

**UECS3414 Digital Image Processing**

**Group Assignment**

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Course** |
| Lee Boon Hao | 2106860 | SE |
| Wong Wen Xuan | 2101604 | SE |
| Teo Yan Ru | 2001416 | SE |
| Wong Joel | 1904749 | SE |

Table of Contents

[**Abstract** 3](#_Toc134218384)

[**Introduction** 4](#_Toc134218385)

[**Literature review** 6](#_Toc134218386)

[**Proposed Method** 8](#_Toc134218387)

[**Result and Analysis** 9](#_Toc134218388)

[**I.** **Performance of model by using “adam” optimizer and “accuracy” metrics** 10](#_Toc134218389)

[**II.** **Performance of the model by using “adam” optimizer and “precision” metrics** 11](#_Toc134218390)

[**III.** **Performance of the model by using “adam” optimizer and “recall” metrics** 13](#_Toc134218391)

[**IV.** **Performance of the model by using “sgd” optimizer and “accuracy” metrics** 15](#_Toc134218392)

[**V.** **Performance of the model by using “sgd” optimizer and “precision” metrics** 16](#_Toc134218393)

[**Test result for each model by using different optimizer and metrics** 18](#_Toc134218394)

[**Comparison of model 5 with LeNet-5 model** 19](#_Toc134218395)

[**Assumption** 20](#_Toc134218396)

[**Recommendation** 21](#_Toc134218397)

[**References** 22](#_Toc134218398)

# **Abstract**

Our project roughly describes the development of a texture classification model. This development is to classify and distinguish various texture images obtained by MiTEC from their production line in a large dataset. The dataset covers a variety of different types of texture images, such as rough, smooth, soft, hard, and others. Therefore, our main goal is to enhance the quality management and control of the production line by trying our best to accurately classify and distinguish these different textures. We hope that the model we have drawn up can meet the expectations of the company and let their manufacturing and engineering process be greatly improved. We will use Convolutional Neural Networks (CNN) to accurately classify or identify our texture images and classify them into their respective categories. In addition, we will use the following 6 methodologies to complete our project, such as conducting a thorough Exploratory Data Analysis (EDA) on the texture dataset to gain insights into the data which comprise the statistical analysis, visualization, interpretation of the dataset as well. Besides, we will be preprocessing the texture images by resizing, normalizing, and augmenting the data to improve the model’s performance, which is known as data preprocessing. Furthermore, the third methodology is model architecture. We will design a CNN architecture suitable for the texture classification task which consists of convolutional layers, pooling layers, and fully connected layers. Moreover, the next methodology that will be used is model training. We will train the CNN model on the training set using appropriate optimization techniques such as Stochastic Gradient Descent (SGD) or Adam. Next, model evaluation is our fifth process. We will be evaluating the performance of the classification model using different evaluation metrics such as accuracy, precision, recall, and F1-score. Last but not least, model deployment is our last methodology that will be used. We deploy the model to the testing set and evaluate the performance on the testing set once the model is trained and evaluated.

# **Introduction**

MiTEC, a well-known company in the field of manufacturing and engineering, recently obtained a dataset covering various types of texture images from their production line. After the dataset is generally collected, it is used as a machine learning project plan, and it is used for a long time in various large and small things, especially for analysis (Datagen, n.d.). As for image data, it consists of miscellaneous digital images. These digital images are for the purpose of testing, training and evaluating the functionality of machine learning and artificial intelligence (AI) algorithms (Datagen, n.d.). These datasets of digital images come in various forms such as rough, smooth, soft, hard and others. In addition, image datasets can be helpful in various fields, especially with image recognition or other cognitive activities. For example, reading license plates in photos, identifying differences in medical images, and sometimes even tumors that humans cannot distinguish. All of this can be achieved by training AI algorithms (Datagen, n.d.).

On the other hand, the objective of this assignment is to develop a classification model to classify the different types of textures in the dataset. Classification models can be defined as a subset of supervised machine learning. In general, it reads input and classifies the input into its corresponding category after analysis. Take emails as an instance, when we read emails, sometimes we can’t find some emails. That’s because they are all classified, which is all credited to the classification model. It will analyze and filter the emails sent to our mailbox, and then classify them and put them in different categories, such as spam as well as inbox (Karim, 2021). Moreover, machine learning classification is also commonly used in various fields in real life. In addition to sorting mail, it could also have a place in healthcare. During the epidemic, machine learning models are used to diagnose whether a person is suffering from the epidemic virus (Zoumana, 2022). Furthermore, researchers can also use machine learning models to predict and study new viruses that may appear in the future, so as to take preventive measures before the virus strikes (Zoumana, 2022).

Back to the topic, since the dataset contains various types of texture images, such as rough, smooth, soft, hard and others, we will use the CNN model to help MiTEC solve this problem. Besides, in order to improve the control and management of their manufacturing and engineering process, we will use Convolutional Neural Networks (CNN) to classify these texture images. According to Xin and Wang (2019), Convolutional neural network (CNN) is one of the most representative and most commonly used deep learning frameworks. At the same time, it has many layers of neural networks. Besides, CNN is a deep learning model that is well-known in image analysis. It is usually used to process data in those network modes. It is a spatial hierarchy that automatically learns features within a certain range of adaptation and is a low-to-high-level model.

# **Literature review**

To begin with, CNN has the most important layers, such as input layer, hidden layer and output layer, and there are four most important layers in the hidden layer, namely convolution layers, ReLU layers, pooling layers and fully connected layer (final layer) (Thilo, 2022). These four layers not only happen once, but they can also happen several times before producing output (Thilo, 2022). Next, let's see how the CNN model classifies those images. As we said before, the numerical pixel values ​​of an image will be read by the image classifier first, and then passed through the CNN model to obtain a final output (Thilo, 2022). The obtained output is the type of images after classification. For example, the image of a car is finally classified in the category of "car". What makes all this happen are the hidden layers in the CNN model.

First, let's take a look at the specific role of the convolution layer in CNN. Convolution layers are the main building blocks in image classifiers. It mainly uses mathematical methods to multiply the combination of two functions to obtain a result. It first reads an input and attaches a filter, and the final output is a feature map (Thilo, 2022). A feature map can be defined as a combination of input and filter. In fact, the purpose of convolution is to extract the features of the image. Feature is some unique features of an original image, such as its corners, shapes, etc (Thilo, 2022). Feature is like the image we mentioned before, and its numeric pixel values ​​will also be read (Thilo, 2022). For instance, in image processing, the pixel value pointed to by black is 1, and the pixel value pointed to by white is 0. Suppose we take a (3x3) pixel value to read an all-black feature, and those pixel values ​​will all be 1. The pixel values ​​generated after reading from the feature are also called filter matrix. It’s not over yet, the filter matrix will pass through the entire image, pixel block by pixel block, the purpose is to scan the entire image (Thilo, 2022). After that, a simple multiplication between two matrices will be performed. If the match is more consistent, then the feature map will finally get the higher result (Thilo, 2022).

Next, ReLU is an abbreviation for rectified linear units (Thilo, 2022). It is mainly for activation functions (Thilo, 2022). Thence, the ReLu layer changes the output of the resulting feature map. It converts those negative values ​​to 0, but positive values ​​remain the same (Thilo, 2022). Although there are other nonlinear activation functions, such as sigmoid as well as tanh, which are suitable for image classifiers, ReLU is usually given priority because ReLU can allow most software engineers to obtain results and achieve goals faster and more efficiently (Thilo, 2022).

Apart from that, the pooling layer is also an integral layer of the CNN model. During the pooling process, like the filter of the convolution layer, the pooling layer will also have a filter to walk through the input matrix to generate a new output matrix with a value corresponding to a subregion (Thilo, 2022). In fact, the main aim of pooling is to shrink the image. This will heighten the calculation speed of the image classifier. The most commonly used pooling method is max pooling (Thilo, 2022). An example of max pooling is that there is a 4x4 input matrix, and a 2x2 filter may be used to walk through the entire input matrix. Then this filter will read the largest value of each subregion and produce a new one output matrix (Thilo, 2022). This filter usually travels through the positions without overlapping (Thilo, 2022). For example, in the first filter of 2x2, there are 10, 5, 7, and 30, then the filter will take 30 as a new, 2x2 output matrix. Then, there are 40, 6, 0, and 2 in the second 2x2 filter, then this filter will take 40 and put it in the 2x2 filter. After that, if there are 45, 80, 100, 120 in the third 2x2 filter, then it will take 120. Finally, if there are 5, 8, 30, 90 in the last 2x2 filter, then it will take 90 As the last value placed in the output matrix. Therefore, looking at it from left to right and top to bottom, we will finally get the results of the output matrix to be 30, 40, 120 and 90.

Last but not least, the fully connected layer serves as the last hidden layer of CNN. Its main role is to take the output matrix obtained by the last pooling layer as an input, combine all the information, and finally classify the obtained results (Thilo, 2022). At the same time, each generated value represents the characteristics of the content in those images, for instance, the probability that a paw belongs to a cat is 70%. The other values ​​that appear later may correspond to other parts of it. Therefore, the fully connected layer will combine these acquired information together, and then perform the final classification (Thilo, 2022).

In summary, it is an outline of the main work of each layer of the CNN model.

# **Proposed Method**

# **Result and Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Convolution Layer | Max Pooling Layer | Flatten Layer | Fully-Connected Layer |
| Model 1 | 2 (Kernel size: (3x3), activation function: “relu”, filter: 16, 32) | 2 (Kernel size: (2,2)) | 1 | 2 (1 with 256 units and "relu" activation, 1 with 10 units and "softmax" activation) |
| Model 2 | 2 (Kernel size: (3x3), activation function: “relu”, filter: 32, 64) | 2 (Kernel size: (2,2)) | 1 | 3 (1 with 256 units and "relu" activation, 1 with 128 units and "relu" activation, 1 with 10 units and "softmax" activation) |
| Model 3 | 2 (Kernel size: (3x3), activation function: “relu”, filter: 32, 64) | 2 (Kernel size: (2,2)) | 1 | 2 (1 with 512 units and "relu" activation, 1 with 10 units and "softmax" activation) |
| Model 4 | 1 (Kernel size: (3x3), activation function: “relu”, filter: 32) | 1 (Kernel size: (2,2)) | 1 | 2 (1 with 256 units and "relu" activation, 1 with 10 units and "softmax" activation) |
| Model 5 | 1 (Kernel size: (3x3), activation function: “relu”, filter: 32) | 1 (Kernel size: (2,2)) | 1 | 2 (1 with 256 units and "relu" activation, 1 with 10 units and "softmax" activation) |
| Model 6 | 1 (Kernel size: (3x3), activation function: “relu”, filter: 32, 64, 128, 256) | 4 (Kernel size: (2,2)) | 1 | 2 (1 with 512 units and "relu" activation, 1 with 10 units and "softmax" activation |

Table 1: Models and Layers

Table 1 shows the convolution layer, max pooling layer, flatten layer, and fully connected layer that were be added in each model. As mentioned in the proposed method part, we will train each model by using different optimizer techniques such as Stochastic Gradient Descent (SGD) or Adam. Besides that, we will also evaluate the performance of each model by using different evaluation metrics such as accuracy, precision, and recall. So, we will display the result in five parts which is:

1. Performance of the model by using “adam” optimizer and “accuracy” metrics
2. Performance of the model by using “adam” optimizer and “precision” metrics
3. Performance of the model by using “adam” optimizer and “recall” metrics
4. Performance of the model by using “sgd” optimizer and “accuracy” metrics
5. Performance of the model by using “sgd” optimizer and “precision” metrics

Next, we will make analysis based on each parts and give recommendations about which is the best model to use to make texture classification for this Flickr Material Database (FMD).

## **Performance of model by using “adam” optimizer and “accuracy” metrics**

Figure 1: Test accuracy for each model by using the “adam” optimizer and “accuracy” metrics

Figure 1 above shows the line graph of test accuracy for each model by using “adam” as optimizer and “accuracy” as the metrics during each epoch. From this figure, we can see that model 1 will be the fastest model to reach 100% test accuracy. The model will reach 100% test accuracy when the epoch reaches 7. Next, models 2, 3, 4, and 5 will also reach 100% test accuracy when the epoch reaches 8 to 10. However, since there is a possibility that the model 1 to 5 has overfitting problem, so in the model 6, we try to add more convolutional and max pooling layer to make the model become more complex and try to overcome the overfitting problem. Thus, the test accuracy for model 6 will not be as good as other models. It will reach 100% test accuracy only when the epoch reaches 20.

Figure 2: Average test accuracy for each model by using “adam” optimizer and “accuracy” metrics

Figure 2 shows the average test accuracy after we trained, evaluated the model and deployed the testing set. We found out that these 6 models will have average test accuracy around 55% to 65%. The average test accuracy is quite low because the dataset for the Flickr Material Database (FMD) is very low. It only has 100 images for each class. The dataset could not provide diverse examples of the types of the image that the model is likely to encounter. As a result, the model will not be able to learn the underlying pattern and features in the data and thus make it hard to make accurate predictions for unseen data. The model that have the highest average test accuracy is model 1 and 4, which has 62% of test accuracy.

## **Performance of the model by using “adam” optimizer and “precision” metrics**

Figure 3: Test precision for each model by using the “adam” optimizer and “precision” metrics

Figure 3 above shows the line graph of test precision for each model by using “adam” as optimizer and “precision” as the metrics during each epoch. From this figure, we can see those models 1, 2, 3, 4, and 5 tend to have test precision of 100% when the epoch reaches around 6 to 7. However, since there is a possibility that the model 1 to 5 has overfitting problem, so in the model 6, we try to add more convolutional and max pooling layer to make the model become more complex and try to overcome the overfitting problem. Thus, the test precision for model 6 will not be as good as other models. We can see that model 6 will have 0% of test precision when the epoch reaches 1, 2, and 3. However, the test precision starts to increase during the fourth iteration over the entire training dataset. The test precision for model 6 will keep fluctuate and increase until it reaches around 95% when the epochs is 20.

Figure 4: Average test precision for each model by using “adam” optimizer and “precision” metrics

Figure 4 shows the average test accuracy after we trained, evaluated the model and deployed the testing set. We found out that these 6 models will have average test accuracy around 60% to 75%. The average test accuracy is quite low because the dataset for the Flickr Material Database (FMD) is very low. It only has 100 images for each class. The dataset could not provide diverse examples of the types of the image that the model is likely to encounter. As a result, the model will not be able to learn the underlying pattern and features in the data and thus make it hard to make accurate predictions for unseen data. However, the average test precision for each model is higher than the average test accuracy for each model that is mentioned in part I. So, metrics of “precision” is a much more suitable metrics to be used that metrics “accuracy” for the model. The model that has the highest average test is model 5 which have the average test accuracy of 75.5%.

## **Performance of the model by using “adam” optimizer and “recall” metrics**

Figure 5: Test recall for each model by using the “adam” optimizer and “recall” metrics

Figure 5 above shows the line graph of test recall for each model by using “adam” as optimizer and “recall” as the metrics during each epoch. From this figure, we can see those models 1, 2, 3, 4, and 5 tend to have test precision of 100% when the epoch reaches around 6 to 10. However, since there is a possibility that the model 1 to 5 has overfitting problem, so in the model 6, we try to add more convolutional and max pooling layer to make the model become more complex and try to overcome the overfitting problem. Thus, the test recall for model 6 will not be as good as other models. We can see that model 6 will have less than 20% of test recall during the first 10 iterations over the entire training dataset. However, the test precision starts to increase during the next iteration over the entire training dataset. The test precision for model 6 will keep fluctuate and increase until it reaches around 95% when the epochs is 20.

Figure 4: Average test precision for each model by using “adam” optimizer and “precision” metrics

Figure 4 shows the average test accuracy after we trained, evaluated the model and deployed the testing set. We found out that these 6 models will have average test accuracy around 57% to 60%. The average test accuracy is quite low because the dataset for the Flickr Material Database (FMD) is very low. It only has 100 images for each class. The dataset could not provide diverse examples of the types of the image that the model is likely to encounter. As a result, the model will not be able to learn the underlying pattern and features in the data and thus make it hard to make accurate predictions for unseen data. Since the average test recall for each model is lower than the average test precision for each model mentioned in part II, the “recall” metrics may not be suitable to be used for the classification of model for this database. The model that has the highest average test recall is model 3 which have the average test recall of 67.5%.

## **Performance of the model by using “sgd” optimizer and “accuracy” metrics**

Figure 7: Test accuracy for each model by using the “sgd” optimizer and “accuracy” metrics

Figure 7 above shows the line graph of test accuracy for each model by using “sgd” as optimizer and “accuracy” as the metrics during each epoch. From this figure, we can see that performance for model 1 is the best among other model as the test accuracy for this model during each epoch is the highest among other models. The performance for model 4 is also not bad as the test accuracy for this model is slightly slower than model 1 during each epoch. However, the performance for other model is not good enough if compared to the performance of model 1 and 4. This may be due to the number of training data or the dataset is too low. Since the number of images for each classes is only 100, which is significantly lower than expected, so the model will not be able to learn the underlying pattern and features in the data and thus make it hard to make accurate predictions for unseen data. Besides that, the reason why the test accuracy is so low might also because the “sgd” optimizer is not suitable to be used for this model as the test accuracy mentioned in part I, which is using “adam” optimizer to train the model tends to have better test accuracy. Besides that, the “sgd” optimizer that we are using for this model might need to be corporate with other metrics to have better result. So, we will the “precision” metrics and “sgd” optimizer to evaluate the performance of each model in part V.

Figure 8: Average test accuracy for each model by using “sgd” optimizer and “accuracy” metrics

Figure 8 shows the average test accuracy after we trained, evaluated the model and deployed the testing set. We found out that these 6 models will have average test accuracy around 24% to 62%. The average test accuracy is quite low because the dataset for the Flickr Material Database (FMD) is very low. It only has 100 images for each class. The dataset could not provide diverse examples of the types of the image that the model is likely to encounter. Besides that, the average test accuracy is quite low may also because the “sgd” optimizer is not suitable for this model or “sgd” optimizer and “accuracy” metrics could not work well with the model. So, we will try “sgd” optimizer and “precision” metrics in part V to find out whether the “sgd” optimizer will affect the performance of the model. From this figure, we can also see that the highest average test accuracy is model 5, which has the average test accuracy of 61.75%.

## **Performance of the model by using “sgd” optimizer and “precision” metrics**

Figure 9: Test precision for each model by using the “sgd” optimizer and “precision” metrics

Figure 9 above shows the line graph of test precision for each model by using “sgd” as optimizer and “precision” as the metrics during each epoch. From this figure, we can see that most of the models will have very low test precision during the first five to ten epochs. This is because the dataset did not have large amount of images, so the model cannot make accurate prediction for the test result, and it will lead to biases in the model’s predictions, as the model may not have seen enough examples of certain patterns or features to recognize them in new images. Next, we can also see that the test precision for each model tends to increase when the epochs increase. This is due to training for more epochs allows the model to learn more complex patterns in the data, and thus th model will be able to make more accurate predictions on new or unseen data.

Figure 10: Average test precision for each model by using “sgd” optimizer and “precision” metrics

Figure 10 shows the average test accuracy after we trained, evaluated the model and deployed the testing set. We found out that these 6 models will have average test accuracy around 64% to 94%. From this figure, we can see that the average test precision for each model is higher than the average test accuracy for each model that is mentioned in part IV, which means that the “precision” metrics can work well with the “sgd” optimizer for this model. The model that has the highest average test is model 2 which have the average test precision of 93.33%. Besides that, we can also see that the average test precision for each model (which is by using “sgd” optimizer and “precision” metrics) is the highest that we may see for the model. We can make a conclusion saying that this “sgd” optimizer and “precision” metrics is the best optimizer and metrics to be used for the model.

## **Test result for each model by using different optimizer and metrics**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model/Optimizer and Metrics | adam\_accuracy | adam\_precision | adam\_recall | sgd\_accuracy | sgd\_precision | Mean |
| Model 1 | 0.62 | 0.6965 | 0.5875 | 0.4825 | 0.6447 | 0.60624 |
| Model 2 | 0.5775 | 0.6596 | 0.5875 | 0.32 | 0.9333 | 0.61558 |
| Model 3 | 0.6075 | 0.6474 | 0.6075 | 0.42 | 0.7228 | 0.60104 |
| Model 4 | 0.62 | 0.754 | 0.59 | 0.595 | 0.8133 | 0.67446 |
| Model 5 | 0.615 | 0.755 | 0.5725 | 0.6175 | 0.8495 | 0.6819 |
| Model 6 | 0.605 | 0.6308 | 0.6025 | 0.24 | 0.875 | 0.59066 |

Table 2: Test result for each model by using different optimizer and metrics

Figure 11: Test result for each model by using different optimizer and metrics

Table 2 shows the test result for each model by using different optimizer and metrics and the data is in statistical form, while figure 11 shows the test result for each model by using different optimizer and metrics in columns form. From this table and figure, we can see that each model will perform differently when using different optimizer and metrics. For example: model 2 has the highest test precision when using “sgd” as optimizer and “precision” as metrics if we compare the result with another model. However, when using “sgd” as optimizer and “accuracy” as metrics, the model will perform badly as it will have the lowest test accuracy result. So, we cannot choose this model as the best model to make texture classification for this database. However, from the figure 11, we can see that model 5 perform very steadily even we use any optimizer or metrics to evaluate the model, and it has the mean result of 68.19%, which is the highest mean test result among other models. So, in short, model 5 is the best model to make texture classification for this database. Thus, we will compare the result that we get for model 5 with LeNet-5 model.

## **Comparison of model 5 with LeNet-5 model**

Figure 12: Test accuracy during each epoch between model 5 and LeNet-5 model

Figure 12 shows the test accuracy during each epoch between model 5 and LeNet-5 model. From this figure, we can see that the test accuracy for model 5 will reach 100% when the epoch reaches 8. However, the test accuracy for LeNet-5 model is very low. When the epoch reaches 8, the test accuracy for LeNet-5 model is around 20%, and the highest test accuracy for LeNet-5 model is only around 35%. The test accuracy for LeNet-5 model is significantly lower than the test accuracy for model 5 that we have created.

Figure 13: Average test accuracy between model 5 and LeNet-5 model

Figure 13 shows the average test accuracy between model 5 and LeNet-5 model after we trained, evaluated the model, and deployed the testing set. We found out that the average test accuracy for model 5 is 61.5%, while the test accuracy for LeNet-5 model is just around 23.25%. The reason why the test accuracy for LeNet-5 model is very low is because LeNet-5 model is used to classify digit in grayscale, and it is not suitable to use to classify texture in color. So, the result for LeNet-5 model will be very not accurate.

## **Assumption**

1. For more details test result, please refer to ‘Analysis.xlsx’ and ‘FullAnalysis.xlsx’.

# **Recommendation**

There are some recommendations that we decided to propose in order to increase the test accuracy of the model.

1. Increase the number of dataset

Firstly, number of images for each class should be increased in order to increase the test result of the model. This is because the CNN model that we used will learn from the training data and generalize to unseen data. If the training data is limited or not diverse enough, the model may not learn all the necessary features to perform well on the test data. Therefore, using a larger and more diverse training dataset can help to improve the test accuracy of the model.

1. Use regularization techniques

Regularization is a technique that helps to prevent overfitting by adding penalties to the model's loss function. Common regularization techniques for CNNs include L1 and L2 regularization, dropout, and batch normalization. Bu using this regularization techniques, the performance of the model will increase.

1. Perform model assembling

Combine the predictions of multiple CNN models to improve performance. You can train several CNN models with different initializations or architectures and then average their predictions or use more advanced assembling techniques such as stacking or boosting.

# **References**

Datagen (n.d.). 34 Amazing Image Datasets for Your Next CV Project. [online] datagen.tech. Available at: https://datagen.tech/guides/image-datasets/image-datasets/ [Accessed 1 May 2023].

Karim, F. (2021). Classification models & thresholds - by Karim Fanous. [online] karimfanous.substack.com. Available at: https://karimfanous.substack.com/p/classification-models-and-thresholds [Accessed 1 May 2023].

Thilo, H. (2022). Image Processing: How Do Image Classifiers Work?. [online] levity.ai. Available at: https://levity.ai/blog/how-do-image-classifiers-work [Accessed 1 May 2023].

Xin, M. and Wang, Y. (2019). Research on image classification model based on deep convolution neural network. EURASIP Journal on Image and Video Processing, [online] volume 2019, (Article number: 40). Available at: https://link.springer.com/article/10.1186/s13640-019-0417-8. [Accessed 1 May 2023].

Zoumana, K. (2022). Classification in Machine Learning: An Introduction. [online] datacamp.com. Available at: https://www.datacamp.com/blog/classification-machine-learning [Accessed 1 May 2023].