




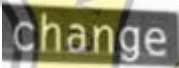




CHAPTER 4

ANALYSIS AND DESIGN

4.1 Analysis

The dataset used is taken from the IIIT 5K word dataset made by Anand Mishra, Karteek Alahari and C. V. Jawahar. And which aims for Scene Text Recognition using Higher Order Language Priors. The IIIT 5K words dataset contains both scene text and born-digital images (a category which recently gained interest in ICDAR 2011 competitions). Born-digital images are inherently low-resolution (made to be transmitted on-line and displayed on a screen) and have variety of font sizes and styles. On the other hand, scene texts are already considered to be challenging for recognition due to the presence of varying illuminations, projective distortions. This dataset is not only much larger than public datasets like SVT and ICDAR2003. IIIT k5 words has 2 parts, namely "Test and Train" and each has 3000 Train and 2000 Test and totals 5000. One dataset image consists of a series of letters or words and numbers and has various sizes, the dataset shows pieces of a word that vary with purpose for the efficiency of processing the dataset. This dataset is in RGB format.

Table 4.1: Tabel Analisis Data

Example :	Text :
 Illustration 4.1: 6_7.png	Loans
 Illustration 4.2: 30_6.png	DELIVERY
 Illustration 4.3: 386_3.png	PIRATES
 Illustration 4.4: 431_3.png	change
 Illustration 4.5: 419_10.png	Banks
 Illustration 4.6: 415_5.png	ROAD
 Illustration 4.7: 616_3.png	prime
 Illustration 4.8: 697_16.png	business

 <p>Illustration 4.9: 826_2.png</p>	FASTER
 <p>Illustration 4.10: 850_11.png</p>	Password

4.2 Pre-processing data

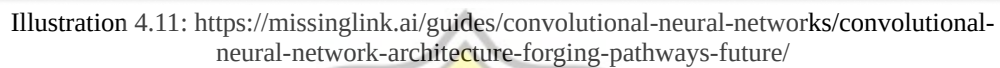
First the train and test data are processed and labeled in the initial way, namely loading the dataset and after loading, the image is made into one same size, namely at size 50 and the image will be categorized and separated into Train data and Test data. After labeling, we will get the results are "X_train and Y_test". Then the image will then be created in Grayscale color and the image will be converted into an array value, the next step is normalization so that the RGB image can be 2 channel only (Black and White) divided by a value of 255. Next the image will be restored to its original size and results. From the image that has been made in the original size, it will be converted into an array with Numpy and saved in binary format.

4.3 Feature Extraction Stem

The process that occurs in this section is to "encode" an image into features in the form of numbers that represent the image (Feature Extraction). The feature extraction layer consists of two parts, namely the Convolutional Layer and the Pooling Layer. A convolutional layer consists of neurons arranged in such a way as to form a filter with length and height (pixels). Convolutional's task is to create feature maps for the input image. In this case, the Convolution layer uses a 3x3 kernel size so that the feature maps process has detailed results. Then the data is entered into the activation process, in this case the RELU activation is used, namely the abbreviation of rectified linear unit. And it has the task of It effectively removes negative values from an activation map by setting them to zero, has the mathematical formula $F(x) = \max(0, X)$. Then after going through the RELU activation process, the next step is to Maxpooling the image which aims to retrieve information importance of an image based on pixel characters.

Maxpooling has the math formula $f_{x,y}(S) = \max_{a,b=0}^1 S_2 X + a, 2Y + b$.

Then the data is reprocessed with a 3X3 convolution layer with a hidden layer 64 using RELU and maxpooling again to perform the same process again from the convolution layer, activating RELU, Maxpooling images, and then forwarding to the branch merging feature.



At the Feature stage, the branch merging, the output of the stem extractor feature is reprocessed by a convolution layer process using a 3X3 kernel size with 128 hidden layers, which aims to produce detailed feature maps that are easy to recognize and process. Then concat the convolutional layer feature stem with the convolutional layer of the feature merging branch and after that unpool to find the important value of the pixels in the image. Then enter the RELU activation function, RELU activation is a rectified linear unit. And it has the task of It effectively removes negative values from an activation map by setting them to zero, has the mathematical formula $F(x) = \max(0, X)$. Next Step is to enter the convolutional layer with a 3X3 kernel with 64 hidden layers then merge the layers again with the convolutional layer with the Merging stem Feature. And then reprocessed in a 3X3 kernel sized convolutional layer with 32 hidden layer and unpooling the layer again, then the output is recalculated using RELU activation and Sigmoid activation function. Then the train model will be compiled by calculating the loss using binary cross-entropy and optimizing the learning rate using the adam optimizer then doing a fit model with a batch size of size 32 and

doing training steps using EPOCH 10 times and after that the model will be saved as East_text. Hard tesnsorflow format.

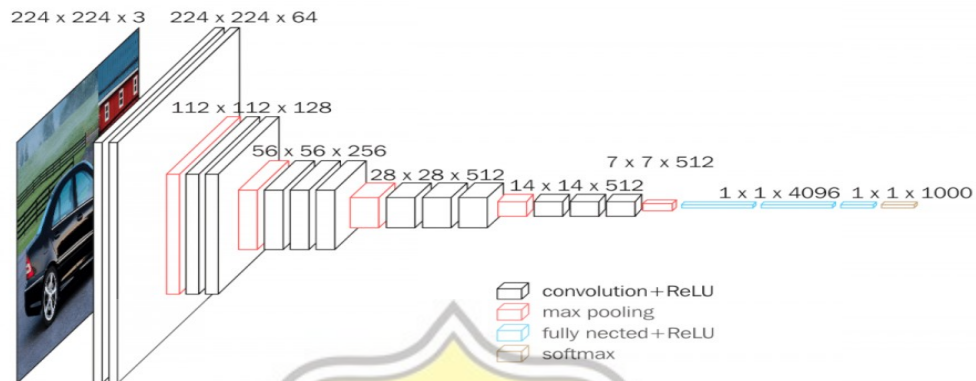


Illustration 4.12: <https://neurohive.io/en/popular-networks/vgg16/>

4.5 Saving model

The saving model process is carried out after the PVAnet process and the Branch Merging Feature. The data will enter the compile model process before entering the fitting model. Compile model is performed using sequential model with loss function calculation, namely binary_crossentropy

$$- \frac{1}{\text{outputsize}} \sum_{i=1}^{\text{outputsize}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log 1 - \hat{y}_i$$

Adam's optimizer

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned}$$

Calculation accuracy using accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

The compiled model will enter the fitting model process to be trained, the train process will be carried out with `batch_size = 32`, `epochs = 10`, `validation_split = 0.3`, by validating the `X_train` data to `Y_test`. And saving model using `'model.save'` with tensorflow.keras format

4.6 Text Detection

At this stage, the saved model train will be reloaded to detect text in the image input. the incoming image will be identified by the model train so that the model can detect the text contained in the image to help the model recognize the text area in the image using openCV using the RBOX geometry method, RBOX geometry has a detection character with an upright position such as writing or a series of alphabets with a standard or straight sideways position. and QUAD geometry, Quad geometry is also usually used to detect the placement of various text in an image, such as writing in a curved model that is not upright and has many angles. for the stage of displaying the text that has been detected using the pytesseract library which functions to extract image to text.

4.7 Evaluation

for the evaluation of our EAST model. Comparisons will be made with the original Based EAST. original EAST will be trained with our dataset, which contains Train 3000 and Test 2000 (from IIIT 5K word). After that the two models will each detect the same 5 images with various sizes to measure the accuracy based on the character of the image testing. Measuring accuracy will be done with

the formula :
$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$