Module No: CSY3025

Module Name: Artificial Intelligence Techniques

Assignment 1- Image Classification Using Deep Learning

Link to dataset: <https://drive.google.com/drive/folders/1puSSa5Um5VP-xX-A6wy-oA9hu71HDVSh?usp=sharing>

Link to .ipynb file: <https://colab.research.google.com/drive/11bAtwvkgqcCcs-KoX9m0hTFLJ1jmP2sU?usp=sharing>

Name: Bipashu Hamal Thakuri

Level 6, BSc. Computing

University Of Northampton, 2022

Contents

1. INTRODUCTION
2. PROBLEM ANALYSIS AND BACKGROUND RESEARCH
   1. Problem Space and Requirements Analysis
   2. Literature Review
      1. Deep Learning Python Frameworks
      2. Sequential Model
      3. Image Classification with Keras
      4. Facial Detection with Deep Learning
      5. Sentiment Analysis with Deep Learning
3. BUILDING DEEP LEARNING NETWORK
   1. Dataset
   2. Network Architecture
   3. Training and Validation
   4. Inference
4. SUMMARY OF ADDITIONAL FEATURES
   1. Additional Classes
   2. Data Visualisation
5. DISCUSSIONS AND CONCLUSIONS
   1. Lessons Learned
   2. Limitations
6. REFERENCES

Figures:

Fig 1: Deep Learning Sequential Neural Network

Fig 2: Facial Detection

Fig 3: Google Colab

Fig4: Datasets (Main directories)

Fig 5: Datasets (Sub directories)

Fig 6: Data pre-processing

Fig 7: Model Structure

Fig 8: Output of training Model

Fig 9: Testing algorithm

Fig 10: Rounded test results.

Fig 11: Data Visualization function

Fig 12: Testing Images Visualization

Fig 13: Confusion matrix function

Fig 14: Confusion Matrix

1. INTRODUCTION

Machine learning (ML) systems are increasingly making decisions that affect people's daily lives and society. For example, machine learning (ML) and artificial intelligence (AI) are already being used to control end-to-end autonomous vehicles, determine how long a criminal should be imprisoned, order in which news is presented to a person, and even diagnose and treat medical patients.

Deep learning system development and deployment is critical in machine learning to avoid unintended side effects and ensure long-term acceptance of algorithms used. Even the seemingly simple task of facial recognition has been shown to attract a great deal of user and industry attention, as it enables related industries to communicate effectively with end users. Deep learning algorithms can be used to create all of these chores. Car manufacturers, for example, can use a sentiment analysis tool, which is part of a facial recognition system, to detect whether a driver is stressed, angry, or tired. Emotion recognition can be used by teachers to assess student engagement. Future computer games may also be designed to adapt game content based on player emotions to improve the gaming experience.

We were given the task of developing an image classifier using “Deep Learning” that analyses different emotions from a closed-up image of a person due to the possible scope described above. The task can be divided into following objectives:

* Establish necessary dataset after proper research on the problem domain.
* Include as many variants as possible for the datasets. Make it inclusive, ethnic wise.
* Build deep learning model for sentiment detection for the closed-up images.
* Make the model as efficient as possible, verifying through testing and evaluation with data visualization.
* Implement user application and comparative solution, if possible.

In this paper, I demonstrate each step that leads us to the final deployment: from problem space research and analysis to architectural design of application, including decisions made at each step of model development and other design diagrams.

1. PROBLEM ANALYSIS AND BACKGROUND RESEARCH

This section will investigate the area surrounding given problem and requirements for it, following the literature review of related sub-topics.

* 1. Problem Space and Requirement Analysis

We are given a task to develop a sentiment detection model. Sentiments come in a variety of forms. All of these can be identified by human facial expressions. We ourselves understand each other through expressions. The rise of automation has led many scientists to train machines on how to understand user emotions, also for the better communication and transparency. Achieving this with computers will improve the user experience and will also result in better experience for humans dealing with machines. For example, many companies can use the sentiment analysis algorithm to deal with customers understanding and make them feel better by improving the company performance hand in hand with detecting the changing emotions. It can be powerful tool for marketing campaigns, since sometimes it can be hard for humans to identify emotions. However, implementing sentiment analysis has many challenges. With machine learning, we can make it easier. Deep learning is a subset of machine learning. It is one of the pillars of AI advancement. It builds algorithms on neural network layers so that it can learn on its own if trained with labelled data. As a result, it uses its own knowledge to draw conclusions from untrained data. Hence, deep learning models can be helpful for our task to develop sentiment analysis model.

The solution synopsis calls for the development of an image classifier to detect emotions based on closed-up facial images. This can be considered a supervised learning problem because we can train this model by combining inputs and corresponding targets. Inputs can be a collection of images from various categories. This is known as model training, and it is the first step after the inputs have been prepared. In terms of emotion detection, the trained model must be able to detect more than 5 emotions to perform well. The use of deep learning models will train the model itself if appropriate inputs are provided. The performance capability of the model can be tested through testing. It should deport reasonable performance. There are various types of deep learning models complimentary to this task, which we will review later. In addition to all of this, the solution requires proper data visualization as additional functionality to keep track of data patterns and model functioning. To complete our task, we can use Python libraries such as Scikit-Learn, NumPy, TensorFlow, and Keras.

The section below will provide more detailed documentation on related topics.

* 1. Literature Review
     1. Deep Learning Python Frameworks

Deep Learning uses neural network for analysis and calculation. There are multiple layers made up of group of neurons where each layer is input to the another transforming information through connecting channels. The first layer is known as the input layer where all the inputs are fed, and the last layer is the output layer where each neuron represents final value. Between those two layers, there are multiple hidden layers where information is transformed [1]. We now understand that neural networks are essential for any Deep Learning-based project.

We will have to use Python libraries to create the neural network whether we are coding from scratch or directly implementing a neural network library. Some of the most popular Python libraries for creating neural networks are scikit-learn, NumPy, Keras, and TensorFlow. Scikit-Learn can be used to create one from the ground up. However, because we are already familiar with the fundamentals of neural network operation, we will use Keras and TensorFlow for this task. Keras and TensorFlow are two of the most efficient frameworks for building neural networks for deep learning. Keras is fully integrated with TensorFlow. Nonetheless, the general idea is to start by importing the Python libraries. Then, in our case, pre-process the data (input), which are image pixels. Add weight and bias to the inputs, which are done automatically by deep learning neural networks after we define and compile the model for one. Then, using those data, train the network. Finally, run the model with new data to see how effective it is (ActiveState, 2021).

While we are developing a deep learning model with Keras/TensorFlow, we will first integrate Python and Colab Notebook into our system. Colab Notebook is a cloud platform where we can run our code, and it is integrated with Google Drive, making it simple for us to collaborate on data in the system. The required Python libraries will then be imported and used to begin developing the model.

* + 1. Sequential Model

When it comes to developing models with Keras/TensorFlow, we have several options. There are two kinds of models: sequential models and functional models. In contrast to Functional Model, which is more flexible for creating models with multiple inputs and outputs, Sequential Model is best for creating simple models with only one input and output. For this task, we will use a sequential model because we will only have one time input and one time output, which means we will provide an image and the model will spit out the result (sentiment).

The sequential model is a layer-by-layer stack.

Diagram

Description automatically generated

Fig 1: Deep Learning Sequential Neural Network

To start building neural network sequentially, we first need to import Sequential Model from Keras library. We can create a Sequential model by passing a list of layer instances to the constructor; these layers are provided by the Keras library once again. We can declare a sequential model by specifying a list of layers, which are the Sequential Model's building blocks. Dense, Activation, and Conv2D are some of the layers. Each layer will be specified with number of dimensional vectors as well as an activation function. The first layer will have an additional input shape attribute. With the summary function, we can see the model structure. It will display the number of parameters as well as the output shape for each layer. After the model has been defined, it must be compiled. The compilation phenomenon requires three components: an optimizer, a loss function, and metrics. All of this contributes to the simplified operation of neural networks. Following compilation, we must train our model with procedure data and labels. The data must be in the format that the model supports. We can check model accuracy throughout the training process for each epoch. Then, we can predict the model accuracy with test data. The model and the parts of the model can be saved in different formats with the use of keras.

This development and working knowledge of sequential model with keras library will aid in developing a model for our project that is adequate for the sequential model as discussed previously.

* + 1. Image Classification with Keras

The categorization of images is known as image classification. In deep learning, we classify images using rules that are built on top of model (neural network) that are created using Python libraries such as Keras. Image classification is dominated by neural networks, which extract features from images while training and classify them based on these features.

To begin Image Classification from scratch, we must first import the Keras/TensorFlow libraries. The raw data (images) can then be downloaded from anywhere. For this task, our data can be images for various sentiment classes. We can either download manually or use the python curl command. Kaggle is the most popular website for downloading datasets. After the datasets have been collected, we can use one of the keras library models to prepare datasets to be loaded into neural network for training purpose. We can normalize the image data with rescaling. Then we can visualize our trained data using matplotlib library. Then, we build a model with layers specification. Conv2D is a two-dimensional layer and is very common layer used in image classification models. The input shape attribute in the first hidden layer defines the batch size, height, width, and channels of the input. We can now train model by specifying epochs (the number of times we want our model to step through a single batch) and training steps (individual batch of images). At last, we can run inference on new data to evaluate model efficiency. We can do this by collecting images for different emotions and test them through model for our task.

* + 1. Facial Detection with Deep Learning

Humans can easily detect faces in photographs, but computers have historically struggled due to the dynamic nature of faces. Faces, for example, must be detected regardless of orientation or angle, light levels, clothing, accessories, hair colour, facial hair, makeup, age, and so on (Brownlee, n.d.).

Neural networks in deep learning take the face data, train themselves to recognize the patterns and predict the output for new set of similar face data. Facebook's tagging feature is one example of facial detection using Deep Learning. If we are about to upload a photo of multiple people, Facebook recognizes the faces with its deep learning algorithm and suggests names for tagging. This mechanism is iterative in the sense that machines become smarter with each dataset analyzation.

There are several approaches for facial detection. One of them is Neural Network Based Facial Detection. It employs a dual-layer system based on convolutional neural networks, with the first network determining the approximate position of faces and the second network analysing the detected face to provide more accurate localization (Mehta, et al., 2018). One of the main fundamentals of Deep Learning is Neural Network.

A group of people smiling

Description automatically generated with medium confidence

‌

Fig 2: Facial Detection

* + 1. Sentiment Analysis with Deep Learning

One of the applications of facial detection with deep learning is sentiment analysis. Sentiment can be analysed through text data and image data. Although, a lot of research work has been done for sentiment analysis of textual data; there has been limited work that focuses on analysing the sentiment of image data. Deep learning techniques are used for sentiment analysis to overcome this challenge, as deep learning models are capable of effectively learning image behaviour or polarity. Image recognition, image prediction, image sentiment analysis, and image classification are some of the fields where Neural Network (NN) has performed well, implying that deep learning has significant performance in image sentiment analysis. (Mittal, et al., 2018).

With the use of deep learning models in sentiment analysis, effective results are foreseen. Hence, the use of deep learning to the given task of sentiment analysis using image classifier is significant.

1. BUILDING DEEP LEARNING NETWORK

This section will demonstrate the flow of model development from identifying and collecting datasets to testing model performance, including the implementation of some additional features.

* 1. Dataset

Before preparing dataset for the further processes, we first decide to assert our system file to the google drive by using Colab notebook, which is done by:

Graphical user interface, text, application

Description automatically generated

Fig 3: Google Colab

I chose to obtain raw dataset from the Kaggle website. The raw data is made up of images divided into two categories: train and test. Both directories are further subdivided into seven subdirectories each. Each of the seven subdirectories represents a different emotion. They are angry, disgust, fear, happy, neutral, sad, and surprise. All seven subdirectories in both contain a collection of relevant grayscale images with a resolution of 48\*48 pixels. Separate datasets are required for training, testing, and validation. So, I decided to use the data in the test directory for validation, train for training, and create another directory with the same seven sub-directories for testing, where I randomly collected images for each sub-directory. The collected dataset was again asserted inside drive, and it looked like as:

Graphical user interface, text, application, website

Description automatically generated

Fig 4: Datasets (Main directories)

The subdirectories within serve as classes for image classification. As a result, our model is a multi-class classifier.

Graphical user interface

Description automatically generated with medium confidence

Fig 5: Datasets (Sub directories)

Following data collection and organization, it was time to process the data to process it before passing it to the model.

I first decided to create variables to assign train and validation dataset path. Because we know that the data model will receive during training must be in the format that the model accepts, I decided to use the ImageDataGenerator class from the Keras image processing library to generate batches of augmented images. While the model is being trained, it will apply random transformations to our images to make the process easier. As a result, a set of images for model inputs was created for training and validation purpose.

Graphical user interface, text

Description automatically generated

Fig 6: Data pre-processing

The image generator will go through all the images in that directory's subdirectories, fetching each image and naming it with the corresponding class/folder name, before resizing each image to the target size of 48\*48 and separating each batch of 64 images. For each image, the colour mode will be grayscale, and the class mode will be categorical, based on the names of folders. The batch for training and validation is now complete. The rescaling described above will convert every pixel value to 0,1 resulting in identical images with equal resolution.

* 1. Network Architecture

With four sequential convolutional layers, we build a convolutional neural network. Convolutional layers are widely used in image classification because they capture image features. Each Convolutional layer has ReLU activation and kernel size is (3,3) which is common size for image data and has 32 neurons for the first one and 64 neurons for the rest layers. The input shape is identified for the first hidden layer with 48 height, 48 width and the number of colour channel is 1 representing the grayscale format images. The second, third, and fourth convolutional layers are followed by the max pooling layer, which reduces the image dimension by half. After each max pooling layer, the dropout layers will remove 25% of the neurons in each epoch and train on the remaining 75%. This will aid in reducing overfitting. The flatten layer is used to convert a multidimensional image from two dimensions into a single dimension. The first dense layer has 1024 neurons, and the last one, the output layer, has 7 neurons corresponding to 7 classes with the activation function SoftMax, which can predict multinomial probability in multiclass classification problems.

Table

Description automatically generated

Fig 7: Model Structure

After building the model, it is compiled for training by specifying the optimizer and loss function. Adam is the optimizer and has a learning rate of 0.0001 and accuracy metrics. Loss function is categorical cross entropy, used frequently in multi class classification.

* 1. Training and Validation

The model will now be set to train using the fit function after it has been compiled. We then passed data generated by image generator previously for training and validating set to the fit function. To specify number of sample batches from our training set that should be passed to the model before declaring one epoch complete, we must use steps per epoch. Because our training set contains 28709 images and our batch size is 50, we set steps per epoch to 574, because 574 batches of 50 samples each will cover our entire training set. To specify this value, I used len function for each set. The number of epochs is set to 50. This means it will complete 574 steps 50 times for each batch. Similar is the case for validating data.

Table

Description automatically generated

Table

Description automatically generated

Fig 8: Output of training Model

We can conclude that the performance is reasonable. However, it can be better. The accuracy on training set reached 85% whereas on validating set 60%.

* 1. Inference

The last step after training is inference where we test our model with new set of data and check accuracy. First, we processed the testing data like training and validating data before. I first decided to create variable to assign test dataset path. I then decided to use the ImageDataGenerator class from the Keras image processing library to generate batches of augmented images. While the model is being tested, it will apply random transformations to our images to make the process easier. As a result, a set of images for model inputs was created for testing purpose.

We perform this testing by using predict function. We can check the accuracy later by visualizing the data as a part of additional functionalities.

Graphical user interface, text, application, chat or text message

Description automatically generated

Fig 9: Testing algorithm

Table

Description automatically generated with medium confidence

Fig 10: Rounded test results.

We can manually check the result with corresponding images to identify accuracy. But we have another method called confusion matrix which I will discuss later.

1. SUMMARY OF ADDITIONAL FEATURES
   1. Additional Classes

The minimum number of classes required was five. However, I used 7 classes to fulfil an additional requirement. There are seven classes: angry, happy, fear, disgust, sad, surprise, and neutral for all three training, validating, and testing purpose. Each class contain set of images labelled with corresponding classes during processing.

* 1. Data Visualisation

The graphic representation of images is known as data visualization. As an added feature, I've used data visualization techniques here and there. I started by using it to visualize a training set of data by importing the mat plot lib library. To view the training set of images, I used a predefined matplot lib function with 5 rows and 10 columns.

Text

Description automatically generated

Fig 11: Data Visualization function

Next, I used the same function for visualizing testing set of images

A collage of a person

Description automatically generated with medium confidence

Fig 12: Testing Images Visualization

To improve visualization, the prediction results were visualized using a confusion matrix. Following the testing, a confusion matrix was generated by passing the true labels and predicted labels of the tested samples. It compares truly predicted image labels and falsely predicted image labels.Then, I decided to use predefined function for plotting confusion matrix.

Text

Description automatically generated

Chart

Description automatically generated with low confidence

Fig 13: Confusion matrix function

Chart, scatter chart

Description automatically generated

Fig 14: Confusion Matrix

We can see that the model accurately predicted 1 angry face, 5 disgust faces, 17 fear faces, 20 happy faces, 19 neutral faces, 19 sad faces, and 20 surprise faces and incorrectly predicted other 28 images. So, the performance shall be evaluated as reasonable; although, it could have been better. At last, I saved the model as JSON file.

1. DISCUSSIONS AND CONCLUSIONS

In this paper, we documented the process of developing a sentiment analysis image classifier, from problem space research to identifying requirements, analysing them, and feeding the results into model design and implementation.

* 1. Lessons Learned
* Deep Learning employs neural network layers to construct algorithms that can learn on their own after being trained with labelled data.
* Python has various open-source libraries to create neural network for deep learning.
* Sequential model is a stack of layers of neurons which is used in simple models with only one input and output.
* Conv2D is a two-dimensional layer and is very common layer used in image classification models with Keras.
* The fundamental of data analysis is to first understand the data and collect reasonable data only.
* Data Visualisation is an effective technique in analysing model performance. One of the core techniques is plotting confusion matrix.
  1. Limitations
* The model's performance was not stellar. This could be the result of algorithm bias. For example, the number of images in each class was vastly different. I could have properly distributed the images first, or I could have chosen an equal number of images from each category.
* Some of the additional features, such as user applications and “in the wild” testing, were not implemented. This could be due to insufficient research for specific purposes.
* The training of model took 5-6 hrs. The use of GPU could have solved the issue.

1. REFERENCES
2. www.youtube.com. (n.d.). Deep Learning In 5 Minutes | What Is Deep Learning? | Deep Learning Explained Simply | Simplilearn. [online] Available at: https://youtu.be/6M5VXKLf4D4 [Accessed 23 Mar. 2022].
3. ActiveState. (n.d.). How To Create a Neural Network In Python – With And Without Keras. [online] Available at: https://www.activestate.com/resources/quick-reads/how-to-create-a-neural-network-in-python-with-and-without-keras/ [Accessed 23 Mar. 2022].
4. www.youtube.com. (n.d.). Sequential Model - Keras. [online] Available at: https://youtu.be/VGCHcgmZu24 [Accessed 23 Mar. 2022].
5. www.youtube.com. (n.d.). Keras Image Classification Tutorial | Image Classification Using Deep Learning | Simplilearn. [online] Available at: https://youtu.be/LKMi8Daf2ts [Accessed 23 Mar. 2022].
6. https://www.facebook.com/jason.brownlee.39 (2019). How to Perform Face Detection with Deep Learning. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/>.
7. www.youtube.com. (n.d.). How face Recognition Work [Neural Network]. [online] Available at: https://youtu.be/xkDKroAlwcw [Accessed 23 Mar. 2022].
8. ‌ J. Mehta, E. Ramnani and S. Singh, "Face Detection and Tagging Using Deep Learning," 2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP), 2018, pp. 1-6, doi: 10.1109/ICCCSP.2018.8452853.
9. ‌ N. Mittal, D. Sharma and M. L. Joshi, "Image Sentiment Analysis Using Deep Learning," 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2018, pp. 684-687, doi: 10.1109/WI.2018.00-11.
10. www.kaggle.com. (n.d.). fer2013. [online] Available at: https://www.kaggle.com/datasets/rkuo2000/fer2013 [Accessed 23 Mar. 2022].