Feature Extraction by Rework Image Recognition (RIR) Learning Model

P. Patchaiammal and G. Sundar

Abstract Image recognition is a widely adopted feature extraction technique in the field of deep learning. Convolutional neural network is one of the algorithms of deep learning used in the field of image analysis. Most of the software under development are prone to fault because of code moderation, so rework-related feature of software development needs to maintain fault-related dataset for future prediction. Rework is the major problem related to time and cost estimation after the software delivery. This paper is used to identify the rework effort of Organization-A, which is to be used for finding rework effort estimation in future projects. In most of the organization, historical dataset can be available in either document format or image format. This paper proposed a new model called RIR (Rework Image Recognition) learning model, which is used to generate the CSV file as the result of extracting the feature from Rework report image of an Organization-A. The model is generated by using a deep learning algorithmic technique called convolutional neural network. This paper also shows the performance of the RIR (Rework Image Recognition) learning model in the base of accuracy and loss level, and the future use of the rework result is also presented.

Keywords Deep learning (DL) · Machine learning (ML) · CNN (convolutional neural networks) · Rework Image Recognition (RIR) learning model · Rework report

P. Patchaiammal (✉) · G. Sundar
Bharath University, Chennai, Tamil Nadu, India

A. Ramu et al. (eds.), *International Conference on Computing, Communication, Electrical and Biomedical Systems*, EAI/Springer Innovations in Communication and Computing, https://doi.org/10.1007/978-3-030-86165-0_18

1 Introduction

1.1 Image Recognition

Image is a visual representation of text or object. Image looks in the human eye is not considered as same when it is converted as data. The image is a set of segments, which are formed with pixels. For dataset transformation, the image should be in same pattern. Every image segment has pixel intensity. The digital colour image of RGB channel should be converted into greyscale BGR channel. The class of image recognition means feed the image into a neural network in order to receive various image labels like objects, people, places or text in the image.

1.2 Deep Learning (DL)

Deep learning is the machine learning technique with set of neural network algorithms. Deep learning is used to make desired behaviour on the given problem with the help of activation of network in the chain of computational stages. Problem-solving using learning method will deliver the accurate result. Learning from the historical data observation to enable the accurate solution of the problem is achieved by natural deep learning. Deep learning is best for big dataset. In this, the solution of the given problem is achieved by machines. DL is used to solve difficult pattern detection problems. In DL, a system is developed from the training examples to recognize the pattern.

A logical activity of the idea is obtained only from human's inherent activity. The activity is described as a mathematical model, which (for a neuron) takes input, processes those inputs and returns an output. ML algorithms are formed by learning system. This learning process is used to perform a particular task or functions. DL is a powerful algorithm based on machine learning techniques to solve realistic problems.

The human brain is the most powerful machine than computers. The process of brain thinking is obtained from the result of direct learning experience. This learning is used for making natural assumption function to implement a computer-based network of neurons.

To categorize new defect entry, there is a need for human-like detective inference for decision-making.

Feature learning is essential for representing the feature for computation. It involves reducing the input features in order to get meaningful results. Deep

learning is a technique used to focus on the features used in the solution. Deep learning creates an architecture known as multi-neural network architecture with three major types.

- ANN: The data present in the form of numbers will be solved using ANN.
- CNN: Transferred learning technique used to analyse the image form of input.
- RNN: Time series-based inputs are solved using RNN.

The previous research work shows the need of fault taxonomy to predict the fault in early software development life cycle [1]. Various research works are done to identify the proper prediction technique for finding early fault prediction [2–4]. But to identify the right place of fault, the necessary rework percentage is to be calculated. This research work helps achieve this by forming rework estimation hours in CSV data file. Features are the major data for any prediction model. This paper also used to set the feature for rework estimation so as to find the exact fault occurrence place of rework. In the future this rework CSV data is used to automate the rework effort estimation.

2 Convolutional Neural Networks (CNN)

A class of deep learning neural networks are used to form the convolutional neural network. CNN is mainly used in image processing and recognition. It is the backbone of artificial intelligence [5–9]. CNN is a category of machine learning algorithm used to perform deep learning, to learn various features in data that differentiates it from another data. CNN pre-processes the data by itself. It does not require a lot of resources in data processing. CNN mainly takes image dataset as input and learns various features through filters by assigning weights and bias to it. It allows filters to learn important features in the dataset, allowing them to distinguish one input from another. CNN has the ability to pre-process the data by itself, so there is no need for more resources in data pre-processing. CNN is continuously evolving with the data growth. During the training set formation, CNN can be able to adapt the learnt features and develop knowledge discovery of its own so that it continuously develops with growth in the dataset.

Dataset is formed from all categories like text, images, audio and video. The formed dataset is considered as the feature of dataset of the model. For images, the feature set is formed by the process known as segmentation [10–14].

CNN has weights that can learn from the input and the biases. The neuron connected to the network receives the input and performs dot product in it. A loss function is evaluated to measure the performance of the model. The only key point of CNN is that it assumes the input in an explicit manner. The CNN technique is used for creating the RIR learning model structure.

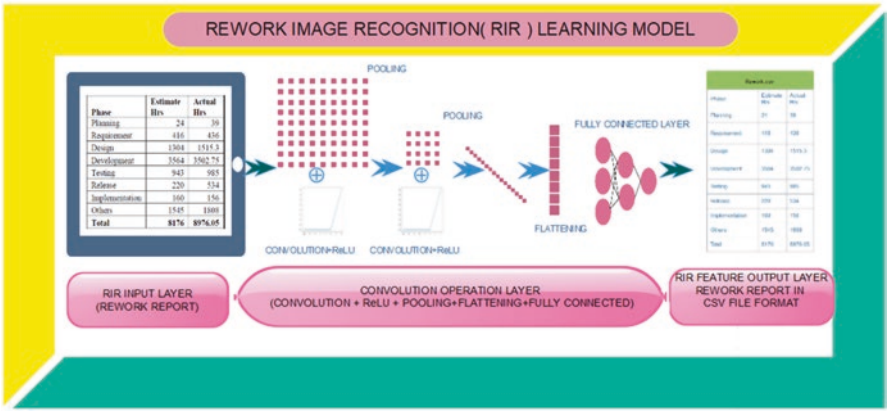


Fig. 1 RIR learning model

3 RIR (Rework Image Recognition) Learning Model

The Rework Image Recognition (RIR) learning model is the deep learning model with CNN algorithm. This model is used to analyse and recognize the image of Rework report produced by Organization-A. The structure of RIR learning model is described in Fig. 1.

CNN learning structure is also formed with three major layers as the ANN structure. The first layer is the input layer, which consists of greyscale image. The second layer is formed by hidden layer set [15–20]. The hidden layers are further divided into convolution layers, ReLU (Rectified Linear Unit) activation function layer, pooling layers and fully connected neural network. The third and the final layer is the output layer, which is a binary or multi-class label.

The RIR model is created by using three major layer structures called RIR input layer, RIR convolution operation layer and RIR feature output layer.

3.1 RIR Input Layer

This CNN learning structure is used for feature detection. This is the beginning layer of the feature mapping. This layer takes the image as the input. The input is provided in the form of pixel. Here filters are applied to represent the image parts. This representation will form a pattern of feature image mapping.

3.2 RIR Convolution Operation Layer

This is the major part of the RIR learning model. In this layer first the image is convoluted with activation function, pooled and flattened so as to form the final fully connected layer.

3.2.1 Convolution

Convolution means formation of representing the image parts. This is used to form the feature map of the image. Then a filtering is used to form network in the image representation. A random vector is generated with network weights and bias value, which is to be shared among the CNN network for the formation of a unique feature set. These are also known as kernels. The filter size is used to measure and identify the pixel affecting rate one at a time. A common filter size used in CNN is 3. It means the height and width of the image in the 3 × 3 pixel area. The depth of the two-dimensional image is represented by the colour channel. For greyscale image the colour channel is 1, and for RGB colour image, the channel value is 3, so the filter size of the colour image is 3 × 3 × 3. The image complexity is preserved by feature mapping, which includes the process of filtering in more than once. The network multiplies the filter value with the pixel value.

3.2.2 ReLU Activation Function

The feature map formed by convolution is passed to the activation function, but in image all the values are in a nonlinear form. The nonlinear value is to be changed into linear so as to insert into the activation function. The major activation function of image processing is ReLU (Rectified Linear Unit).

It comes under a nonlinear activation function. The special quality of this function is that the neurons are not activated at the same time. It is best for hidden layers only. It has output 0 if the input is less than 0, and it has raw output otherwise. It avoids backpropagation errors. It is very close to the working of neuron, to normalize the different range values and make the result between 0 to 1 and −1 to 1. This activation function is used. The ReLU equation is defined in Eq. (1):

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{1}$$

3.2.3 Max Pooling

The activated image data is passed into the pooling layer. Now this layer will remove the unwanted image parts and keeps only the relevant image features. It means this layer only serves on the relevant image parts. Overfitting of data is avoided because of identifying only the relevant parts.

3.2.4 Flattening

Before forming an image into a tabular form, overfitting is to be reduced. The reduction is done between the neurons. This is done by flattening. Flattening means adding a dropout layer to prevent overfitting of CNN algorithm, while training the data dropout is used to reduce the correlation between the data in activation map.

3.2.5 Fully Connected Layer

This layer is the last part of the convolution neural network. Here the dropout layer produced in the previous flattening convolutional operation layer is converted into the vector. In order to capture the relationship of image complexity between the features, the formed vector is fed into this fully connected layer. The final result of this layer is one-dimensional feature vector.

The fully connected layer is also known as dense layer or ANN layer. This layer is used to analyse the input features in order to form classification. First of all, the neuron collections are formed to represent the various parts of the object considered. This neuron collection is stored in a set if the collection is set until fully formed. If the set is full, then the neurons are activated in accordance with the input image for classification of the object.

3.3 RIR Feature Output Layer

This layer is used to form the final resultant CNN layer of the feature mapping. This layer is used to perform the image feature mapping from the previous set of convolution operation layers. Here the accuracy and loss values are calculated. The accuracy value more than 75% is considered for feature mapping classification.

In order to find the accuracy value, the difference in the expected and the calculated result in the training set is found by ANN. The learning rate is adjusted in accordance with the error value to optimize the proposed model performance. The above process is repeated to the learnt association of feature input in accordance with the result. The final fully connected layer will receive the result of the layer before the delivery. At this step, the image classifier model is trained. Now by

passing the image into the CNN model, one will get the content feature mapping of the image. Then it is stored into the CSV file for future prediction.

4 RIR Learning Model Implementation

RIR implementation is achieved by using the four major steps: input dataset formation, filtration, convolution operation and feature extraction.

4.1 RIR Input Layer Preparation

The image is taken into the RIR model to prepare the input layer. Further the input set is formed with filter details.

4.1.1 Input Dataset Formation

To form the CSV file dataset from the image, the Rework report of 'Organization-A' shown in Fig. 2 is considered.

Machine learning data must have a feature. In this paper, image of Rework report is used as the data feature is used in CNN input feature for prediction. From an existing image dataset, ML learns how to gather the desired results with minimal error rate.

Features are input data of neural network. In image recognition, the feature is a group of pixels for pattern analysis. The first step is the conversion of the image into the greyscale level. This is done by using the Python code 'cvtColor' from a cv2 file. The greyscale-converted image of Rework report in Fig. 1 is shown in Fig. 3.

Fig. 2 Rework report of Organization-A

Phase	Estimate Hrs	Actual Hrs
Planning	24	39
Requirement	416	436
Design	1304	1515.3
Development	3564	3502.75
Testing	943	985
Release	220	534
Implementation	160	156
Others	1545	1808
Total	**8176**	**8976.05**

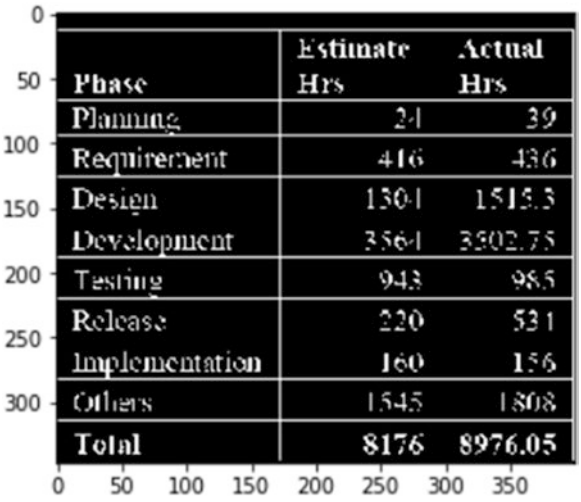


Fig. 3 Plot of greyscale conversion of Rework report

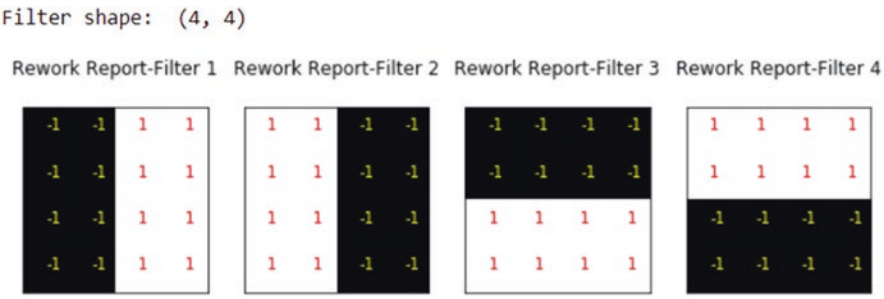


Fig. 4 Filter shape for CNN learning structure

Figure 1 shows the Rework report of 'Organization-A' in image format. This image data is converted into CSV file using CNN technique. The Python code is used to input the image shown below.

4.1.2 Filter Formation

This is the first step of CNN learning structure. A randomly generated array is used to measure and identify the pixel affecting rate one at a time. A common filter size used in CNN is 3. In our model creation, the filter used is shown in Fig. 4 from the Python result.

```
Net(
  (conv): Conv2d(1, 4, kernel_size=(4, 4), stride=(1, 1), bias=False)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
)
```

Fig. 5 Convolution model definition

4.2 RIR Convolution Operation Layer

CNN (convolutional neural network) operation is done by five major steps known as convolution, activation function, pooling, flattening and fully connected layer formation.

4.2.1 Convolution

The convolution layer is defined by using PyTorch Python library. Here the net weight of CNN learning structure is defined, and the designed model is displayed with the help of assigning net (weight) to model variable. The result is shown in Fig. 5.

The maximum convolution layers in reference with Rework report of Organization-A of Fig. 2.

4.2.2 ReLU Activation Function

This layer considers the feature map in order to identify the nonlinear relationship that occurs in Fig. 6. Dimensionality reduction is applied here for the reconstruction of the image. This is done by creating a Python user-defined function 'activation_layer' for applying relu() activation function (Fig. 7).

4.2.3 Pooling

This layer is used for further reduction of nonlinear feature of convoluted value in order to reduce the difficulty in big data. The technique used here is max pooling which is defined by a user-defined function 'pooling_layer' using Python as shown in Fig. 8.

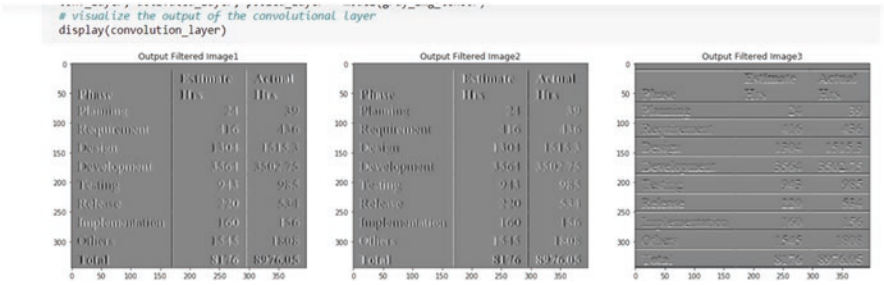


Fig. 6 Full convolution result of greyscale conversion of Rework report

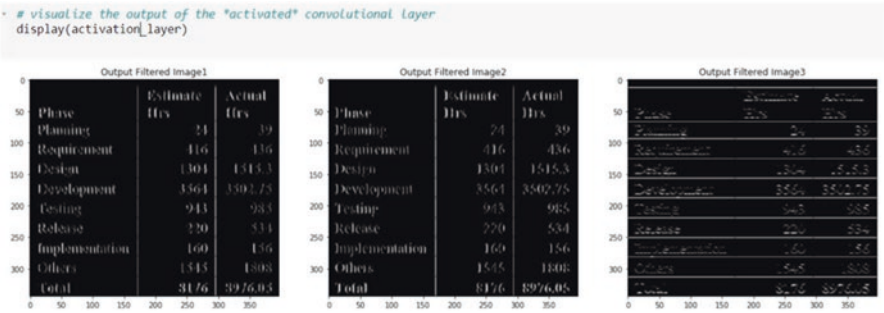


Fig. 7 Full ReLU activation layer result of greyscale conversion of Rework report

4.2.4 Flattening

After pooling flattening is used to convert the pooled image into a single long column sequential order known as vector. This flattening helps for forming image prediction model.

First shape is used to find the image shape, and then flatten() method of NumPy Python library is used.

The flattening result is shown in Fig. 9.

4.2.5 Fully Connected Layer

In this final convolution step, ANN is added with the CNN. It means the ANN attributes are combined for class of prediction with a greater level of accuracy. The error loss is also calculated and adjusted with the feature in order to optimize the model. This work is repeated till the maximum accuracy level of the model appeared. The following steps show the prediction execution result using Python.

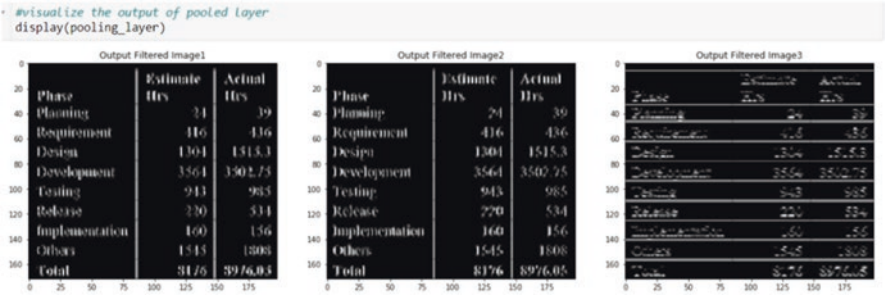


Fig. 8 Full max pooling layer result of greyscale conversion of Rework report

```
img.shape
```

```
(347, 400)
```

```
import numpy as np
img_data = np.array(img)
flattened = img_data.flatten()
flattened.shape
```

```
(138800,)
```

```
flattened
```

```
array([255, 255, 255, ..., 255, 255, 255], dtype=uint8)
```

Fig. 9 Flatten result of greyscale conversion of Rework report

Step 1: Data Pre-processing

- Collect the image after flattening to train it. This is done by using the method imread('image path') of Python library cv2.
- Here the input X_train and x_test dataset is normalized. To do this first by using the Python code astype('float32'), the training set values are converted and it is divided by 255.0.
- Now the y_train and y_test datasets are encoded by using np_utils.to_categorical(), and the y_test data shape is assigned to class_num.

Step 2: Model Creation

- Different hyper-parameters and activation functions are executed; finally all the models are built and summarized in Python, and the corresponding results are shown in Fig. 10.

Step 3: Model Fitting

Here the model created in the previous layer is instantiated, and the value is fitted into the training data prepared. Here the model time period is defined within the fit function by using epochs. If the epoch value is

```
model.build(input_shape)
```

```
model.summary()
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dropout_67 (Dropout)         (600, 32, 32, 1)          0
_________________________________________________________________
conv2d_64 (Conv2D)           (600, 30, 30, 32)         320
_________________________________________________________________
activation_71 (Activation)   (600, 30, 30, 32)         0
_________________________________________________________________
max_pooling2d_22 (MaxPooling (600, 15, 15, 32)         0
_________________________________________________________________
dropout_68 (Dropout)         (600, 15, 15, 32)         0
_________________________________________________________________
batch_normalization_55 (Batc (600, 15, 15, 32)         128
_________________________________________________________________
conv2d_65 (Conv2D)           (600, 15, 15, 64)         18496
_________________________________________________________________
activation_72 (Activation)   (600, 15, 15, 64)         0
_________________________________________________________________
max_pooling2d_23 (MaxPooling (600, 7, 7, 64)           0
_________________________________________________________________
dropout_69 (Dropout)         (600, 7, 7, 64)           0
_________________________________________________________________
batch_normalization_56 (Batc (600, 7, 7, 64)           256
_________________________________________________________________
conv2d_66 (Conv2D)           (600, 7, 7, 128)          73856
_________________________________________________________________
activation_73 (Activation)   (600, 7, 7, 128)          0
_________________________________________________________________
dropout_70 (Dropout)         (600, 7, 7, 128)          0
_________________________________________________________________
batch_normalization_57 (Batc (600, 7, 7, 128)          512
_________________________________________________________________
flatten_9 (Flatten)          (600, 6272)               0
_________________________________________________________________
dropout_71 (Dropout)         (600, 6272)               0
_________________________________________________________________
dense_35 (Dense)             (600, 256)                1605888
_________________________________________________________________
activation_74 (Activation)   (600, 256)                0
_________________________________________________________________
dropout_72 (Dropout)         (600, 256)                0
_________________________________________________________________
batch_normalization_58 (Batc (600, 256)                1024
_________________________________________________________________
dense_36 (Dense)             (600, 128)                32896
_________________________________________________________________
activation_75 (Activation)   (600, 128)                0
_________________________________________________________________
dropout_73 (Dropout)         (600, 128)                0
_________________________________________________________________
batch_normalization_59 (Batc (600, 128)                512
=================================================================
Total params: 1,733,888
Trainable params: 1,732,672
Non-trainable params: 1,216
_________________________________________________________________
```

Fig. 10 Model summary of the Rework report

```
print(accuracy)

782/782 [==============================] - 30s 38ms/step - loss: 0.3097 - accuracy: 0.8742
782/782 [==============================] - 30s 38ms/step - loss: 0.3097 - accuracy: 0.8742
0.8741999864578247
0.8741999864578247
```

Fig. 11 Accuracy and loss level of the Rework report

increased, then the model performance is also increased. In some cases it may cause an overfitting result. Therefore, proper epoch's value will save the network weights during data training.

Step 4: Model Evaluation

It is done by checking the accuracy level with that of the loss value of the model in the epochs. The accuracy level of the sequential model used for the image recognition is shown in Fig. 11.

From Fig. 11 it is clearly shown that the image recognition accuracy level is 87% so the image is used for Rework report analysis.

After all the convolution layer steps are done, the image is reconstructed and the new grey level is displayed in Fig. 12.

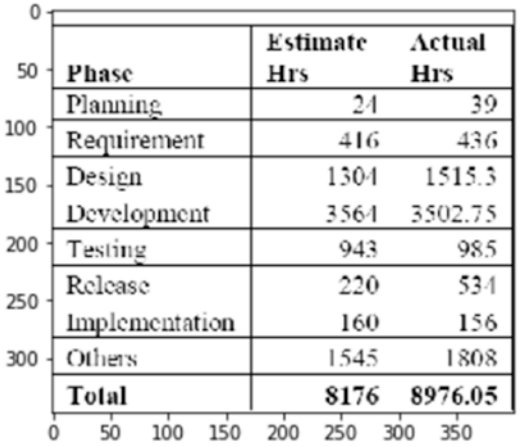
4.3 RIR Feature Output Layer

Here the final improved Rework report of the Organization-A is considered for CSV file conversion. It is done by using the command image_to_string of the Python library, and the result of the command is saved with the help of write() method. The saved file is opened by using the open() command, and the corresponding converted CSV file of Fig. 12 is shown in Fig. 13.

5 Conclusion

This paper is used to propose a new learning model (RIR) for image recognition using Python. The RIR learning model is based on CNN deep learning algorithm. The model is implemented with various performance measures like accuracy and loss values. The RIR accuracy level is 87%. This result shows that the image recognition has reached its maximum level, so that it can be used for CSV file conversion. The converted CSV file is to be used as the data for rework effort estimator of the Organization-A. This research work helps reach maximum image recognition, which will be used to insist the need of fault taxonomy to predict fault in the early software development phases.

<matplotlib.image.AxesImage at 0x1e58529d908>



Phase	Estimate Hrs	Actual Hrs
Planning	24	39
Requirement	416	436
Design	1304	1515.3
Development	3564	3502.75
Testing	943	985
Release	220	534
Implementation	160	156
Others	1545	1808
Total	**8176**	**8976.05**

Fig. 12 Final improved plot Rework report

Fig. 13 Converted CSV file of the Rework report

```
img_csv_read.head(10)
```

	Estimate | Actual
0	Phase Hrs Hrs
1	Planning 24 39
2	Requirement 416 436
3	Design 1304 | 1515.3
4	Development 3564 | 3502.75
5	Testing 943 985
6	Release 220 534
7	Implementation 160 156
8	Others 1545 1808
9	Total 8176 | 8976.05

References

1. Patchaiammal, P., Thirumalaiselvi, R.: Software fault prediction exploration using machine learning techniques. Int. J. Recent Technol. Eng. **7**(6S3), 109–113 (2019) ISSN: 2277-3878. Published By: Blue Eyes Intelligence Engineering & Sciences Retrieval Number: F1022376S19/19©BEIESP Publication
2. Rajesh, M.: A signature-based information security system for vitality proficient information accumulation in wireless sensor systems. Int. J. Pure Appl. Math. **118**(9), 367–387 (2018)
3. Patchaiammal, P., Thirumalaiselvi, R.: Genetic evolutionary learning fitness function (GELFF) for feature diagnosis to software fault prediction. Int. J. Innov. Technol. Explor. Eng. **8**(11S), 1151–1161. ISSN: 2278-3075. Published By: Blue Eyes Intelligence Engineering & Sciences Publication Retrieval Number: K123309811S19/2019©BEIESP (2019). https://doi.org/10.35940/ijitee.K1233.09811S19
4. Mateen, A., Nazir, M., Awan, S.A.: Optimization of test case generation using genetic algorithm (GA). Int. J. Comput. Appl. **151**(7), 6–14 (2016)
5. Patchaiammal, P., Thirumalaiselvi, R.: Enrichment of fault features by forming ML hypothesis. Test Eng. Manag. **82**, 6276–6288 (2020) ISSN: 0193-4120. Publication Issue: January–February 2020 Published by: The Mattingley Publishing Co., Inc
6. Siegrist, K.: Hypothesis testing – introduction. www.randomservices.org. Retrieved March 8, 2018
7. Patchaiammal, P., Thirumalaiselvi, R.: GKS algorithmic technique for early defect prediction (GKS: a genetic feature K-means clustering with support vector machines). Int. J. Psychosoc. Rehabil. **24**(01), 1805–1822. ISSN: 1475-7192 (2020). https://doi.org/10.37200/IJPR/V24I1/PR200282
8. Just, R., Jalali, D., Ernst, M.D.: Defects4J: a database of existing faults to enable controlled testing studies for Java programs. In: ISSTA 14, July 21–25, 2014, San Jose, CA, USA
9. Dallmeier, V., Zimmermann, T.: Automatic extraction of bug localization benchmarks from history. In: ASE '07, November 2007
10. https://towardsdatascience.com/human-pose-estimation-with-stacked-hourglass-network-and-tensorflow-c4e9f84fd3ce
11. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018
12. Pranav, J.V., Anand, R., Shanthi, T., Manju, K., Veni, S., Nagarjun, S.: Detection and identification of COVID-19 based on chest medical image by using convolutional neural networks. Int. J. Intell. Netw. **1**, 112–118 (2020)
13. Thakkar, K., Goyal, A., Bhattacharyya, B.: Emergence of deep learning as a potential solution for detection, recovery and de-noising of signals in communication systems. Int. J. Intell. Netw. **1**, 119–127 (2020)
14. Arulmurugan, R., Anandakumar, H.: Early detection of lung cancer using wavelet feature descriptor and feed forward back propagation neural networks classifier. In: Computational Vision and Bio Inspired Computing. Lecture Notes in Computational Vision and Biomechanics, pp. 103–110. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-71767-8_9
15. Deeplearning4j. https://deeplearning4j.org
16. https://en.wikipedia.org/wiki/Precision_and_recall
17. https://bugs.eclipse.org/bugs/buglist.cgi?quicksearch=functional
18. https://www.bugzilla.org
19. https://commons.wikimedia.org/wiki/File:Convolution_arithmetic_-_Same_padding_no_strides.gif
20. Shih, Y.-F., et al.: Deep co-occurrence feature learning for visual object recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2017)