Epoch Tuning Hyperparameter in Fire Image Classification at University Sjakhyakirti

Ahmad Bahri Joni Malyan[1], Rian Rahmanda Putra[1(✉)], Ema Laila[1], Agum Try Wardhana[1], Muhammad Fikri[1], and Indra Griha Tofik Isa[2]

[1] Department of Computer Engineering, Politeknik Negeri Sriwijaya, Palembang, Indonesia
rianrahmanda@polsri.ac.id

[2] Department of Informatic Management, Politeknik Negeri Sriwijaya, Palembang, Indonesia

Abstract. Epoch is a factor that affects the time of training an AI model and affects the accuracy value of the AI model. In this study, hyperparameter tuning of the epoch value was carried out to see the optimization of the resulting accuracy value. The object to be observed is the condition of the fire image which is classified in the position of (1) Fire Condition, and (2) Non-Fire Condition. The image data observed were 4650 fire images sourced from environmental images of the University Sjakhyakirti as well as fire conditions from public sources like Google images and Kaggle. So the formulation of the problem from this research is how to measure the Epoch value in producing fire image prediction accuracy. The purpose of this study is to measure the best accuracy value by performing hyperparameter tuning on the Epoch. The stages of the research are (1) Dataset Integration; (2) Image Augmentation; (3) Data Pre-processing; (4) CNN modelling; (5) Hyperparameter Tuning; (6) Determination of Accuracy Value. The tuning hyperparameters carried out are with values of 30, 50, 75, 100 and 120. From the results of the Epoch tuning hyperparameter, based on the effectiveness of the time required along with the accuracy results, the epoch 30 value has an optimal accuracy result of 82.5% of the test data testing, and has The effective time duration is relatively short, namely 30 min assuming the time required is ±1 min for one epoch.

Keywords: Hyperparameter tuning · Epoch · Image Processing · Fire Image Classification

1 Introduction

The ease of using technology results in a large amount of data available on the internet. This data can be used to dig up useful information for users [1]. The industrial revolution 4.0 has characteristic in utilizing data into new knowledge, through artificial intelligence, internet of things technology, big data technology and other technologies that utilize these data. Especially in artificial intelligence, where human intelligence is modeled or simulated on devices and machines [2]. One of the aspects of artificial intelligence is Computer Vision, which means the ability of a device, in this case a computer, to view and interpret visual data into a parameter to be processed into certain outputs. Of course,

N. L. Husni et al. (Eds.): FIRST-ESCSI 2022, AHE 14, pp. 355–364, 2023.
https://doi.org/10.2991/978-94-6463-118-0_37

in Computer Vision, a series of techniques and visual image processing are needed, which is called image processing or image processing.

Image processing technology has now developed and is widely implemented in technological devices, including fingerprint image recognition, where Convolutional Neural Network implementation is carried out for the fingerprint image classification process which produces training data accuracy outputs up to 100% [3]. The CNN AI model is also used in image-based classification of Genus Panthera cats with a training dataset of 3840 images. The CNN architecture used consist of 1 input layer, 5 convolutional network and 2 fully connected layers. The CNN model has an accuracy value of 92,1% [4]. While other image processing implementations are used for skin cancer classification through the VGG-16 architecture, which results in an accuracy of 99.70% [5].

Convolutional Neural Network is one of the algorithms that is often used in image processing which has similarities with neural networks algorithms in general where the network can be displayed visually as a collection of neurons arranged as an acyclic graph [6]. The main difference from common neural networks is that hidden layer neurons are only connected to a subset of neurons in the previous layer [7]. Due to this rare connectivity, it is able to learn features implicitly. In general, CNN has 4 components called: (1) Convolution layer (2) ReLU layer (Activation Function) (3) Pooling layer (4) Fully connected layer. CNN is a model that has received attention because of its classification ability based on contextual information.

In the study conducted implementing the CNN algorithm to measure the level of accuracy of the predictions generated. In optimizing the CNN algorithm, it is necessary to perform hyperparameter tuning for several aspects, including adjusting the Epoch value to measure the best accuracy of the observed model. The object to be observed is the condition of the fire condition is classified in the positions of (1) Fire Conditions, and (2) Non-Fire Conditions. The problem formulation of this research is how to measure the optimum Epoch value in producing fire image prediction accuracy. As for the objects observed, 4650 fire images sourced from environmental images of the Universitas Sjakhjakirti and fire conditions from public sources, google image. The purpose of this study is to measure the best accuracy value by performing hyperparameter tuning on the Epoch, so as to produce the highest accuracy value through the epoch measurement scenarios of 30, 50, 75, 100 and 120.

2 Methodology

The study was conducted with 4650 fire image objects. Software used online through Google Colab with Python programming. The research methodology used in this study can be seen in Fig. 1.

Dataset integration is carried out using the content-based image retrieval (CBIR) method where the images used are retrieved from Google Image using the keywords “fire”, “building fires”, “forest fires” for data that will be labeled fire and the keywords “comfortable atmosphere”, “building atmosphere” and “forest atmosphere” for data labeled “Non-Fire”. The dataset used is 4650 images that have been annotated. The dataset property used consists of two labels, namely FIRE and NON_FIRE, with the number of data FIRE. Where FIRE data is 2865 and NON_FIRE is 1785 data, while the

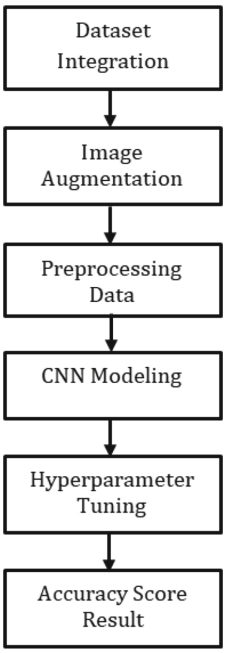

Fig. 1. Research Methodology

percentage for the distribution of test data is 25%, which is 1162 data. The image used is a color image type.

The next stage is image augmentation which aims to provide tolerance flexibility for the model when receiving fire or non-fire images with non-ideal conditions, such as certain zoom conditions, shear conditions, and so on. In this research, the implemented image augmentation are (1) Rescale; (2) Rotation Range; (3) Zoom Range; (4) Shear Range; (5) Fill Mode.

CNN modeling by implementing convolution, relu, and pooling. Convolution is done to simplify the image through matrix calculations making it easier for the system to read the dataset from the observed image. Then the ReLu method is used to assign a value of 0 to the numbers that produce negative values in the convolution stage [8]. While pooling is done to determine the highest value based on the results of matrix calculations [9].

The next step is to perform hyperparameter tuning on the epoch. Epoch is a hyperparameter that determines how many times the learning algorithm will work to process the entire training dataset. This epoch value is an important hyperparameter for the learning algorithm. One epoch shows that each sample in the training dataset has the opportunity to updaet the internal model parameters [10].

Fig. 2. Dataset image FIRE and NON FIRE

3 Result and Discussion

3.1 Dataset Integration

As explained in the previous chapter, the data taken is from the condition of Sjakhjakirti University and fire images from public repositories such as google image and kaggle. In dataset integration, data extraction and image classification are carried out into the Train and Test repository, each of which consists of a FIRE folder for fire image conditions and NON_FIRE for non-fire image conditions. The observed image is visualized in Fig. 2 using the matplotlib.pyplot and matplotlib.images libraries.

From the dataset above, it can be seen that there are several fire conditions in which there is a fire or called FIRE and there is no fire condition or NON_FIRE. For FIRE conditions, it is indicated by the smoke and fire around the image, while for NON_FIRE conditions, it is indicated by normal room conditions, no smoke and fire.

3.2 Image Augmentation

Image Augmentation is done to increase the accuracy of the modeling to be built, so that the model can provide tolerance for some outlier image conditions but still in the context of the observed image domain [11]. Before setting the image augmentation, first step is to observe the condition of the FIRE and NON_FIRE image data by looking at (1) the size of the object of the image (zoom condition or not); (2) How is the slope of the object, whether the resulting horizontal line is even or not; (3) the distortion of the vehicle driver's image. So there are 4 parameters set in Image Augmentation using the ImageDataGenerator library from tensorflow, as shown in Table 1.

Based on Table 1, it can be seen that the rotation range value is 20, which indicates that the model will tolerate the image being tested or trained if there is a slope of 200 to the right or to the left. The zoom range value is 0.2, which means that it provides

Table 1. Image Augmentation Parameter

Number	Variable	Value
1	Rescale	1/255
2	Rotation Range	20
3	Zoom Range	0.2
4	*Shear Range*	0.2

```
BASE_PATH = '/content'
PROJECT_PATH = join(BASE_PATH, 'Project')
DATASET_PATH = join(PROJECT_PATH, "Datasets")

EXTRACT_PATH = join(DATASET_PATH, "EXTRACT")

TRAIN_PATH = join(EXTRACT_PATH, "TRAIN")
FIRE_PATH = join(EXTRACT_PATH, "TRAIN", "FIRE")
NON_FIRE_PATH = join(EXTRACT_PATH, "TRAIN", "NON_FIRE")

TEST_PATH = join(EXTRACT_PATH, "TEST")
FIRE_TEST = join(TEST_PATH, "FIRE")
NON_FIRE_TEST = join(TEST_PATH, "NON_FIRE")

folders = [PROJECT_PATH, DATASET_PATH, EXTRACT_PATH,
          TRAIN_PATH, FIRE_PATH, NON_FIRE_PATH,
          TEST_PATH, FIRE_TEST, NON_FIRE_TEST]

for folder in folders:
  mkdir_if_not_exist(folder)
```

Fig. 3. The process of bulding Spesific Folder for FIRE and NON_FIRE

tolerance for images with the condition of the object zooming in up to 20%. While the Shear Range has a value of 0.2 which means it provides tolerance for images with a deviation of up to 20%.

3.3 Pre-processing Data and Modelling

In the preprocessing stage, the data is done by dividing the training data into 2 parts, namely training data and validation data, each of which consists of 2 classes, namely a class with a label with a value of FIRE and a label with a value of NON_FIRE. The percentage for the distribution of training data is 75%, which is 4650 data which is divided into FIRE as many as 2865 and NON_FIRE as many as 1785 data, while the percentage for distribution of validation data is 25%, which is 1162 data. Data preprocessing is done by using the zipfile library as image data extraction, which is finally inserted to the specific folder. The code as shown in Fig. 3.

```
fire_images_folder = join(EXTRACT_PATH, "fire_dataset", "fire_images")
non_fire_images_folder = join(EXTRACT_PATH, "fire_dataset", "non_fire_images")
try:
  move_all_file(non_fire_images_folder, NON_FIRE_PATH)
  move_all_file(fire_images_folder, FIRE_PATH)
except:
  pass

print(f"File di folder fire {len(os.listdir(FIRE_PATH))}")
print(f"File di folder non-fire {len(os.listdir(NON_FIRE_PATH))}")

try:
  shutil.rmtree(join(EXTRACT_PATH, "FIRE-SMOKE-DATASET"))
  shutil.rmtree(join(EXTRACT_PATH, "Fire-Detection"))
  shutil.rmtree(join(EXTRACT_PATH, "fire_dataset"))
except:
  pass
```

```
File di folder fire 2865
File di folder non-fire 1785
```

Fig. 4. The results of dividing and reading the number of images on the train data repository

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
class_dataset = ["FIRE", "NON-FIRE"]
jumlah = [len(os.listdir(FIRE_PATH)), len(os.listdir(NON_FIRE_PATH))]
ax.bar(class_dataset, jumlah)
plt.show()
```

Fig. 5. Visualization data train process for labeled of FIRE and NON_FIRE

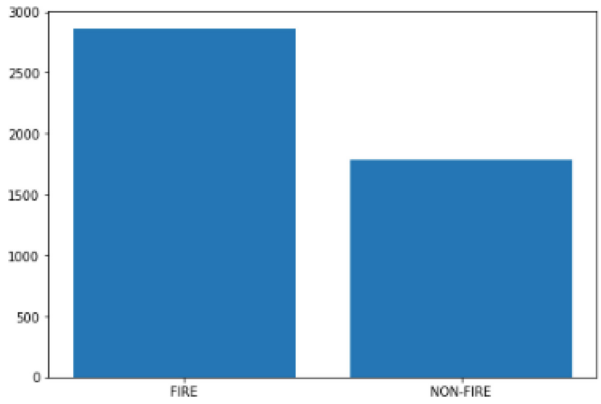

Fig. 6. Visualtization data train graph result for labeled of FIRE and NON_FIRE

After the folder is formed where the repository of the images to be classified. The next step is to divide and read the number of images on each label FIRE and NON_FIRE in the data train as code shown in Fig. 4.

Furthermore, the FIRE and NON_FIRE image datasets are visualized through the matplotlib library so that it can be seen how the comparison between the datasets labeled FIRE and NON_FIRE with the code as shown in Fig. 5.

The results of the visualization are shown in Fig. 6, where the dataset labeled FIRE is worth more than the dataset labeled NON_FIRE.

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(256, 256, 3)),
    MaxPool2D(2,2),

    Conv2D(16, (3,3), activation='relu'),
    MaxPool2D(2,2),

    Conv2D(32, (3,3), activation='relu'),
    MaxPool2D(2,2),

    #Flatten agar data bisa di oleh oleh Dense Layer
    Flatten(),

    Dense(32, activation='relu'),

    Dense(16, activation='relu'),

    Dense(32, activation='relu'),

    Dense(1, activation='sigmoid')
])
```

Fig. 7. The implemented of Modelling

After the data preprocessing process through labeling and data visualization, the next is Modelling which is done by building CNN modeling through Convolution, Relu and Pooling. Convolution process is carried out 3 times with Convolution 1 is 32, Convolution 2 is 16, and convolution 3 is 32. From each of these convolutions a relu process is carried out which aims to give a value of 0 for the convolution process that produces a negative value. As well as pooling to choose the highest value from the convolution results using a matrix pattern (2×2). Figure 7 is the coding implemented in the modeling:

Figure 8 is a resume of the implemented model where the output parameters are 932,897 data generated from training data and the number of convolution/dense.

3.4 Hyperparameter Tuning

In this stage, parameter tuning is carried out which aims to measure the level of accuracy of the implemented model. In addition, it is also part of training data and validation data so as to achieve optimal accuracy points. There is a parameter called Epoch which is then set to train the number of times the training data and validation data are trained to produce model with optimum accuracy. In this study, the epoch values given are 30, 50, 75, 100 and 120. The results of the hyperparameter tuning of the epoch values are shown in Table 2.

In Table 2 it can be seen that the hyperparameter tuning for the epoch with a value of 30 produces an accuracy value of 0.825 which indicates that the accuracy rate is 82.5% with the training process for training data and validation data carried out 3 times. Meanwhile, the highest accuracy results are found in the epoch value of 100 with an accuracy result of 0.835 or if converted into a percentage the accuracy value is 83.5%. While the lowest accuracy value is the Epoch value of 75 with a percentage of 69.5% accuracy value.

The epoch process requires a duration of ± 1 min, which means that if the Epoch value of 30 is running, it requires a duration of 30 min. In terms of effectiveness, when

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 254, 254, 32)      896

 max_pooling2d_6 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_7 (Conv2D)           (None, 125, 125, 16)      4624

 max_pooling2d_7 (MaxPooling  (None, 62, 62, 16)       0
 2D)

 conv2d_8 (Conv2D)           (None, 60, 60, 32)        4640

 max_pooling2d_8 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 28800)             0

 dense_8 (Dense)             (None, 32)                921632

 dense_9 (Dense)             (None, 16)                528

 dense_10 (Dense)            (None, 32)                544

 dense_11 (Dense)            (None, 1)                 33

=================================================================
Total params: 932,897
Trainable params: 932,897
Non-trainable params: 0
_________________________________________________________________
```

Fig. 8. Resume of Modelling Implemented

Table 2. Accuracy Epoch Result

Number	Epoch Value	Accuracy Result
1	Value of 30	0.825
2	Value of 50	0.800
3	Value of 75	0.695
4	Value of 100	0.835
5	Value of 120	0.735

compared with the Epoch value of 100, in terms of the duration of the epoch and the accuracy results are not much different, it can be concluded that the recommended optimum epoch point is the Epoch value of 30.

4 Conclusion

The study conducted to see the performance of Epoch on FIRE and NON_FIRE image conditions with Epoch value of 30, 50, 75, 100, and 120 resulted in different accuracy values. Based from the effectiveness of the time needed in the epoch process and the

resulting accuracy value, the epoch value of 30 is the best epoch value, where the accuracy value generated for the test data is 82.5% while the duration required in the epoch process is 30 min assuming the time required for the epoch process. it takes ±1 min for one epoch.

Acknowledgments. The Research Team would like to thank the Center for Research and Community Service (P3M) of Politeknik Negeri Sriwijaya for research funding support. And to the research partner in University Sjakhjakirti for the research dataset of the FIRE or NON_FIRE image.

References

1. N. K. S. Astini, "PENTINGNYA LITERASI TEKNOLOGI INFORMASI DAN KOMUNIKASI BAGI GURU SEKOLAH DASAR UNTUK MENYIAPKAN GENERASI MILENIAL," in Prosiding Seminar Nasional Dharma Acarya ke-1. Tantangan dan Peluang Dunian Pendidikan di Era 4.0, (2019), no. 1, pp. 113–120.
2. M. H. Santoso, "Application of Association Rule Method Using Apriori Algorithm to Find Sales Patterns Case Study of Indomaret Tanjung Anom," Brill. Res. Artif. Intell., vol. 1, no. 2, pp. 54–66, (2021), https://doi.org/10.47709/brilliance.v1i2.1228.
3. R. D. Nurfita and G. Ariyanto, "Implementasi Deep Learning berbasis Tensorflow untuk Pengenalan Sidik Jari," Emit. J. Tek. Elektro, vol. 18, no. 1, pp. 22–27, (2018), https://doi.org/10.23917/emitor.v18i01.6236.
4. G. A. Anwar and D. Riminarsih, "Klasifikasi Citra Genus Panthera Menggunakan Metode Convolutional Neural Network (Cnn)," J. Ilm. Inform. Komput., vol. 24, no. 3, pp. 220–228, (2019), https://doi.org/10.35760/ik.2019.v24i3.2364.
5. S. R. Reynaldi, J. Apri, and S. A. Wahyu, "Klasifikasi Penyakit Kanker Kulit Menggunakan Metode Convolutional Neural Network," Data Inst. Teknol. Telkom Purwokerto, vol. 2, no. 1, pp. 52–57, (2022).
6. P. S. Bhairnallykar, A. Prajapati, A. Rajbhar, and S. Mujawar, "Convolutional Neural Network (CNN) for Image Detection," pp. 1239–1243, (2020).
7. D. E. Saputra and A. F. Ibadillah, "Pengolahan Citra Digital Dalam Penentuan Panen Jamur Tiram," J. Tek. Elektro dan Komput. TRIAC, vol. 6, no. 1, pp. 2–6, (2019), https://doi.org/10.21107/triac.v6i1.4356.
8. F. F. Maulana and N. Rochmawati, "Klasifikasi Citra Buah Menggunakan Convolutional Neural Network," J. Informatics Comput. Sci., vol. 1, no. 02, pp. 104–108, (2020), https://doi.org/10.26740/jinacs.v1n02.p104-108.
9. M. R. Rasyid, Z. Tahir, and N. Syafaruddin, "Digital Image Processing for Detecting Industrial Machine Work Failure with Quantization Vector Learning Method," J. Pekommas, vol. 4, no. 2, p. 131, (2019), https://doi.org/10.30818/jpkm.2019.2040203.
10. D. Setiawan, J. E. Candra, and C. E. Suharyanto, "Perancangan Sistem Pengontrol Keamanan Rumah dengan Smart CCTV Menggunakan Arduino Berbasis Telegram," InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan), vol. 4, no. 1, pp. 185–190, (2019), https://doi.org/10.30743/infotekjar.v4i1.1598.
11. R. B. Arif, M. A. B. Siddique, M. M. R. Khan, and M. R. Oishe, "Study and observation of the variations of accuracies for handwritten digits recognition with various hidden layers and epochs using convolutional neural network," in 4th International Conference on Electrical Engineering and Information and Communication Technology, iCEEiCT (2018), pp. 112–117. https://doi.org/10.1109/CEEICT.2018.8628078.