Turkish Journal of Computer and Mathematics Education Vol.12 No. 11 (2021), 2507- 2521
Research Article

A modern approach to building a data science framework delivery pipeline using DevOps practices

Surabhi Saxena[1], Shashi Kant Gupta[2], Dr. S. Poongodi[3], Prabhdeep Singh[4]

[1]Assistant Professor, Department of BCA, Koneru Lakshmaiah Education Foundation, Guntur, AP, India
[2]Researcher, CS&E Department, Integral University, Lucknow, UP, India
[3]Professor, ECE Department, CMR Engineering College, Hyderabad, Telangana, India
[4]Assistant Professor, School of Computer Applications, BBD University, Lucknow, UP, India
Corresponding Author: Shashi Kant Gupta

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract:
For data science, the potential for commercialization is significant. Recent work indicates that the philosophy of DevOps is a perfect way of addressing the challenges of software development. And both data science and software engineering from a product perspective need to provide customers with digital services. The feasibility of using DevOps in data science must therefore be studied. This article outlines a method for creating a framework for the use of DevOps methods within a data science program. I used four pipeline practices: version control, platform server, containerization and continuous integration and delivery. DevOps is, however, not a theory specifically designed for data science. This means that the DevOps methods which are currently available cannot solve all the problems of production data science. I have used DevOps' practice to address such a problem with a data science practice. I've learned and engaged in the process of transfer learning. This paper describes a parameter-based method to move parameters from one dataset to another. I have examined the impact of model transmission learning on a new dataset. The result demonstrates the adaptation of the learning process to modify the model without re-training from scratch when the dataset changes but is identical to the old one. This is a safe idea to freeze the convolutive layer if the current model is to reach the same degree of efficiency as the previous one. When the new model will achieve higher than the original model, the loading of weights is better choice. In conclude, the current methods of DevOps do not need to be restricted if we use DevOps in data science.

Keywords: Data science, DevOps, Convolutional neural network, Transfer learning

1. Introduction

In recent years, the research field of data science has become so popular that no one can ignore it[1]. Data science is so hot for many reasons. First of all, the rapid development of the Internet and digitization have caused the world to generate massive amounts of data all the time. Such a scale of data creates huge business needs for data science. Moreover, the rapid development of machine learning in recent years has provided powerful nutrients for data science. However, with the development of data science, some problems exposed that had not been noticed by re- searchers before [2]. The data science projects are more difficult to be applied to production than traditional software engineering projects. The solution to similar problems in software engineering is DevOps. DevOps[22] integrates development and operation to work on a closed-loop pipeline. It is more difficult to maintain a data science model system in production compared with software systems.

In the field of software engineering, teams responsible for different tasks collaborate to develop and maintain the product. The division brings a prob- lem: the natural goals of each team are different: the developer team pursues the timely response to customer requirements, develops new functions, but the priority task of the operation team is to ensure the stability of the system. Which build a "wall" between different teams. The methodology of DevOps is to break this "wall", integrates development and operation team, and makes them share the same goal that responding to customer needs in time while maintaining the quality of the products. This is the goal of data science too. Which means DevOps can be a solution to deal with the the problems of data science in production.

However, the currently available practices are not sufficient to handle all the problems of data science in production [23] Because there is much difference between software engineering and data science.

Besides the set of DevOps practices, we need involving more practices to deal with the unique problems of data science in production. Transfer learning is a promising data science practice which has been proved[3] that is very useful to leverage a pre-trained model to fit new data.

2. Background

A. Data Science

Data science is a basic theory about extracting information and knowledge from data [4]. Data science uses statistics, data analysis, machine learning and related methods to understand and analyze actual phenomena through data [5]. As Figure 1 shows that data science covers a very wide range of research, including data mining, data analysis, and data science. Each branch contains a large number of directional algorithms and theories.

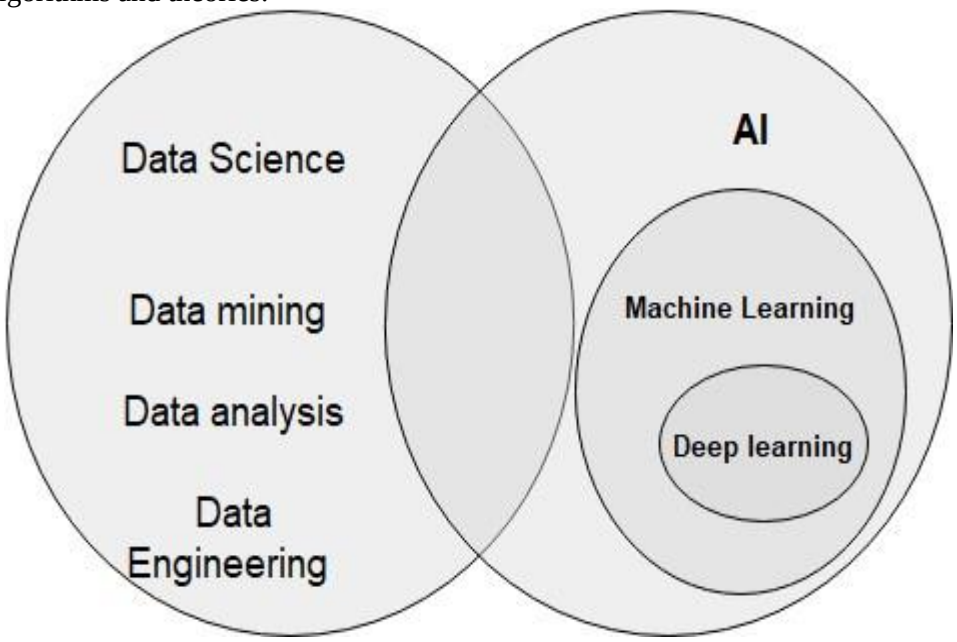


Figure 1 the scope of the data science

In this era, there are many shreds of evidence that can prove that data science has a promising future. With the digital movement progressing, the scale of available data has grown dramatically. On one side, computer hardware is becoming more and more powerful, while on the other side, hardware costs are steadily decreasing. Coupled with the popularity of the Internet and break- through in algorithm research, we can conduct a more in-depth analysis of data than before [4]. **1. Conventional Neural Networks**

The models used in this project were generated depended on the convolutional neural network (CNN) theory. In order to understand the principles of CNN, the reader first needs to understand some basic concepts of Artificial neural networks (ANN). Because CNN theory is developed based on ANN. An ANN is made up of a large number of neurons and is divided into many layers. Figure 2 shows the basic structure of ANN. An ANN includes an input layer, some hidden layers, and an output layer. The input neurons construct the first layer output whatever input they are fed. But the rest of the neurons work in another way. The data flow of that kind of neurons is shown in Figure 3. This kind of neuron is also called the linear threshold unit (LTU).

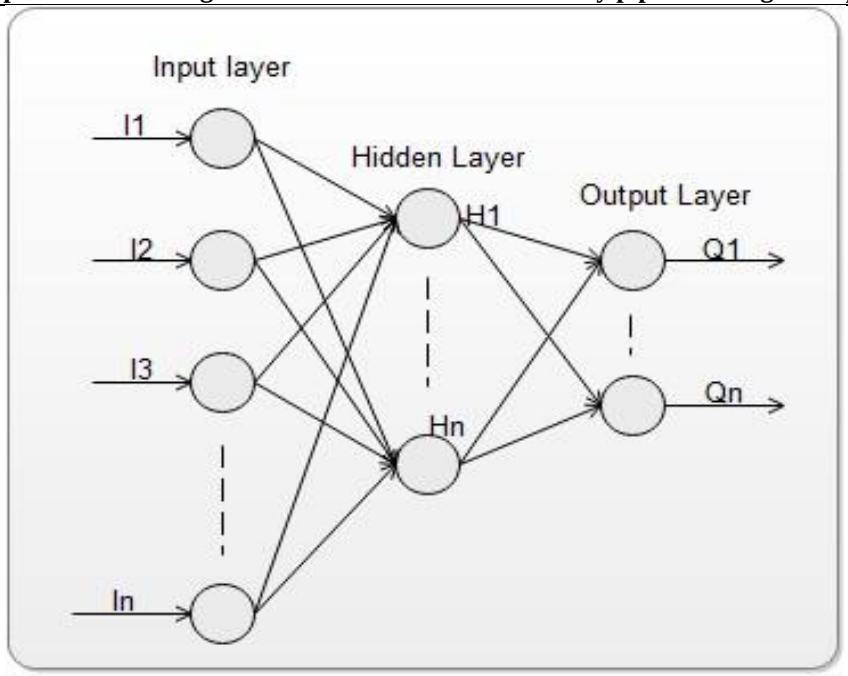


Figure 2: Artificial neural network model example

Inputs of an LTU are numbers, and each input connection is associated with a weight. Then a weighted sum of its inputs can be computed and applies a function to that sum. Then the error can be calculated by comparing the desired value with the output value. A simple machine learning model works like a single LTU, and can only understand simple linear relationship exists in the dataset. The benefit of the network structure used by ANN for deep learning is that the model can mine the complex relationships that exist in the data set. However, ANN does not perform well when faced with computer vision problems. For example, if we feed each pixel of an image as an input to ANN, these pixels flow through each layer of the network and processed by each LTU with the approach described above. Finally, a prediction is obtained at the output layer. The shape information of the picture is lost, and the picture information changes from two-dimensional data to a one-dimensional pixel sequence.

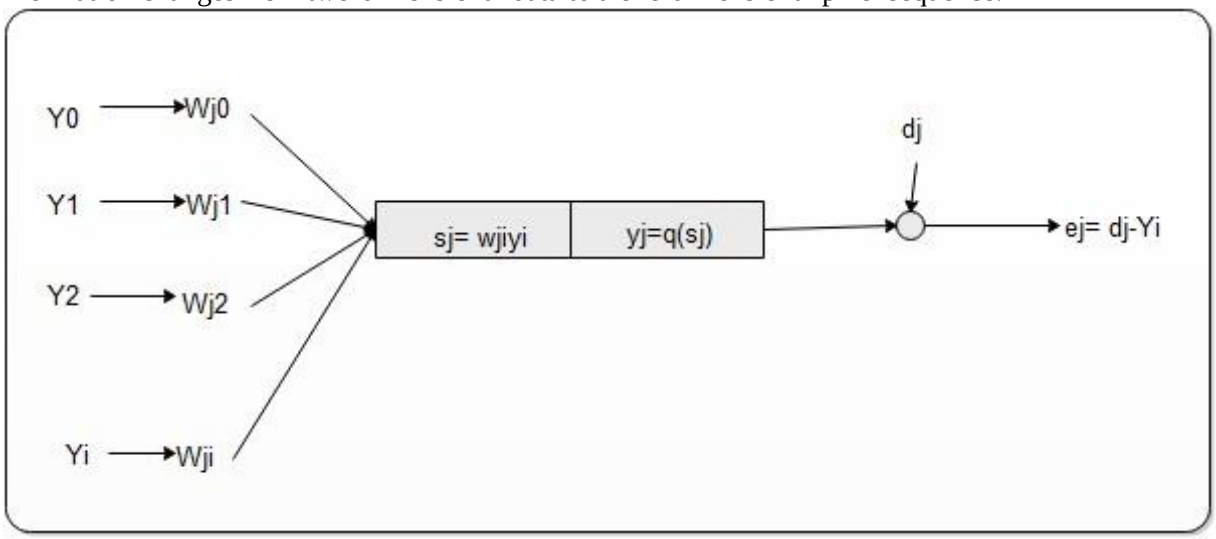


Figure 3: Neuron j Signal Flow

2. Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to another similar task [5]. In other words, the basic principle of transfer learning is to reuse the models. Learning time can be reduced by freezing some layers in a network [6]. Yosinski and his team [3] experimented on the ability to transfer features from a base dataset to a target dataset. Their experiments showed that: transferring the features gained from a basic model to a new model can produce a boost to generalization performance. Today, transfer learning methods are applying in many fields, most notably in data mining, machine learning and

applications of machine learning and data mining [7]. This project is focused on the application of transfer learning on CNN.

B. DevOps

The DevOps philosophy emphasizes continuous collaboration between the developers and the operations, as well as everyone in between, such as software testers to quickly respond to customer requirements [26]. It looks very relevant to data science. Data scientists also facing a changing rapidly market. In this case, data scientists have to dig out the volume of data as soon as possible. Which means the software engineering and data science has a common problem to be resolved: How to complete work faster while ensuring product quality. DevOps theory seems to be a very suitable choice for data scientists because software engineers solve this problem well by applying this theory. By extending DevOps theory, data scientists can also solve the problems they face in production. Some researchers call this kind of extension of DevOps as DataOps [8]. DataOps is not a brand-new concept and has been proposed for some years, but there is still no universally accepted definition. **1. Tool chain**

DevOps practices rely heavily on tools of various kinds, including tools for container management, continuous integration, orchestration, monitoring, deployment, and testing [9][10]:

- Coding: creation and analysis of code, tools for handling the source code and combining code.
- Design – resources for continuous integration, building status.
- Testing – continuous tools for monitoring that provide quick and prompt business risks feedback.
- Packaging – server, pre-deployment device staging.
- Releasing – management of transition, authorisation of releases, automation of releases.
- Setup – setup and management of the network, as tools for application infrastructure.
- Tracking – performance control software, end-user experience.

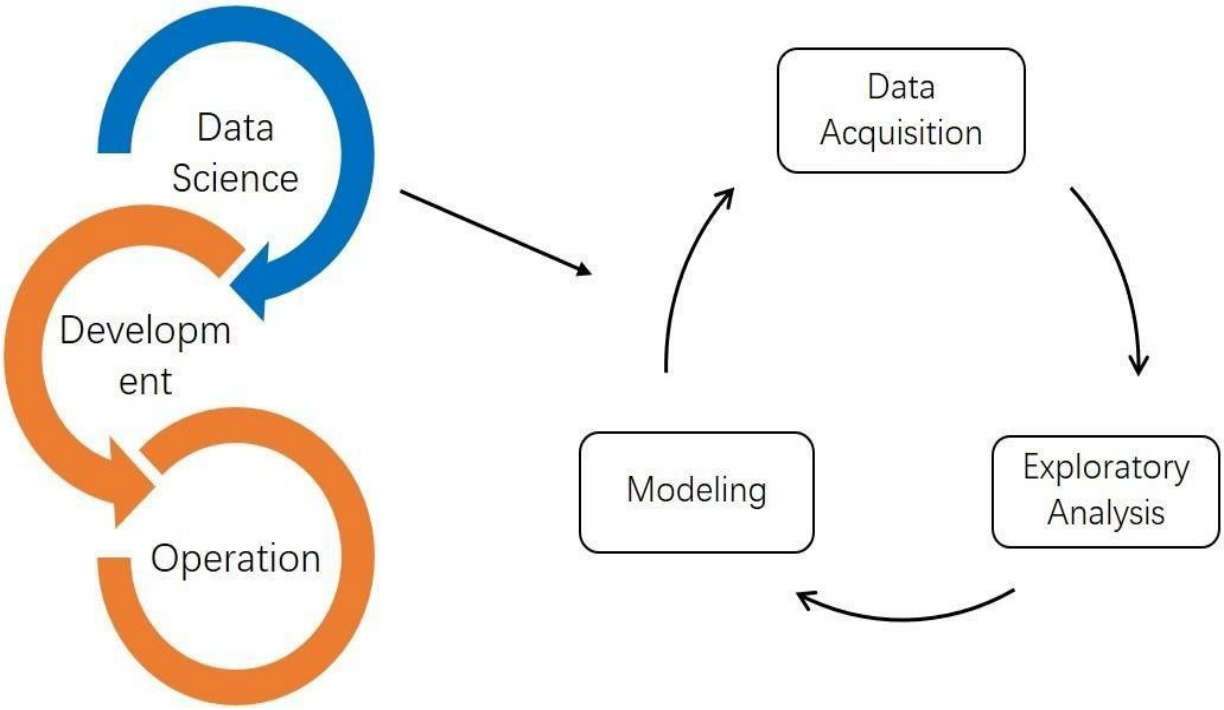
2. Limitations of DevOps

We can apply some best DevOps practices to data science. Software engineering and data science have certain similarities when we thinking of the model as code. If the DevOps can benefit software engineering to face the production environment, it can benefit data science as well. The DevOps theory is originally design for software engineering. However, there is much difference between data science and software engineering. We have to treat the data science model as a component of the system when we talking the data science from a production perspective. Because the model cannot serve the custom just by itself, it must work as a function of a system. The life cycle of developing a system contains a data science model is showed in Figure 4. DevOps practices can cover the most part of the cycle but cannot cover the inner cycle: the life cycle of data science. The general process of the data science development model is data analysis, experiment, and modeling. In order to get a qualified model, this process needs to be repeated many times. This process is very different from the software development process. Which means we need more practices besides the practices of DevOps[25].

Figure 4: Life cycle of the system and the inner cycle of data science

3. Pipeline Development for System Design

To develop a delivery pipeline for a machine learning system applying DevOps approach, I applied four practices of DevOps on a machine learning system, these practices are version control, model server, containerization, and CI/CD.



A. Version Control

From a traditional software engineering perspective, version control refers to the managing of the source code. However, from the field of data science perspective, the version of data is also needed to be managing. For machine learning, an important branch of data science, version control of data is especially important. Because there is a common problem in the field of machine learning: historical models are often very difficult to reproduce. Even if the data scientist uses the same algorithms and datasets, it is still difficult to obtain the same performance as the model. Even for the creators of the model, it is very difficult to achieve the same performance. There are many reasons for this problem. One meaningful reason is that data versions are not well managed. Data pre-processing is an essential step in the pipeline of data science. To get an excellent model, data scientists often need to repeat data pre-processing and model training. This means that every time the data is pre-processed, a new version of the data is obtained. Data itself directly affects the performance of the model; it is necessary to control the version of the data.

Meanwhile, it also means that we need to manage the correspondence between the version of the data and the version of the source code. Another purpose of managing data version is to solve the problem of cloud storage of data. Enterprise data is generally stored in local data warehouses or cloud storage service platforms. If we directly use traditional version control tools to manage the data version, we can only upload the data to the traditional source code hosting service platform. In other words, the data is stored in both the enterprise's own data warehouse and the source code hosting platform. In this way, storage resources would be wasted. Based on the above analysis, the version control of this project includes source code version control and data version control. At the same time, this project also supports for cloud storage, code and data are hosted on different cloud storage platforms. The data version control tool generates a metadata file of the dataset, which is managed by the traditional version control tool together with the source code. When the source code version is switched, the corresponding data is switched through the version of the metadata file.

B. Model Server

There are two modelling approaches in data science as shown in Figure 5. One is that the data scientist who manually develops the model based on his or her relevant expertise, and then analyses the data through the model. The analysis result is generally a report used to explain historical data, to provide support for business decisions. The model itself is not a product that needs to be delivered to the customer, the product is the result of the model-based analysis. The parameters of the model are all set manually by the data scientist. The establishment of this kind of modelling approach requires both the data scientist's excellent programming skills and the data scientist's in-depth experience in the field of the data it analyses. Another modelling approach does not require data scientists to be so versatile. The

task of modelling is mainly done by the data itself. The data scientist only needs to build a basic model according to the results of data analysis and related algorithms, and then let the model fit the data. Through continuous learning of the data, the performance of the model finally reaches a reasonable level. In this process, the parameters of the model are continuously optimized by the data according to the algorithm. The model obtained in this way is generally used to predict the future.

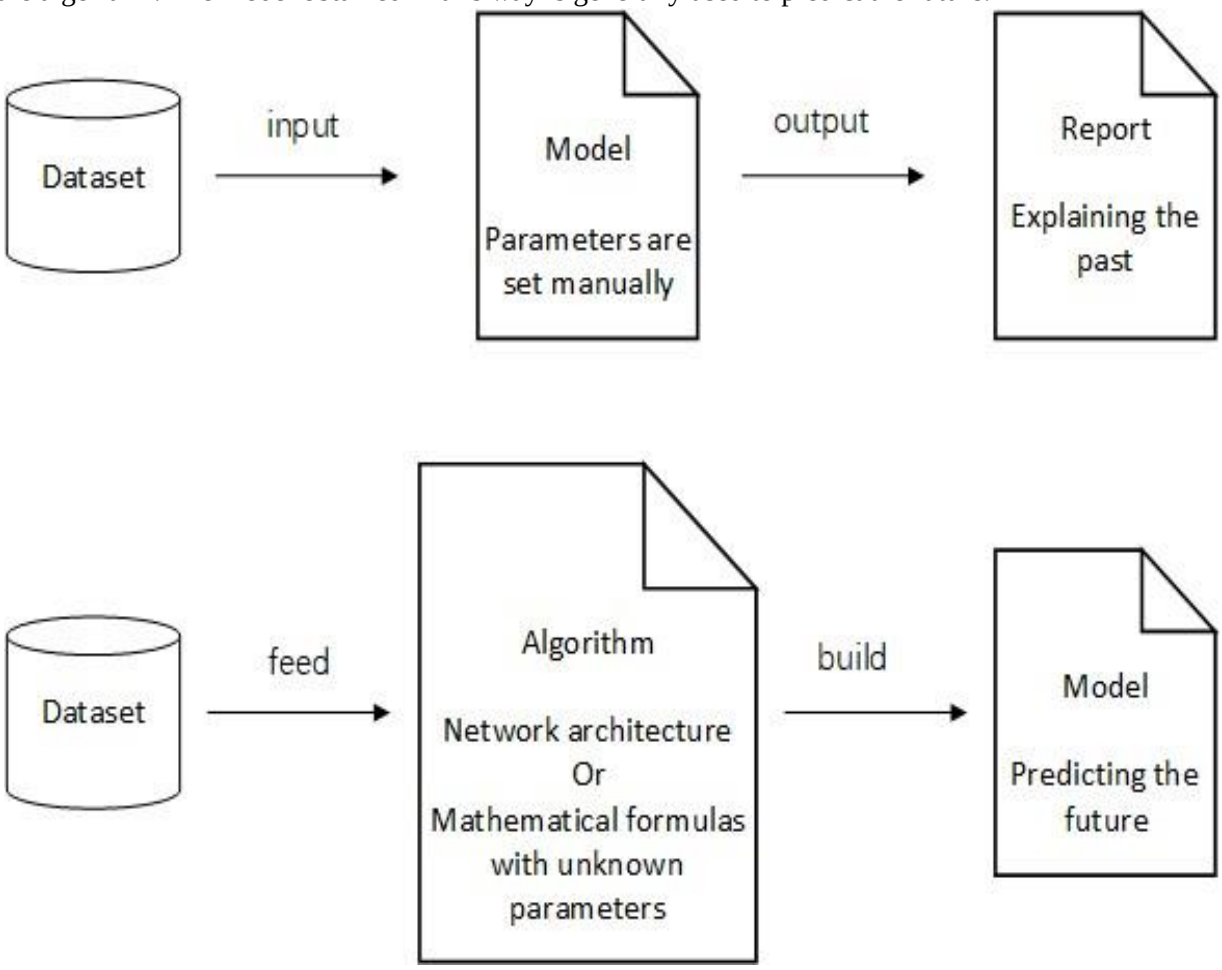


Figure 5: Two modelling approaches

This work is based on the second modelling approach. Because the model obtained in the second approach servers users in the production environment, and this project is focused on the operation of the model in production. But models produced by data scientists cannot be used directly by users. To use the model file directly, the users have to know how to program. We can't force users to do anything. In a production environment, data scientists as part of a team need to work with system developers.

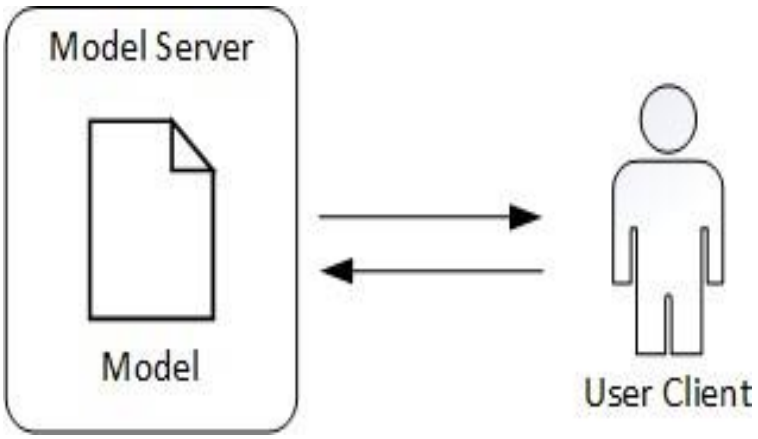


Figure 6: The architecture of model server

Data scientists produce models, and system developers integrate the models into the company's systems. However, the language used by system developers may be different from the language used by data

scientists. It is not a feasible solution for data scientists to directly throw model files to system developers. In this project, after I got the model, I packaged it into a model service and provided a graphical interface so that anyone can use the model very conveniently. In conclusion, the model server provides easy-to-use API for end-users or software developers. It is also useful for model A/B testing: Swapping different models out depending on the inference. As Figure 6 shows that the model is loaded in a server which allows API requests and infers the model predicts.

Table 1: The statistics of the datasets

Purpose	Split	Labels	Examples
Basic model training	Train	Cat	4000
		Dog	4000
	Validation	Cat	1000
		Gog	1000
Transfer learning	Train	Cat	4000
		Gog	4000
	Validation	Cat	1000
		Gog	1000
Total			20000

4. Implementation A. Dataset

The data used in this degree project consists of images of cats and dogs. Which I got from Kaggle related to a very famous image classification problem: Cat vs. Gog. The images of the data set look like Figure 7. As shown in Table1 that the dataset was split into two parts. One is used for training a basic convolution neural network. Another one is used to experimenting the transfer learning.

Figure 7: Data examples

B. CNN

Table 2: Model summary

Layer	Output Size	Output Channel	Kernel Size	Parameters
Convolution	150×150	16	3	448
Max pool	75×75	16	–	0
Convolution	75×75	32	3	4640
Max pool	37×37	32	–	0
Convolution	37×37	64	3	18496
Max pool	18×18	64	–	0
Convolution	18×18	128	3	73856
Max pool	9×9	128	–	0
Flatten	1	10368	–	0
Fully connected	1	256	–	2,654,464
Fully connected	1	512	–	131,584
Fully connected	1	1	–	513
Total parameters: 2,884,001				

To achieve the desired model to experiment transfer learning on it, I trained a convolutional neural network as the model of the project. Compared to the design of the model itself, this project is more concerned about the performance of transfer learning in model fine-tuning. Therefore, I did not choose those famous but complex CNN-type models. Under the premise of ensuring model performance, I adopted a network structure that is as simple as possible to reduce experimental interference. As shown in Table 2, the model includes three convolution modules and a fully connected module. Each convolution module contains a convolution layer and a pooling layer. The fully connected module consists of a flattening layer and two fully connected layers. To reduce computation during back propagation, all convolution layers and the first fully connected layer use the ReLU activation function. By using the sigmoid activation function in the last fully connected layer, the model can output class probabilities based on binary classification.

After getting a basic model I performed three experiments to answer the second question. In all these three experiments I compared the training accuracy and validation Accuracy, and training loss and validation loss. The dataset used in all experiments is the transfer learning dataset.

Experiment 1:

To evaluate transfer learning I again used the CNN architecture described in Section IV (A). After applied rescaling, and resized the images of the transfer learning dataset into the required dimensions, I trained the network on the pre-processed images. I retrained the model from scratch but using a completely different dataset. According to it, the performance of the model on the training and test sets reached an ideal level after 50 epochs.

Table 3: The parameters of Experiment 1: Training from scratch

Parameters	**Quantity**
Total	2,884,001
Trainable	2,884,001
Non-trainable	0

Experiment 2:

In this experiment, I first loaded the network weights that were trained in Section IV(B). Then I retrained the model on the new dataset. During the training process, I didn't freeze any layer, as same as the approach taken in the experiment where Yosinski got the best results. According to it, the performance of the model on the training and test sets reached an ideal level after 20 epochs.

Table 4: The parameters of Experiment 2: training with trained weights

Parameters	**Quantity**
Total	2,872,001
Trainable	2,872,001
Non-trainable	0

Experiment 3:

As same as in the previous experiment, I first loaded the network weights that were trained in Section IV(B). Then I retrained the model on the new dataset. However, different from the previous experiment, all the weights of the convolution layer were frozen. During the training process, only the weights of the fully connected module were retrained. According to it, the performance of the model on the training and test sets reached an ideal level after 20 epochs.

Table 5: The parameters of Experiment 3: training with frozen convolution layers

Parameters	**Quantity**
Total	2,884,001
Trainable	2,786,561
Non-trainable	97,440

I used multiple Python libraries in this project. The Keras [11] library and TensorFlow [12] library was used for implementing the models and experiments on the transfer learning [13]. Matplotlib [14] was used to plot the graph and display images in the training and validation data..

5. Results

This chapter presents the results of each experiment. And the results are analyzed by comparison.

A. Results of each experiment

Table 6: The training accuracy and loss the trained after 30 epochs between the basic model and the model of Experiment 1

Models	Epoch	Training accuracy	Training Loss

Training basic model	30	0.7928	0.4454
Training model form scratch	30	0.7967	0.4328

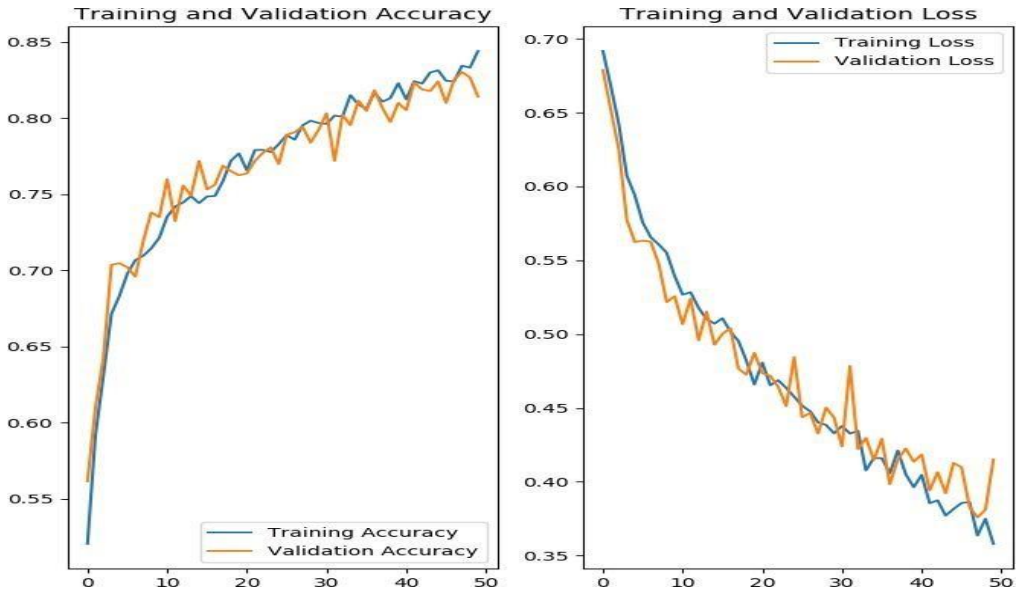


Figure 8: Training and validation accuracy and loss of Experiment 1: training from scratch

The specific data of each experiment are shown in Table 3, Table 4 and Table 5. The left figure in Figure 8 shows the comparison of the change process of training and validation accuracy in 50 epochs of the experiment that retraining the model from scratch [15]. And the right one shows the comparison of the change process of training and validation accuracy in 50 epochs. The results show that the model is free from over fitting and under fitting. As the training process progresses, the model's performance on the training and validation sets remains the same. As the same as the result of the experiment that retraining the model from scratch, the result of the other two experiments shows that neither over fitting nor under fitting occurred [16]. But the result of the last two experiments only covers 20 epochs. Which is shown in Figure 9 and Figure 10. we can find that the change curve of the model trained from scratch is very similar to the change curve of the basic model. As shown in Table 6, after 30 epochs of training, the accuracy and loss rates of the two models are almost the same[17]. This means that if we train a model from scratch to fit new data, it will take almost the same time as the original model to make the new model achieve the same accuracy and loss as the original model.

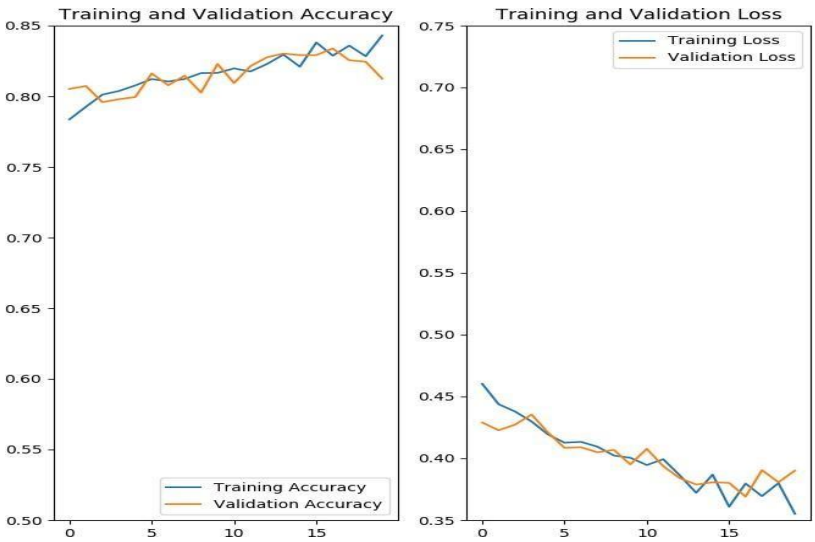


Figure 9: Training and validation accuracy and loss of Experiment 2: training with trained weights

As shown in Figure 9, which is very different from Figure 8, the change curve of the accuracy and loss in Experiment 2 is relatively flat in the first 20 epochs. The performance of the model is slowly improving. The model has a high performance after only one epoch training.

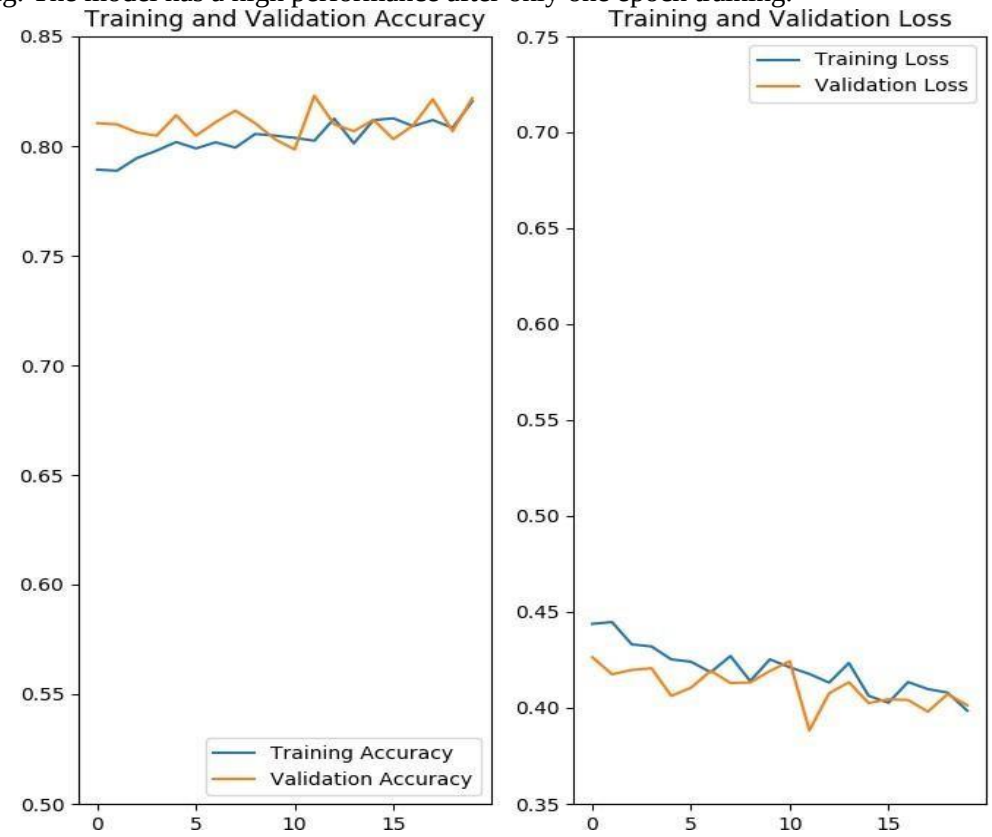


Figure 10: Training and validation accuracy and loss of Experiment 3: training with frozen convolution layers

As shown in Figure 10, the model of Experiment 3 also has a very high performance after only one epoch training. But in the following training process, the model's performance improves very slowly.

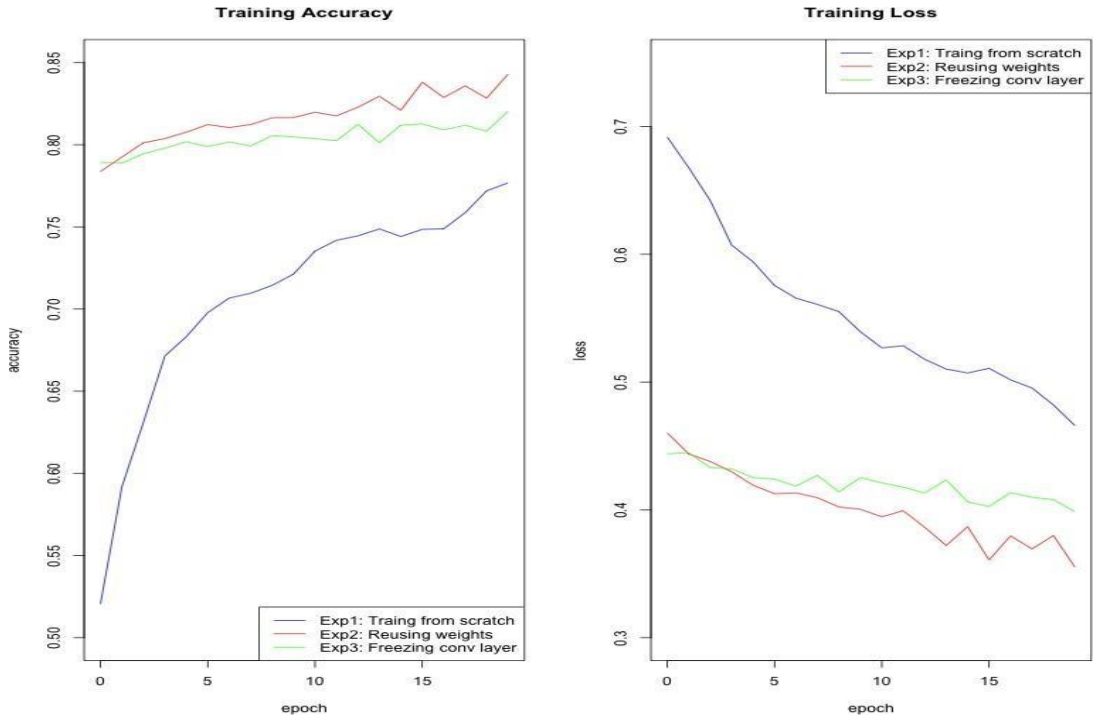


Figure 11: Comparison among training accuracy and loss

B. Comparison of test results

The main content of this section is the comparison of the results of three experiments. The training accuracy and loss for the first 20 epochs of the three experiments are plotted in Figure 11. As shown in Figure 11, the training results of Experiments 2 and 3 in the first 20 epochs are significantly better than Experiment 1. The accuracy of the model of Experiment 1 starts slowly from a very low value, while the models of Experiments 2 and 3 achieved a good performance After an epoch training. If the results of Experiment 2 and Experiment 3 are compared, we can find that the accuracy of the model of Experiment 3 remains basically the same and grows very slowly. However, in the model of Experiment 2, the accuracy rate has continued to increase, and the growth rate is significantly higher than that of Experiment 3. This means that the method of fitting new data with the original model in experiments 2 and 3 is significantly better than training the model from scratch.

The models of Experiments 2 and 3 achieved such high accuracy and such low loss after only one epoch training. This phenomenon seems to indicate that the model perfectly transfers the knowledge obtained by the model on the old data set. The data in Table 7 confirm this.

Table 7: Training accuracy and loss of basic model after epoch 30 and the models of Experiment 2 and Experiment 3 after epoch 1

Models	epoch	Training Accuracy	Loss Training
Training basic model	30	0.7928	0.4454
Loading trained weights	1	0.7837	0.4603
Freezing the convolution layers	1	0.7893	0.4438

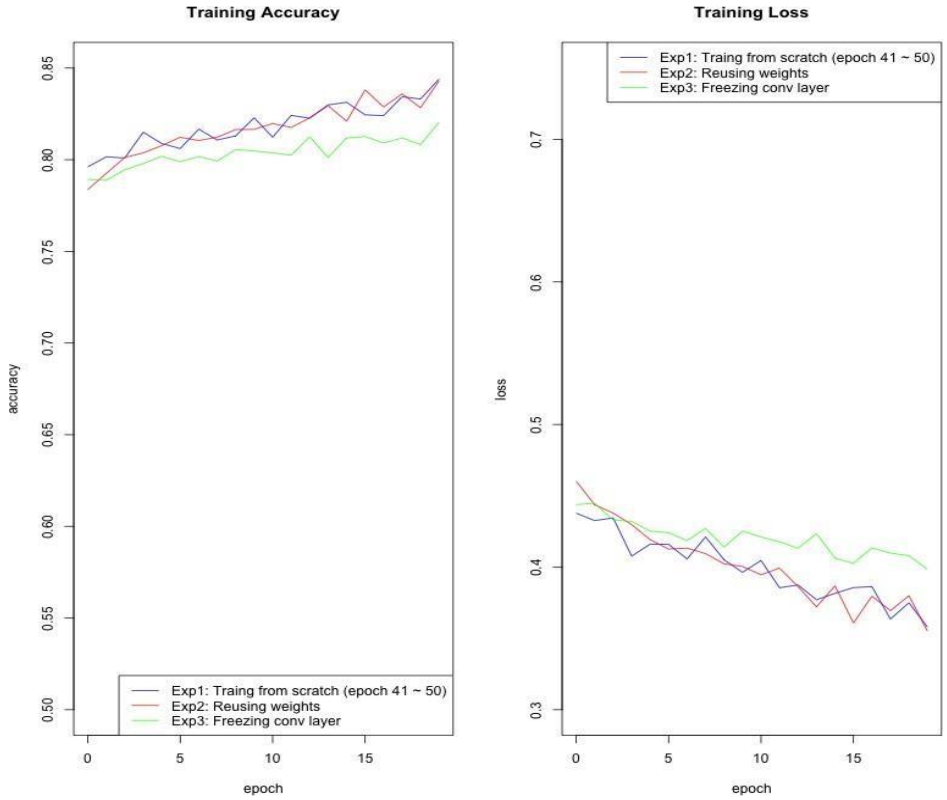


Figure 12: The training accuracy and loss of the Experiment 1 from epoch 41 to epoch 50 and the training accuracy and loss of the Experiment 2 and 3 from epoch 1 to epoch 20

The models in Experiments 2 and 3 transferred the knowledge of the basic model training after 30 epochs. This means that in addition to compare the experimental results of the first 20 epochs of the three experiments as shown in Figure 11. We should also compare the results of epoch 41 to epoch 50 of Experiment 1 with the first twenty epoch results of Experiments 2 and 3. This kind of comparison shows in Figure 12. We can see that the accuracy of Experiment 1 and Experiment 2 increased with basically the same trend, and the loss rate decreased with the same trend too[18]. However, the accuracy and loss rate of Experiment 3 was relatively unchanged. We cannot conclude that freezing the convolutional layer is unnecessary. Let us compare the data in Table 6 and Table 7. We can find that freezing the convolutional layer means that we need to train fewer parameters[19]. The more complex the network structure and the more convolutional layers, the more parameters that do not require retraining [24]. **C. Analysis**

The convolutional layer applies a specified number of convolution filters to the image. It is used to detect the edges' information and patterns of the images[20]. And fully connected layers are used to combine the patterns and classify the image. A CNN model that performs very well means that it has learned enough and accurate edge information and patterns for a certain class of pictures[21]. It's no necessary to make this desired model learn the knowledge it already gained when we won't use this model to fit a similar image dataset if these new images have the same edge information and patterns as the previous ones. **6. Conclusion**

Based on the first research direction, we need to think about the team culture of Data Science, the roles involved in data science, and what barriers exist in each role. Furthermore, we must not only think about the team culture of data science but also how to integrate the data science team with the software engineering team. Because when applying data science in production, the data science team inevitably collaborates with the system development and operation teams. When studying the roles within the science team, the first difficulty we need to overcome is that the discipline of data science is too broad and the roles' responsibility of some branches overlap to others' to some ex- tent. Some common roles in data science are data engineers, data analysts, data scientists, and machine learning engineers. Among them, data scientists are often the "superman" in the team. The skillset they need to master covers data engineering, data analysis, machine learning, deep learning, environment configuration, and programming, etc. Data scientists even have to handle some operational tasks, because the operation team does not understand models as deep as the data scientist. Today, data science is rapidly evolving. In the result that some branches of data science are converging, and new branches are constantly emerging.

So in my opinion, reforming of internal cultural of data science is difficult, and it is difficult to find out a solution that is effective and has long-term effects. Data science is still a very "young" field compared to software engineering. Unlike the latter one, which has achieved business separation after years of development, such as development, testing, and operation. The data science team does not achieve that. And there is no that apparent "wall" be- tween the roles involved in data science. Therefore, I think that at this stage, it is more valuable to integrate the external and internal culture of data science. In other words, the cultural integration of the data science team and the soft- ware engineering team. Depends on this idea we can think that the data science team contains only one kind of super role, the data scientist, then the problem is how to integrate data science, development, and operation, but not how to integrate the roles inside the data science team. I call it Data DevOps. Based on the previous analysis, we can find that the familiar "wall" appears between data scientists and operations. Data scientists and developers need to respond to business requirements quickly and develop new features to respond to rapidly changing markets and data. Meanwhile, the operation team would be worried about this kind of change would negatively affect the stability of the system running in production. Another new "wall" appears be- tween data scientists and developers. System developers need to incorporate models produced by data scientists into the system. Based on the consideration of business separation, we assume that developers do not need to know data science deeply. When a data scientist throws the model to a system developer, it is like the scenario that the developer throws the new features directly to the operation team on another side of the "wall". Just as developers don't understand data science, the operations team doesn't understand development.

References

[1] Wil Van Der Aalst. "Data science in action". In: Process Mining. Springer, 2016, pp. 3–23. [2] Sebastian Neubauer. Oracle Data Science Blog. Jan. 2019. URL: https:// blogs. oracle. com/ datascience/ why- is- it- so- hard-to-put-data-science-in-production.

[3] Jason Yosinski et al. "How transferable are features in deep neural net- works?" In: Advances in neural information processing systems. 2014, pp. 3320–3328.

[4] Foster Provost and Tom Fawcett. "Data science and its relationship to big data and data-driven decision making". In: Big data 1.1 (2013), pp. 51–59.

[5] AD Gordon. Julian Ereth. "DataOps-Towards a Definition." In LWDA. 2018, pp. 104– 112.

[6] W Paul Vogt. Quantitative research methods for professionals. Pear- son/allyn and Bacon Boston, 2007.

[7] Len Bass, Ingo Weber, and Liming Zhu. DevOps: A software architect's perspective. Addison-Wesley Professional, 2015.

[8] Vasant Dhar. "Data science and prediction". In: (2012).

[9] Anthony JG Hey, Stewart Tansley, Kristin M Tolle, et al. The fourth paradigm: data-intensive scientific discovery. Vol. 1. Microsoft research Redmond, WA, 2009.

[10]Thomas H Davenport and DJ Patil. "Data scientist". In: Harvard business review 90.5 (2012), pp. 70–76.

[11]Saeed Aghabozorgi and Polong Lin. Data Scientist vs Data Engineer, What's the difference? 2016. URL: https://cognitiveclass. ai/blog/data-scientist-vs-data-engineer (visited on 08/10/2010).

[12]Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: The handbook of brain theory and neural networks 3361.10 (1995), p. 1995.

[13]Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: nature 521.7553 (2015), p. 436.

[14]Jeremy West, Dan Ventura, and Sean Warnick. "Spring research presentation: A theoretical foundation for inductive transfer". In: Brigham Young University, College of Physical and Mathematical Sciences 1 (2007), p. 32.

[15]Maarten C Kruithof et al. "Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers". In: Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII. Vol. 9995. International Society for Optics and Photonics. 2016, 99950K.

[16]Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: IEEE Transactions on knowledge and data engineering 22.10 (2009), pp. 1345–1359.

[17]Andrej Dyck, Ralf Penners, and Horst Lichter. "Towards definitions for release engineering and DevOps". In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering. IEEE. 2015, pp. 3–3.

[18]Liming Zhu, Len Bass, and George Champlin-Scharff. "DevOps and its practices". In: IEEE Software 33.3 (2016), pp. 32–34.

[19] R Seroter. "Exploring the ENTIRE DevOps Toolchain for (Cloud) Teams". In: Cloud Automation and Management (2014), p. 5.

[20] P.W.D. Charles. Project Title. https://github.com/charlespwd/ project-title. 2013.

[21] Martın Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[22] Angara, J., Gutta, S., Prasad, S. , "DevOps with continuous testing architecture and its metrics model", Advances in Intelligent Systems and Computing , 2018 709 pp271-281 doi://10.1007/978981-10-8633-5_28.

[23] Goyal, D., Goyal, R., Rekha, G., Malik, S., Tyagi, A.K. , "Emerging Trends and Challenges in Data Science and Big Data Analytics", International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE ,2020. Doi://10.1109/ic-ETITE47903.2020.316.

[24] Rajesh, P., Bharadwaj, Alam, M., Tahernezhadi, M , "A Data Science Approach to Football Team Player Selection", 2020 IEEE International Conference on Electro Information Technology, July,pp175- 183.

Doi;// 10.1109/EIT48999.2020.9208331.

[25] Sowjanya, V., Divyambica, C.H., Gopinath, P., Vamsidhar, M., Babu, B.V, "Improved prediction of diabetes based on glucose levels in blood using data science algorithms ", 2019 International Journal of Engineering and Advanced Technology (IJEAT)) volume 8, issue 4, pp 877-881.

[26] Botcha, V.M., Koll, B.P, "Predicting breast cancer using modern data science methodology", International Journal of Innovative Technology and Exploring Engineering (IJITEE) , vol 8 issue 10, pp:4444-4446

Doi:\\ 10.35940/ijitee.J1077.0881019