International Journal of Innovative Studies in Sciences and Engineering Technology

ISSN 2455-4863 | Open Access | Volume 7, Issue 3, 2021

DOI: https://doi.org/10.20431/2455-4863.070301

Study on Identification of Taylor Vortex Development Process Through Deep Learning

Hiroyuki Furukawa[1], Kensuke Matsuoka[1]

[1]*Department of Mechanical Engineering, Faculty of Science and Technology, Meijo University, Nagoya, Japan*

ABSTRACT

The Taylor vortex flow has been studied as one of important vortexes since the classical research by G. I. Taylor in 1923. When the peripheral velocity of the inner cylinder of a coaxial rotating double cylinder is gradually increased from zero, a Couette flow is generated between the inner and outer cylinders. It has been verified that, when the peripheral velocity is further accelerated, the mode of the flow on the torus, known as a cell, transitions to multi-layered Taylor vortex flow and wave Taylor vortex flow. The Taylor vortex flow can be classified into two types, the normal mode and the anomalous mode. When the top and bottom ends of the double cylinder are fixed, the normal mode indicates a mode wherein there is a flow from the outer cylinder to the inner cylinder at both the top and bottom end walls. In addition, the anomalous mode indicates a mode with flow from the inner cylinder to the outer cylinder at both, or one of, the top and bottom end walls. In this study, we use deep learning to create a program that can automatically identify the result of simulations that determine the flow mod of Taylor vortex flow. For the program created in this study, the machine learning using the method known as supervised learning, using the images from the simulation results of the Taylor vortex flow as input data. Following this, we created a program that can automatically identify whether a flow is in the normal mode and the anomalous mode and the number of vortices based on the images from simulation results.Machine learning is the process of iteratively learning from image data in order to identify the patterns hidden in the same; among these, one of the learning processes that more accurately approximates the human brain is the technology known as deep learning. It uses a multi-layered neural network modeled after the neural circuits of the human brain. Here, a neural network indicates a combination of mathematical models of the neurons in the human cranial nervous system.In this study, approx. 12,000 of 300,000 collected images were selected as input data, and with the created program, it became possible to automatically distinguish between the normal mode and the anomalous mode in terms of these images.

Keywords: Taylor vortex flow, Deep learning, Mode Identification

INTRODUCTION

The Taylor vortex [1, 2] flow has been studied as one of important vortexes since the classical research by G. I. Taylor in 1923. When the peripheral velocity of the inner cylinder of a coaxial rotating double cylinder is gradually increased from zero, a Couette flow is generated between the inner and outer cylinders. It has been verified that, when the peripheral velocity is further accelerated, the mode of the flow on the torus, known as a cell, transitions to multi-layered Taylor vortex flow and wave Taylor vortex flow. Fig.1 shows a diagram of the Taylor vortex flow.

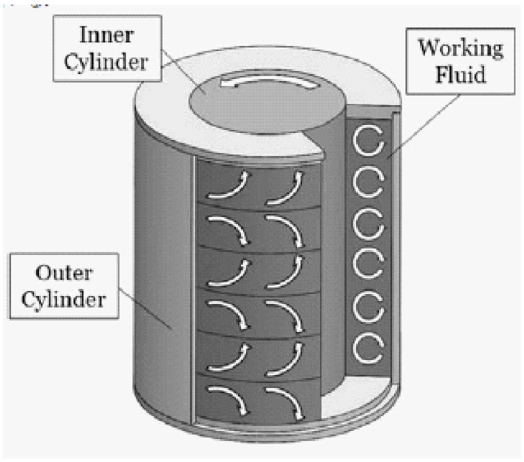


Fig1. Taylor-vortex flow (assuming both ends are fixed)

The Taylor vortex flow can be broadly classified into two types, the normal mode and the anomalous mode. When the top and bottom ends of the double cylinder are fixed, the normal mode indicates a mode wherein there is a flow from the outer cylinder to the inner cylinder at both the top and bottom end surfaces. In addition, the anomalous mode indicates a mode with flow from the inner cylinder to the outer cylinder at both, or one of, the top and bottom end surfaces. In this study, we use deep learning [3, 4] to create a program that can automatically identify the result of simulations that determine whether the vortex is in the normal mode or anomalous mode. Fig.2 shows examples of images from the simulation results of the Taylor vortex flow. For the program created in this study, the machine underwent learning using the method known as supervised learning, using the images from the simulation results of the Taylor vortex flow as the teacher data. Following this, we created a program that can automatically identify whether a flow is in the normal mode and the anomalous mode based on the images from simulation results.

Machine learning is the process of iteratively learning from image data in order to identify the patterns hidden in the same; among these, one of the learning processes that more accurately approximates the human brain is the technology known as deep learning. It uses a multi-layered neural network modeled after the neural circuits of the human brain. Here, a neural network indicates a combination of mathematical models of the neurons in the human cranial nervous system. The concept is based on the mechanism of the human brain, and it is a mathematical model created for computers to reproduce some of the characteristics of brain function.

The human brain consists of numerous nerve cells (neurons), and information processing is performed through the complex binding of these neurons. A neural network is a simplified reproduction of this mechanism of the information processing network of the brain in the form of a computer. In Japanese, a neural network would be translated as the network of nerve cells.

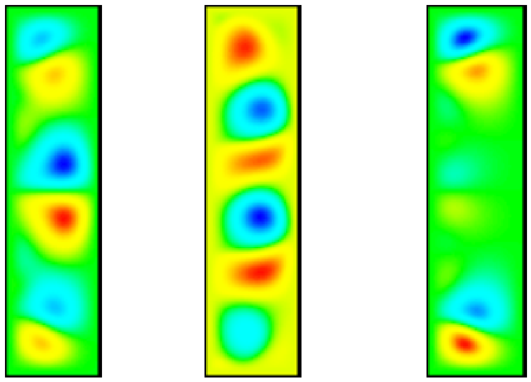


Fig2. Simulation results of Taylor vortex flow

ANALYSIS METHOD

Governing Equation

The continuous equation of incompressible fluid and the nonconservative system Navier-Stokes equation that are the governing equations in physical space used for collecting images in this study are thus:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla) u = -\nabla p + \frac{1}{Re} \nabla^2 u \qquad (1)$$

$$\nabla \cdot u = 0 \qquad (2)$$

Here, the velocity vector of fluid is u, pressure is p, and the Reynolds number is Re.

Analysis Conditions

The calculation is conducted as an analysis of the fixed end in 2D. The numerical data items were taken from existing studies. In this study, r lattice number = 41, time interval = 0.012 and time step = 250,000, and they were fixed at these numbers. The aspect ratio, z lattice number and Reynolds number were arbitrarily set in the ranges between 3.0 and 7.0, 122 and 241, and 1,000 and 5,000, respectively, when performing analysis. At this time, the z lattice number was determined by r lattice number × aspect ratio.

RESEARC FLOW

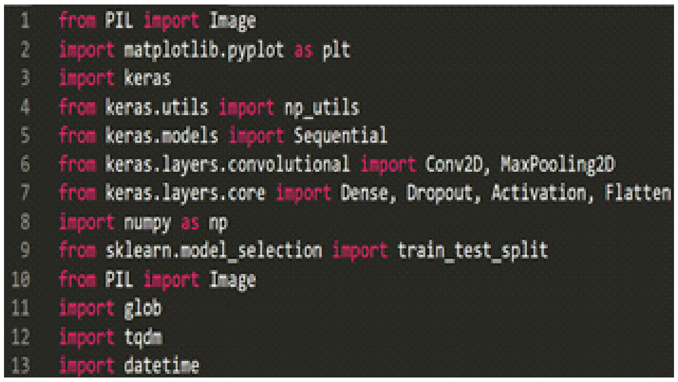
Image Collection

As the program created for this experiment used the method where machine learns through supervised learning in which teacher data are the images of Taylor vortex flow simulation results, these images to be used as the teacher data were collected first. As it was necessary to distinguish the collected images according to the conditions to be learned by the machine, this study sorted the collected images into seven folders under conditions wherein they are first distinguished as being in normal mode or anomalous mode, and then according to the number of vortexes.

Preparation for program creation

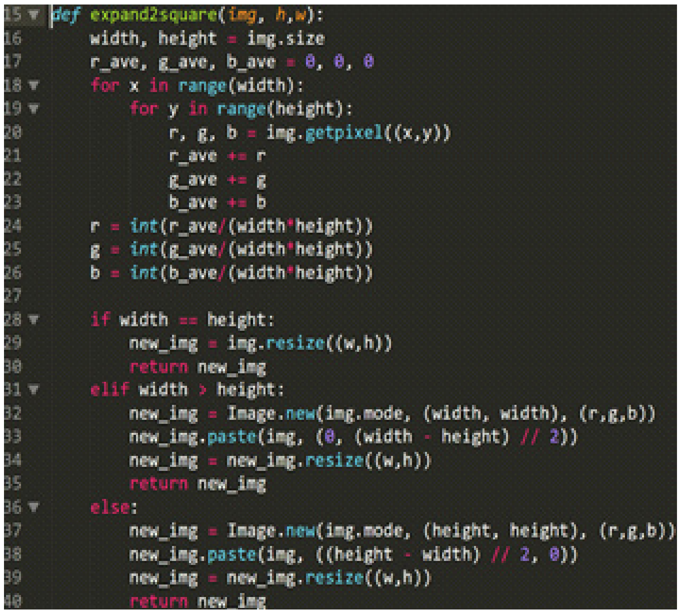
Next, it was necessary to install the tools used to create the program. In this study, the program created using the editor called Sublime was executed. Sublime was used because it can handle vast amounts of data, as the image data was too large. Sublime can be installed by visiting its official website.

The tools necessary for creating the program were imported prior to creation. Fig.3 shows the imported tools. Tools can also be imported during the programming.



```
from PIL import Image
import matplotlib.pyplot as plt
import keras
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Dense, Dropout, Activation, Flatten
import numpy as np
from sklearn.model_selection import train_test_split
from PIL import Image
import glob
import tqdm
import datetime
```

Fig3. Tool contents



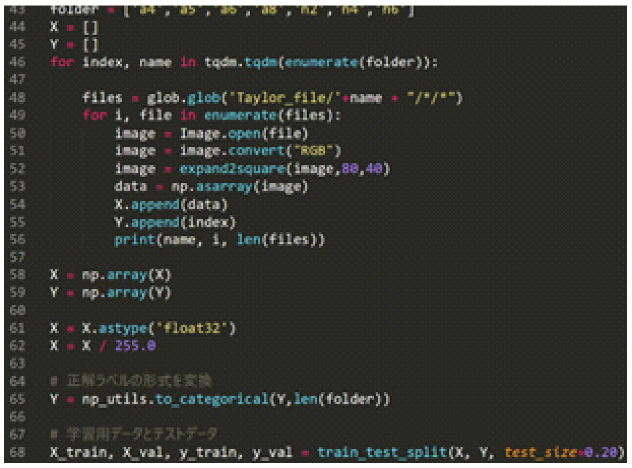
```
def expand2square(img, h,w):
    width, height = img.size
    r_ave, g_ave, b_ave = 0, 0, 0
    for x in range(width):
        for y in range(height):
            r, g, b = img.getpixel((x,y))
            r_ave += r
            g_ave += g
            b_ave += b
    r = int(r_ave/(width*height))
    g = int(g_ave/(width*height))
    b = int(b_ave/(width*height))

    if width == height:
        new_img = img.resize((w,h))
        return new_img
    elif width > height:
        new_img = Image.new(img.mode, (width, width), (r,g,b))
        new_img.paste(img, (0, (width - height) // 2))
        new_img = new_img.resize((w,h))
        return new_img
    else:
        new_img = Image.new(img.mode, (height, height), (r,g,b))
        new_img.paste(img, ((height - width) // 2, 0))
        new_img = new_img.resize((w,h))
        return new_img
```

Fig4. Resize of images

Creation Of Machine Learning Program

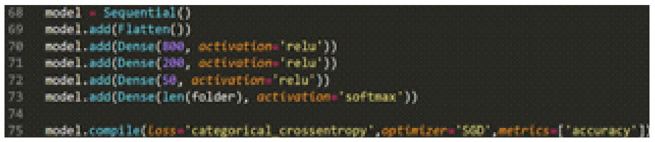
Next, as the sizes of the teacher data images were not entirely similar, their sizes were equalized to facilitate learning by the machine. As the aspect ratio of the images was 2:1, and therefore vertically long, it was difficult to equalize the size of every image. Thus, a method was used wherein each image was pasted onto a square image, upon which the required blank parts were filled with one color. This led to uniform image sizes. Fig. 4 shows how to resize the images.

Next, the image data to be learnt was collected. If every image is learnt, the amount of data will become too large to process, causing overfitting and making it impossible to evaluate the validity of the program. Thus, approx. 1,000 a4–n6 images were selected for this purpose. Moreover, labels between 0 and 6 were attached to the a4–n6 images. Teacher images were input to X=[], and labels between 0 and 6, corresponding to the images, were input to Y=[]. Fig.5 shows the program for preparing the data.



```
folder = ['a4','a5','a6','a8','n2','n4','n6']
X = []
Y = []
for index, name in tqdm.tqdm(enumerate(folder)):

    files = glob.glob('Taylor_file/'+name + "/*/*")
    for i, file in enumerate(files):
        image = Image.open(file)
        image = image.convert("RGB")
        image = expand2square(image,80,40)
        data = np.asarray(image)
        X.append(data)
        Y.append(index)
        print(name, i, len(files))

X = np.array(X)
Y = np.array(Y)

X = X.astype('float32')
X = X / 255.0

# 正解ラベルの形式を変換
Y = np_utils.to_categorical(Y,len(folder))

# 学習用データとテストデータ
X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.20)
```

Fig5. Program for preparing the data.



```
model = Sequential()
model.add(Flatten())
model.add(Dense(800, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(len(folder), activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='SGD',metrics=['accuracy'])
```

Fig6. Epoch contents

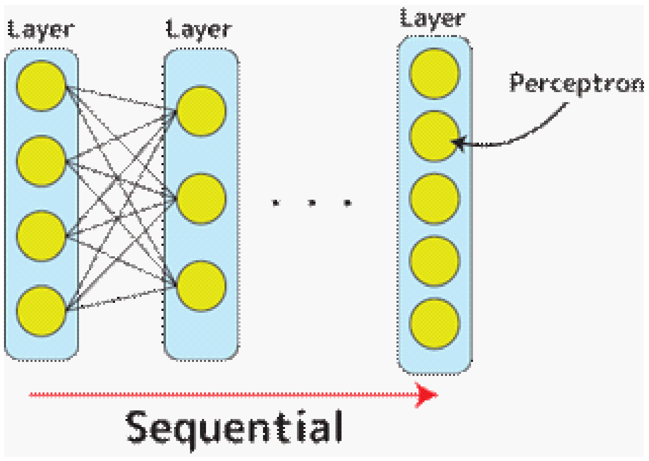


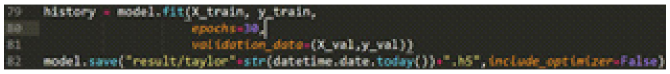
Fig7. Sequential model

Fig.6 shows the process of building the neural network layer discussed in the introduction, after preparing the data. When performing deep learning, there are three or more layers in principle, and the possibility for learning improves when there are more layers. However, that does not guarantee a successful result. Therefore, in this study, complex images were used to first verify whether the created program is running properly, and the number of layers was set to three.

After preparing the data, the program for learning must be built. While there are many ways to build programs, for this study, simple programs were preferred rather than complex programs, as the images used were simple.

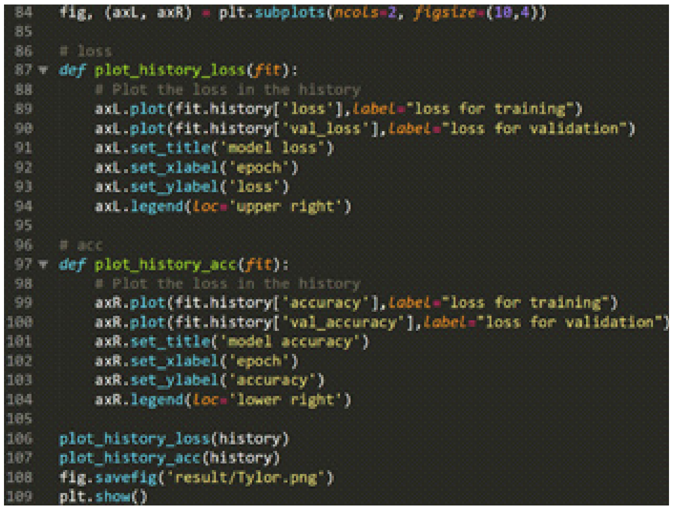
A neural network is, essentially, a complex structure wherein the elements (i.e., neurons) form a web, and it is difficult to reproduce it in a program. Thus, this study used model = Sequential(), which is a type of simplified model. As its name indicates, model = Sequential() is joined in one, line-like sequence. Each node of the current layer is connected to every node of the previous layer with allows. The origin of these connections is a mass of neurons (Perceptron) called the layer. The neurons inside a layer send signal to next layer, and the Sequential model continues this sequentially until the last layer. This model allows endless addition of layers. However, as the images handled in this study were simple, as mentioned above, a program with only three layers were built. Fig.7 shows the diagram of the Sequential model. First, enter model = Sequential to define the usage of the Sequential model. Next, as add() allows for the addition of layers, enter model.add(Dense(800, activation='relu')) to define the first layer. Dense indicates a fully connected neural network, and this line alone represents one neural network. As a fully connected neural network cannot handle spatial information, it is impossible for it to learn 3-D images. However, as this study only used 2-D images, Dense could be used. Activation indicates an activation function. The output dimensionality of this layer was 800, and the activation function was relu. Similarly, the output dimensionalities of the 2nd and the 3rd layers were set to 200 and 50, respectively, and relu was also designated as the activation function. The model.add(Dense(len(folder), activation='softmax')) command was used for the last layer, with the dimensionality at 7 (number of files of a4–n6 (labels from 0 to 6)), and the activation function was activation. This was the building process of the network using the Sequential model for this study.

Once the program is ready up to this point, the machine can start learning. "How many times one iteration of training data is learnt repeatedly" is called the epoch. When the number of parameters is large, they cannot be learnt properly unless the training data is learnt in an iterative manner. However, when this is overdone, it can cause overfitting. Overfitting indicates a type of learning where the program does not respond well to any samples other than the training data when it is trained to fit perfectly with each learning data. Here, an ideal epoch is one that offers both good training precision and prediction precision without causing overfitting. For this study, the epoch at the moment when the degree of accuracy started to converge was judged to be optimal, as it allowed verification of the degree of accuracy during the learning process. Fig.8 shows the epoch program.



```
history = model.fit(X_train, y_train,
                    epochs=30,
                    validation_data=(X_val,y_val))
model.save("result/taylor"+str(datetime.date.today())+".h5",include_optimizer=False)
```

Fig8. epoch program



```
fig, (axL, axR) = plt.subplots(ncols=2, figsize=(10,4))

# loss
def plot_history_loss(fit):
    # Plot the loss in the history
    axL.plot(fit.history['loss'],label="loss for training")
    axL.plot(fit.history['val_loss'],label="loss for validation")
    axL.set_title('model loss')
    axL.set_xlabel('epoch')
    axL.set_ylabel('loss')
    axL.legend(loc='upper right')

# acc
def plot_history_acc(fit):
    # Plot the loss in the history
    axR.plot(fit.history['accuracy'],label="loss for training")
    axR.plot(fit.history['val_accuracy'],label="loss for validation")
    axR.set_title('model accuracy')
    axR.set_xlabel('epoch')
    axR.set_ylabel('accuracy')
    axR.legend(loc='lower right')

plot_history_loss(history)
plot_history_acc(history)
fig.savefig('result/Tylor.png')
plt.show()
```

Fig9. Program to show a loss function

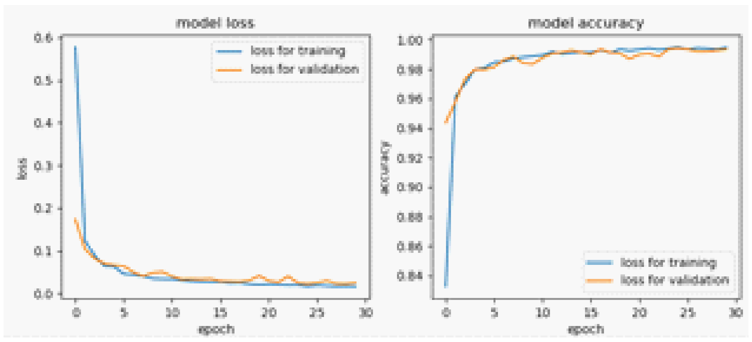
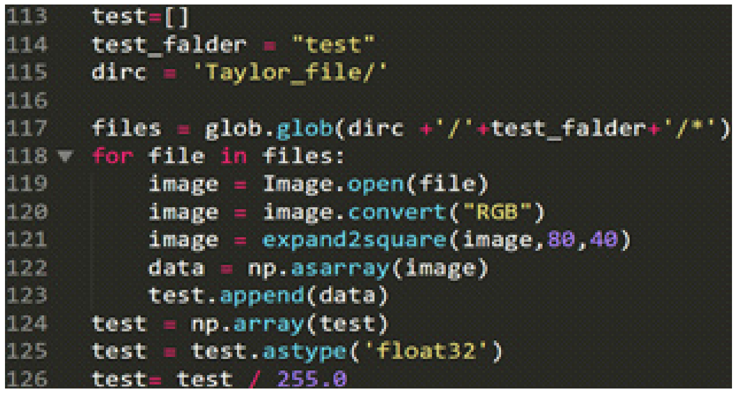


Fig10. Graph of the loss function

After learning is completed, display the loss function graph to verify the accuracy.

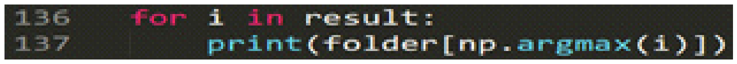
To control the output of a neural network, it is necessary to measure how far the output is from the expected result. The loss function of the network enables this measurement. Fig. 9 shows the program for displaying the loss function, and Fig.10 shows its graph.

In Fig.10, the vertical axis is the accuracy and the horizontal axis is the epoch. Loss for training is the graph that represents the transition of the learning data and loss for validation is the graph that represents the transition of the test data. Generally, the accuracy approaches 1.00 each time the epoch is increased; however, in this study, it neared 1.00 when the epoch was still small. This is considered to be due to the small number of factors determining the mode of Taylor vortex flow and the simplicity of the images making the identification of the mode relatively easy. Moreover, as the program structure was simple, the calculation time was very short.



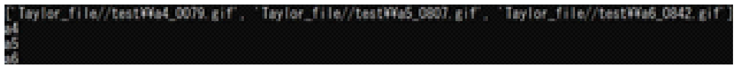
```
test=[]
test_falder = "test"
dirc = 'Taylor_file/'

files = glob.glob(dirc +'/'+test_falder+'/*')
for file in files:
    image = Image.open(file)
    image = image.convert("RGB")
    image = expand2square(image,80,40)
    data = np.asarray(image)
    test.append(data)
test = np.array(test)
test = test.astype('float32')
test= test / 255.0
```

Fig11. Program contents



```
for i in result:
    print(folder[np.argmax(i)])
```

Fig12. Result of Program



```
['Taylor_file//test¥¥a4_0079.gif', 'Taylor_file//test¥¥a5_0807.gif', 'Taylor_file//test¥¥a6_0842.gif']
a4
a5
a6
```

Fig13. Analyzed result

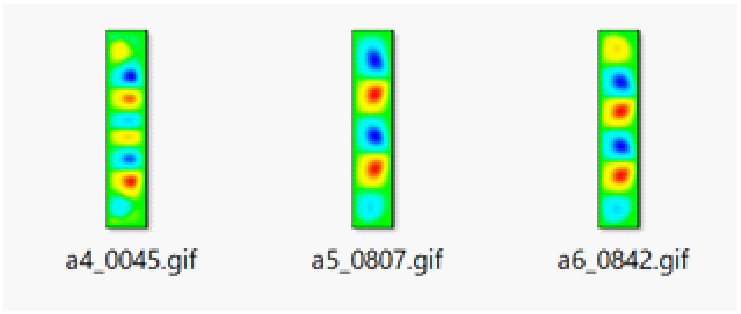


Fig14. Used image

Creation of Program For Identifying Mode and Its Results

After the learning by the machine, it is necessary to verify whether it learnt as desired. For this purpose, randomly selected images were gathered in one folder and the machine was made to identify the same.

Fig.11 shows the program for loading the images in the folder prepared for the purpose of identification.

Once loading is complete, the identification result is output. Fig.12 shows the created program, Fig.13 shows the result, and Fig.14 shows the images used for the test. As the output of the result was the print(folder[np.argmax(i)]) command, it was output as the folder name. While only three images were used on this occasion, the program was still judged to be successful, as it correctly identified all the modes as well as the number of vortexes.

By comparing Fig.13 and Fig.14, one can recognize that the mode development process was correctly identified.

CONCLUSION

In this study, approx. 12,000 of 300,000 collected images were selected as teacher data, and with the created program, it became possible to automatically distinguish between the normal mode and the anomalous mode in terms of these images. On this occasion, instead of using the entirety of the 1,000 images in each folder, 10 images were selected from every 100. In the future, it will be necessary to change the selection conditions and continue with the identification of the mode development process.

ACKNOWLEDGEMENT

We would like to show our gratitude to members in our laboratory for sharing their pearls of wisdom with us during the course of this research.

REFERENCES

1. G. I.Taylor, “Stability of a viscous liquid contained between two rotating cylinders”, Phil. Trans. Roy. Soc. A 223, pp. 289-343, 1923
2. P. R. Fenstermacher, H. L. Swinney, J. P. Gollub, “Dynamical instabilities and the transition to chaotic Taylor vortex flow “, Journal of Fluid Mechanics, Vol. 94, pp. 103-128, 1979
3. S. Lee, D, You, “Data-driven prediction of unsteady flow over a circular cylinder using deep learning”, Journal of Fluid Mechanics, Vol. 879, pp. 217-254, 2019
4. B. Liu, J. Tang, H. Huang, X. Lu, “Deep learning methods for super-resolution reconstruction of turbulent flows”, Phys. Fluids, Vol. 32, 025105, 2020

Citation: *Hiroyuki Furukawa, Kensuke Matsuoka. Study on Identification of Taylor Vortex Development Process Through Deep Learning. Int J of Innovative Studies in Sciences and Engineering Technology. 2021; 7(3): 1-7. DOI: https://doi.org/10.20431/2455-4863.070301.*