Architecture

I used the Sequential Model to implement my Convolutional Neural Network (CNN). The Input Layer of my CNN is a Conv2D layer with 32 neurons. It has a kernel size of 2. The kernel size is used to specify the height and width of the 2D convolutional window. The strides of the convolutional have the same value 2 along both height and width. The padding used was "same" which means evenly padding on the up/down and right/left of the input such that output has the same height/width dimension as the input. This input layer is complimented with a MaxPooling2D layer which reduces the dimensionality of the input image. The activation function used is Rectified Linear Unit also known as ReLU. It is the most commonly used activation function in neural networks especially in CNNs.

The Hidden Layers comprise of two pairs of Conv2D Layer, MaxPooling2D Layer, Flatten and Dense Layer. I increased the number of neurons to 64 in the Conv2D Layers which is double as compared to the input layer. Then, I have used the Flatten layer to flatten the input. It removes all the dimensions of the input tensor except for one. The shape of the tensor after passing through the Flatten Layer is equal to the number of elements present in the tensor. The shape is 1D. Then, in order to prevent the Convolutional Neural Network from overfitting, I have used a Dropout Layer with a rate of 0.25. It randomly rejects 25% of the features of the training data so that the neural network doesn't learns too much and as a result doesn't overfits. After that the output is passed through two Dense Layers comprising of 128 neurons each with activation function "relu".

Finally, the output of the Dense Layers is put across the Output Layer which is also a Dense Layer. It has the same number of neurons in it as that of the classes i.e., 6. The activation of the Output Layer is 'Softmax'. The model is then compiled using the "adam" optimizer. The probabilistic loss is set to categorical entropy because we have more than 2 label classes in our neural network. Then, the model is finally fit with epochs value set to 3. After training the model, training and validation accuracy were 99.64% and 99.61% respectively. This is a great output accuracy of a deep learning model. In order to evaluate my model, I passed the testing dataset to the trained model. This gave me an accuracy of 99.59%. The model can be seen in the Figure 6.

4. Results

The convolutional neural network managed to achieve an accuracy of 99.52% on the testing data.

4.1 Accuracy Curve

The accuracy curve shows the accuracy at each epoch of the training and the validation dataset; see Figure 3.

4.2 Loss Curve

The loss curve is the loss of training and the validation dataset at each epoch; see Figure 4.

4.3 Confusion Matrix

A confusion matrix, also known as error matrix, is a table that shows the visualization of the performance of an algorithm. The confusion matrix of my model can be seen in Figure 5. It shows that 2 images belonged to class 2 i.e ChestCT but were classified as Hand. 31 images belonged to actual class 3

but were misclassified as class 0. 4 images belonged to class 5 but were classified as class 4 and 5 images were actually from class 4 but misclassified as class 2. Rest images were correctly classified.

5. Critical Analysis

I got this high accuracy of ~99% by exploring and integrating many different layers and their parameters in my convolutional neural network. I started with the full dataset to check if it'll be compatible to load and test this big dataset. As an outcome, it was pretty fast and I was able to load and train the model. In order to make it better for the model I dd some pre-processing like sheer-range, rescaling, zoom range so that the model can learn the images from different angles/views. While exploring the layers of the convolutional neural network, I selected Conv2D layer as it gives tensor of output. Secondly, I used MaxPooling2D Layer which helped in reducing the feature dimensionality of these complex images by taking the maximum value over the window. The window size I chose was 2. The most important Layer was Dropout Layer. The training accuracy can be achieved by adding more layers with a combination of neurons and activation functions, but how well the model performs on unseen/new data is the ultimate goal of any deep learning model. This accuracy determines how good the model is. In order to prevent the model from learning each feature (overfitting) I have used the Dropout layer. It played an important part to validate my model on the validation dataset and to overcome the problem of overfitting. I even explored activation functions such as Softmax, ReLU, Sigmoid but Softmax gave the best result. Overall, analyzing and experimenting different things, the model gave exceptional accuracy on the testing data. If the number of classes and images increase for this dataset in the future then maybe some modifications will be required but as of now it works well on the present dataset.

6. Conclusion

Deep Learning is a great technique for classifying image datasets. I have used Tensorflow and Keras which is well documented and can be easily understood to implement my neural network. I got exceptional accuracy of 99.58% on the testing data which states that my convolutional network was trained and validated well. The CNN was able to classify such complex images easily. This architecture can be enhanced more if the number of classes increase from 6 to a big number. For this number of classes, it is the best I could get.

7. References

[1] H.-C. Shin et al., ''Deep convolutional neural networks for computer aided detection: CNN architectures, dataset characteristics and transfer learning,'' IEEE Trans. Med. Imag., vol. 35, no. 5, pp. 1285–1298, May 2016.

[2] Wang, G.: A perspective on deep imaging. IEEE Access 4, 8914–8924 (2016)

[3] Lundervold, A.S., Lundervold, A.: An overview of deep learning in medical

imaging focusing on MRI. Z. Phys. 29(2), 102-127 (2018)

[4] Sahiner, B., Pezeshk, A., Hadjiiski, L.M., Wang, X., Drukker, K., Cha, K.H., Summers, R.M., Giger, M.L.: Deep learning in medical imaging and radiation therapy. Med. Phys. 46(1), e1–e36 (2019)

[5] Y. LeCun, Y. Bengio, and G. Hinton, ‘‘Deep learning,’’ Nature, vol. 521, pp. 44–436, May 2015.

[6] (Sep. 1, 2016). [Online]. Available: http://people.idsia.ch/~juergen/deeplearning-conspiracy.html

[7] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016

[8] R. D. Hof, 10 Breakthrough Technologies—Deep Learning. Cambridge, MA, USA: MIT Technology Review, 2013

[9] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, ‘‘A learning algorithm for Boltzmann machines,’’ Cognit. Sci., vol. 9, no. 1, pp. 147–169, 1985.

[10] P. Smolensky, ‘‘Information processing in dynamical systems: Foundations of harmony theory,’’ in Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations, vol. 1, D. E. Rumelhart, Ed. Cambridge, MA, USA: MIT Press, 1986, pp. 194–281.

[11] G. van Tulder and M. de Bruijne, ‘‘Combining generative and discriminative representation learning for lung CT analysis with convolutional restricted Boltzmann machines,’’ IEEE Trans. Med. Imag., vol. 35, no. 5, pp. 1262–1272, May 2016.

[12] M. A. Carreira-Perpinan and G. E. Hinton, ‘‘On contrastive divergence learning,’’ in Proc. Aistats, vol. 10. 2005, pp. 33–40.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, ‘‘A fast learning algorithm for deep belief nets,’’ Neural Comput., vol. 18, no. 7, pp. 1527–1554, 2006.

Appendix

1. Images of each class of the dataset

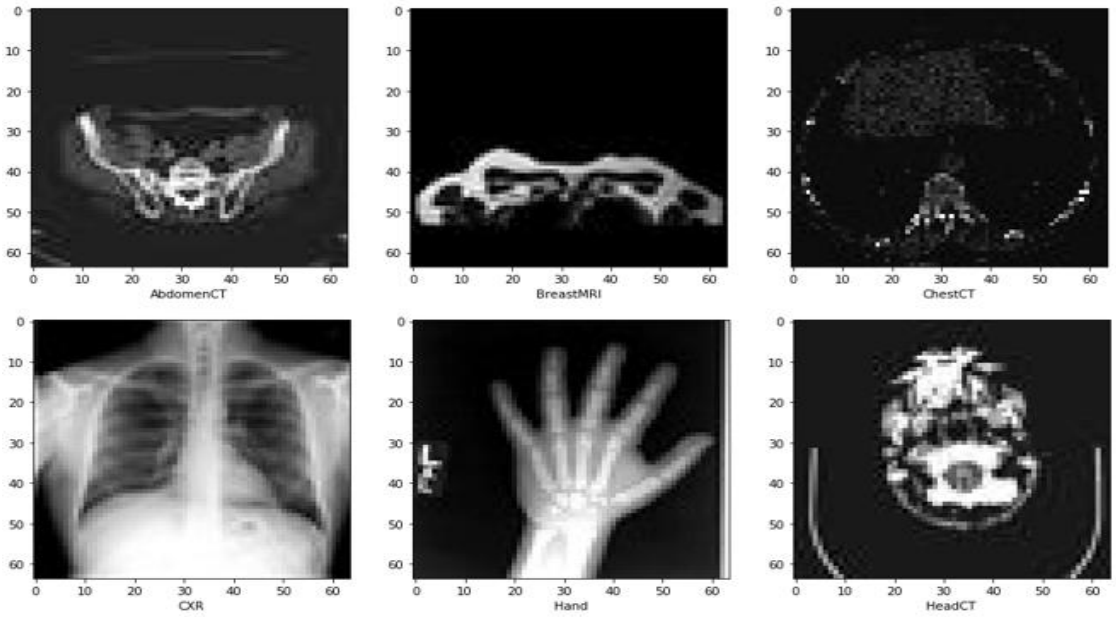
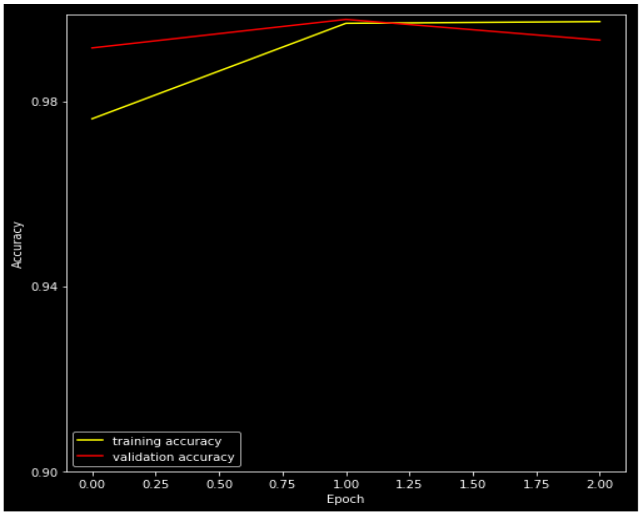


Figure 2. Images of 6 different classes in the Medical MNIST dataset.

2. Accuracy curve of the CNN model




```
#test data
evalu = model.evaluate_generator(test_data_generator)

print("Accuracy : "+str(evalu[1]*100))
```

```
Accuracy : 99.58677887916565
```

Figure 3. Shows the accuracy curve of the model on training and validation dataset and the accuracy of the trained model as evaluated on the testing data.

3. Loss Curve of the Model

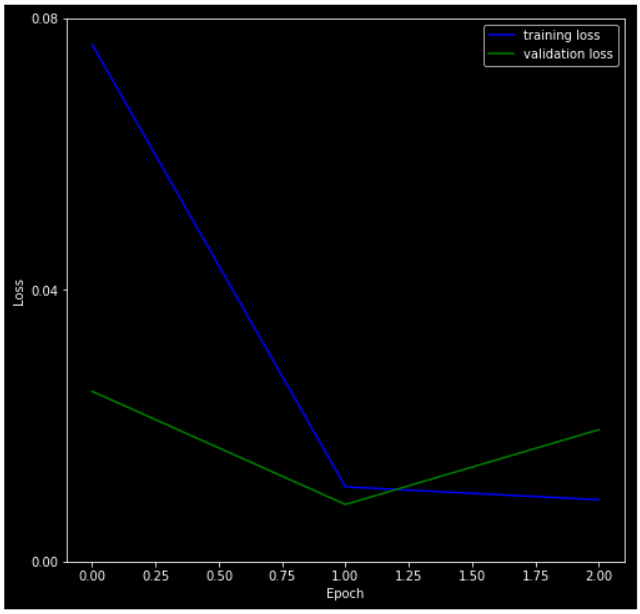


Figure 4. Shows the loss of the model on training and validation dataset.

4. Confusion Matrix

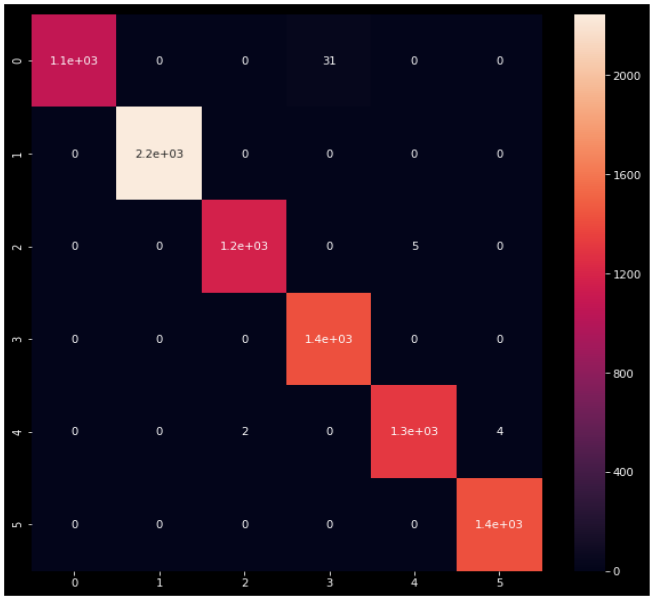


Figure 5. Shows the confusion matrix