

# BINARY IMAGE CLASSIFICATION OF MELT POOL USING CONVOLUTION NEURAL NETWORK

Nikulkumar Dabhi (400490758); Pavani Jana (400492032); Maulik Khanakara (400305736); Rudra Patel (400406223)

**ABSTRACT:** This Project aims to detect the presence of a melt pool by employing convolutional neural networks (CNN). A new CNN model was developed, trained and analyzed for optimistic results. The model was analyzed by varying different parameters and by comparing it to other existing supervised learning models. The present model showed better accuracy than existing models for the given dataset and the model is performing optimum with the sigmoid activation function.

## I. INTRODUCTION

Machine learning is a subset of Artificial intelligence that learns from the data and uses it for a specific task like a prediction. Machine learning is categorized into three types of algorithms: supervised, unsupervised and Reinforcement learning. Supervised learning is where the machine is provided with labelled data and training, whereas unsupervised learning learns from the data without any labelling.

Supervised learning algorithms are trained using a labelled dataset, where the images were given a specific label associated with the expected outcome. It teaches the machine to predict the correct label for the new dataset. In the case of image classification, images were given as input and the outcome will be in the form of pre-defined labels.

As the images that are processed through ML are of high resolutions, it is important to extract features and shorten them to get desired and efficient outputs. For this, Convolutional neural networks (CNN) plays an important role, which we are using for our present model. CNNs are best known for identifying images. These networks can extract various aspects of images by using convolution and pooling operations of images and reducing the input size to turn them into binary results.

While using the CNN model, the dataset should be divided into three parts: training, testing and validation. The images should be randomly split into these sub-sets with a ratio such that the training set has a higher number of images compared to testing and validation. Training enables the model to extract major features of the image while the validation set is used to prevent over-fitting of the images by controlling the network parameters. The performance of the model is then assessed using the testing dataset.

After training the CNN, we can apply it to classify new images by passing them through the network and receiving a predicted label. The predicted label is determined by the category with the highest probability in the final activation layer of the network. We can evaluate the model's accuracy on the testing set by comparing predicted and actual labels.

## II. MELT POOL

### A. Data acquisition

Data acquisition is the process of acquiring data from multiple sources or sensors and recording it for later analysis and processing. It entails gathering raw data and transforming it into a form that can be used for automation, research, analysis, monitoring, and a variety of other functions.

Data collection may be done in a number of ways, including manually entering data, automating data gathering with sensors, tools, or equipment, extracting data from outside sources, and combining data from several sources. According to the application and the type of data being collected, the obtained data may be in a variety of forms, including numerical data, text, photos, audio, video, or other sorts of data.

The quality, precision, and reliability of the data that is obtained directly affect the quality of the subsequent data analysis and decision-making, making data collection an essential phase in the data lifecycle. To make sure that the gathered data is precise, consistent, and appropriate for the intended purpose, it frequently incorporates data validation, data cleansing, and data transformation operations.

In our project, the images are taken from a camera, which generates a 128X120 size bmp image through the additive manufacturing process and is stored in various image folders. These data are further manually sorted into two types: melt pool and no melt pool. In the melt pool, all the defects are taken into consideration in one data set, and in the no-melt pool, dark images with no defects are counted.

### B. Images data description

When anybody uses the term "image data description," they often mean the process of describing or characterizing images using different machine learning or computer vision techniques. In order to perform tasks like image recognition, image retrieval, image segmentation, and object detection, it is necessary to extract useful information or characteristics from images. This is known as image data description.

**Handcrafted features:** In this method, features from images, such as colour histograms, texture features, edge features, or local binary patterns, are manually designed or chosen. The models created by machine learning algorithms using these attributes are then applied to tasks involving picture analysis.

Convolutional neural networks (CNNs) have been extensively employed for picture data description since the advent of deep learning. During the training phase, CNNs automatically learn structures from the images, and these features can be utilized to represent images in a variety of tasks.

### C. Data processing

Data processing is the process of converting unprocessed data into information that is both useful and meaningful. In order to produce insights, results, and reports that may be utilized for judgment, problem-solving, and other business or academic reasons, it entails the collection, organization, manipulation, and analysis of data.

Although it can be done manually, software tools and computer-based systems are commonly used for data processing. Data collection, data entry, data validation, data cleansing, data transformation, data storage, data analysis, and data visualization are just a few of the steps that the process may go through. Depending on the type, volume, and complexity of the data being processed, each stage may entail a variety of approaches, methodologies, and tools.

Seeing the characteristics of the images, manually segregating the images into the 2 groups: Melt pool and no melt pool which cannot be defined by the code as there were some errors due to which the perfect classification cannot be obtained which was manually separated.

### D. Labelling

Data labelling is crucial to machine learning because it offers real-world labels that can be used as a benchmark when developing and testing machine learning models. Machine learning models' performance and dependability can be strongly impacted by the quality and accuracy of data labelling, hence special consideration should be made to guarantee correct and consistent labelling for efficient model development.

A well-known open-source Python package called NumPy (Numeric Python) offers support for carrying out numerical operations on arrays and matrices. A NumPy array, sometimes referred to as an array (N-dimensional array), is a fundamental data structure made available by NumPy. It is a multi-dimensional, standard data container for storing and effectively processing huge amounts of data.

A NumPy array, also known as a 1D (one-dimensional), 2D (two-dimensional), or higher-dimensional array, is a grid of values that can have one or more dimensions. The same data type, such as integers, floating-point numbers, or other sorts of data, may be included among its components.

According to this, datasets the images are classified as (1,0) which gives identification to the melt pool as 1 and no melt pool as 0.

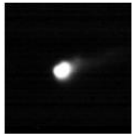

	Melt Pool Image Example	No Melt Pool Image Example
Images (x_train)		
Labels (y_train)	Label = 1	Label = 0

Fig. 1: Labelling the images

## III. METHODOLOGY

In the process of learning from labelled data to produce predictions or judgements, an algorithm is said to be learning under supervision. In supervised learning, the algorithm is trained on a labelled dataset that consists of related output data and input data (commonly referred to as features) (often called labels or targets). Using the given labelled data as a guide, the algorithm learns how to translate inputs to outputs.

The purpose of supervised learning is to develop an algorithm that can correctly anticipate an output given unknown input data. The algorithm generates predictions by extrapolating from the patterns it discovers during training on the labelled data. The algorithm may be used to generate predictions on fresh, unexplored data after being taught.

Applications for supervised learning include recognition of speech, identifying spam, sentiment analysis, image classification, and many more tasks where labelled data is available. Decision tree models, support vector machines, logistic regression (LR), and artificial neural networks are typical supervised learning methods.

In machine learning, a form of supervised learning job called binary classification has the objective of categorizing input data into one of two potential classes or categories. In other terms, it requires determining which of two mutually exclusive groups a given input data item belongs to.

Binary classification often uses two labels to represent the two classes, such as 0 and 1, positive and negative, or yes and no. The objective is to develop a model that can precisely predict the class label based on the feature values. The input data used for binary classification is frequently represented as a set of features or attributes.

Many different applications use binary classification, including spam detection (classifying emails as spam or not spam), medical diagnosis (determining whether a patient has a specific condition or not), sentiment analysis (classifying text as positive or negative), and fraud detection (determining whether a transaction is fraudulent or legitimate).

Logistic regression, assistance vector machines, decision trees, random forests, and deep neural networks are common binary classification techniques. Accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve are some of the evaluation measures used for binary classification.

Hardware	software
16 GB RAM	Python
CPU I7	Jupyter Operating System: Windows

Fig.2: Machine Specifications

#### IV. MODEL

##### A. Model Architecture

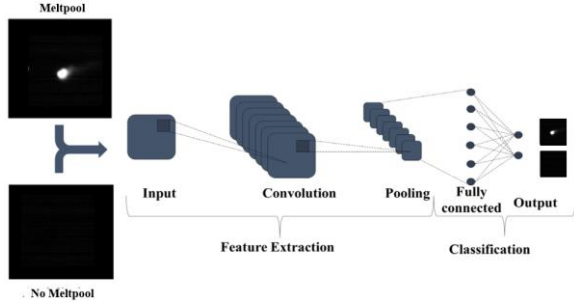


Fig.3: Model Architecture

Below defined is our model architecture consisting of convolution and max pooling layers at the core. The convolution layer is used with kernel size of 3X3 normally and varied for analysis while the max pooling layer is kept at size of 2X2. Initially the model was trained and validated for 3 convolution and 3 max pooling layers but as the model proved to be too strong for the classification process, only 1 convolution and 1 max pooling layer were utilized.

Moreover, for the activation function between the 2 layers, the Rectified Linear Unit (ReLU) function is used, while for the smoothing of the output data, the sigmoid function is used.

```
# Define the CNN model architecture using the functional API

inputs = tf.keras.Input(shape=(height, width, 3))
x = tf.keras.layers.Conv2D(filters=z, kernel_size=(f, f), activation='relu')(inputs)
x = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(x)

# x = tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu')(x)
# x = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(x)

# x = tf.keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu')(x)
# x = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(x)

x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(units=512, activation='relu')(x)
# x = tf.keras.layers.Dropout(0.1)(x)

outputs = tf.keras.layers.Dense(units=1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

Fig.4: Model

Thus, the flowchart below appropriately describes the flow of images in the form of data through the model.

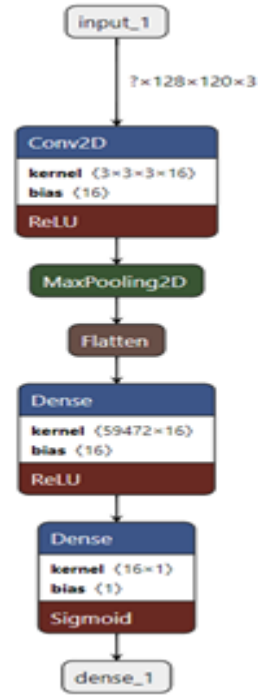


Fig.5: Flowchart describing steps involved in model

##### B. Compile the model

Compiling a model is necessary to finish the model and make it fully functional. We must define an optimizer and a loss function for compilation. With the compile property, we can compile a model.

Before we use it, let us first examine its specifications of optimizer: Here, we may specify the optimizer to use. SGD, Adam, and more optimizers are available.

- Loss: In this, we may specify a loss function for the model.
- Metrics: We can pass the measure by which we want the model to be graded here.

Currently, to compile the model, Adam is used as an optimizer, as based on research and analysis, we found that Adam is the most efficient optimizer for binary classification models. We used binary cross entropy as the loss function.

##### C. Training the model

```
# Split the data into train and validation dataset
x_train, x_val, y_train, y_val = train_test_split(images, labels, test_size=0.2, r

# Train the model on the training data and validation
history = model.fit(x_train, y_train, epochs=5,
                    batch_size=20, validation_data=(x_val, y_val))
```

Fig 6: Training

Learning (determining) suitable values for all of the weights and the bias from labeled samples is what training a model is all about. A machine learning algorithm generates a model in supervised learning by studying numerous instances and tries to find a model that minimizes loss.

An epoch is defined as the total number of iterations of all the training data in one cycle for training the machine learning model when all of the training data is used at once. Every sample in the training dataset has had one epoch to change the internal model parameters. An era is made up of one or more batches.

For example, the batch gradient descent learning approach is used to define an Epoch with only one batch. To train the model, 5 epochs and batch size of 20 was used. The dataset selected for training is divided into 2 parts i.e., with melt pool and no melt pool. Now data is retrieved from these sets rationally, keeping the same number of images from both folders and using 80 percent of the dataset for training and 20 percent of the dataset for validation.

```
Epoch 1/5
3/3 [=====] - 1s 245ms/step - loss: 112.0637 - accuracy: 0.5312 - val_loss: 0.3992 - val_accuracy: 0.8750
Epoch 2/5
3/3 [=====] - 0s 49ms/step - loss: 42.2155 - accuracy: 0.6562 - val_loss: 25.9841 - val_accuracy: 0.3750
Epoch 3/5
3/3 [=====] - 0s 48ms/step - loss: 7.3077 - accuracy: 0.8125 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 4/5
3/3 [=====] - 0s 76ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 5/5
3/3 [=====] - 0s 70ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

Fig.7: Epochs

#### D. Validate the Model

Model validation is the process of evaluating the trained model with a testing data set following model Training. The testing data may or may not be a subset of the same data set as the training data. In our project the testing data is kept different from the data used for training and validation. It is critical to validate the machine learning model outputs to assure their correctness.

A large amount of training data is utilized while training a machine learning model, and the main goal of validating the model validation gives a chance for machine learning engineers to increase the data quality and quantity. As it occurs, relying on the model's forecast without first evaluating and confirming it is not a good idea.

In sensitive sectors such as healthcare and self-driving vehicles, any error in object identification might result in severe deaths owing to incorrect judgements made by the machine in real-life forecasts. The following are some additional benefits of Model validation:

- Scalability and adaptability
- Minimize your expenses.
- Improve the model's quality.
- More mistakes are being discovered.
- Prevents the model from becoming overfit or underfit.

Furthermore, verifying the ML model throughout the training and development stages aids the model in making accurate predictions.

While there is no direct relationship between training accuracy, validation accuracy and training loss, validation loss

but it is observed that there is an inverse relationship between the two.

```
# Evaluate the model on the validation data
val_loss, val_acc = model.evaluate(x_val, y_val)
```

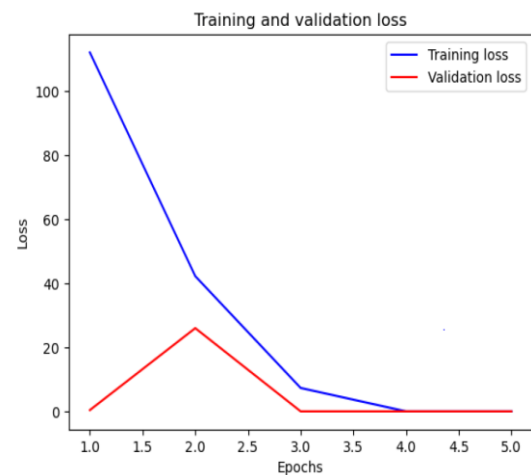
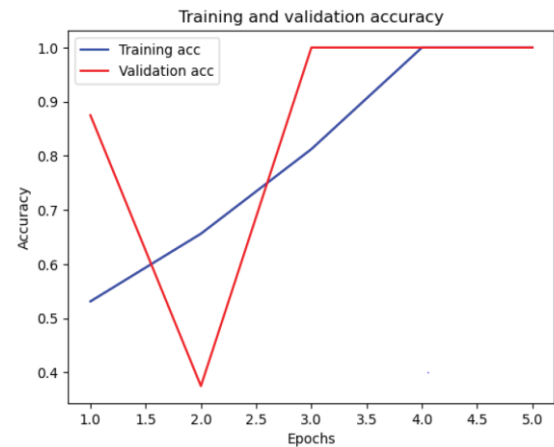


Fig.8: Training and Validation

#### E. Testing the Model

In our project, we assigned directories to test images after resizing if needed to resize. Then, the images are shuffled along with labels to randomize the test dataset. Next, the images are being predicted by the model and classified based on melt pool and no melt pool. Finally, the image output is compared to the predicted output to analyze model accuracy.

```
# Predict labels for each image in test data
y_pred = model.predict(x_test)
```

```
y_pred
array([0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1])

y_test
array([1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

Fig.9: Label Prediction

#### F. Matrix for Analysis

The confusion matrix was used for the presentation assessments of the techniques utilized after the arrangement. The prediction summary is represented in matrix form by a confusion matrix. It depicts the number of correct and incorrect predictions for each class. It aids in the comprehension of classes that are misunderstood by the model as other classes.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Fig.10: Confusion Matrix

**True Positive (TP):** The positive class is correctly predicted by the model (the prediction and the actual are both positive). In our case, we predicted an image with melt pool as an image with melt pool correctly.

**True Negative (TN):** The class that the model correctly predicts is negative (both the prediction and the actual are negative). In our case, predicting an image with melt pool as an image without melt pool incorrectly.

**False Positive (FP):** The model predicts the incorrect negative class (predicted-positive, actual-negative). In our case, predicting an image without melt pool as an image without melt pool correctly.

**False Negative (FN):** model wrongly predicts the positive class (anticipated negative, real certain). In our case, predicting an image without melt pool as an image with melt pool incorrectly.

For a test input of total 200 melt pool images and 200 no melt pool images, the confusion matrix is as below:

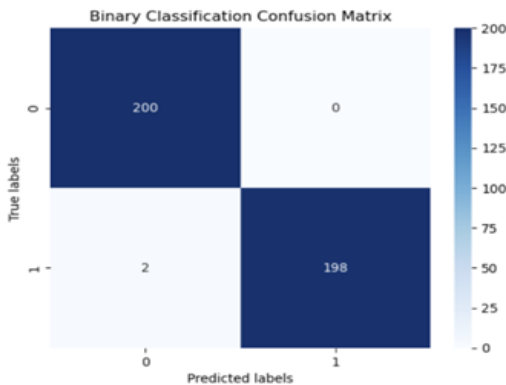


Fig.11: Confusion matrix based on model data.

The above confusion matrix for a test set of 200 melt pool and 200 no melt pool images show model predicting 200 true positive images, 0 false positive images, 2 false negative images and 198 true negative images. Other mathematical parameters to predict effectiveness of model are precision, recall, F1 score and accuracy which are formulated in next section.

#### G. Metrics to understand the performance of the trained model

$$\begin{aligned}
 \text{precision} &= \frac{TP}{TP + FP} \\
 \text{recall} &= \frac{TP}{TP + FN} \\
 F1 &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \\
 \text{accuracy} &= \frac{TP + TN}{TP + FN + TN + FP}
 \end{aligned}$$

Fig.12: Formulae of model metrics calculations

**Accuracy:** The ratio of true positives and true negatives to all positive and negative observations is the model accuracy metric, which is used to evaluate the performance of machine learning classification models. In other words, accuracy tells us how often, out of the total number of predictions it makes, our machine learning model will correctly predict an outcome. Numerically, it addresses the proportion of genuine positives and genuine negatives out of the multitude of expectations.

**Precision:** The model precision score estimates the extent of decidedly anticipated marks that are really right. The positive predictive value is another name for precision. In order to make a trade-off between false positives and false negatives, precision and recall are used together. The distribution of classes has an impact on precision. The quality or exactness of something can be measured in terms of precision. We would select a model with high precision if we want to reduce the number of false negatives. When there is a greater cost associated with false positives than with false negatives, such as in medical diagnosis or spam filtering, precision is mostly used to predict the positive class. When the classes are very unbalanced, the precision score is a useful way to measure how well predictions worked. It is the ratio of true positives to the total of true positives and false positives, mathematically speaking.

**Recall:** The model's recall score shows how well it can correctly predict positive outcomes based on actual positive outcomes. This is not normal for precision, which estimates the number of expectations made by models that are really certain out of all sure forecasts made. For instance: The recall score would be the percentage of positive reviews that your machine learning model correctly predicted as positive if it were attempting to identify positive reviews. To put it another way, it measures how adept our machine-learning model is at locating every genuine positive in a dataset. The true positive rate and sensitivity are other names for recall.

The higher the review score, the better the AI model is at recognizing both positive and negative models. A model's



ability to identify positive examples is demonstrated by its high recall score. A low recall score, on the other hand, indicates that the model does not do a good job of identifying positive examples. It is a mathematical representation of the ratio of true positives to the total of false negatives and true positives.

**F1-score:** The model score as a function of precision and recall is represented by the model F1 score. F-score is an AI model execution metric that gives an equivalent load to both accuracy and review when estimating its exhibition with regards to precision, making it an option in contrast to exactness measurements (it doesn't expect us to know the complete number of perceptions). It is frequently utilized as a single value that provides high-level information regarding the quality of the model's output. When attempting to optimize either the precision or recall score, this is a useful model measure in situations where model performance suffers. It can be represented mathematically as a harmonic mean of the precision and recall scores [5].

Confusion matrix, precision, recall, and F1 score gives better experiences into the forecast when contrasted with exactness execution measurements. Information retrieval, word segmentation, named entity recognition, and numerous other applications make use of precision, recall, and the F1 score.

## V. RESULTS & DISCUSSION

### A. Based on the learning rate

The learning rate is an important parameter for training the machine learning model. The learning rate talks about how fast or slowly the model is learning the data. It defines step size after which the optimizer updates its weights.

If the learning rate is too high, the model will optimize its weight in large step size resulting in divergence or overshoot which consequence in overfitting and less accuracy of the model. On the other hand, if the learning rate is too slow the model will take too much time to converge and will stop changing its weight at a certain point giving constant accuracy after a few epochs.

From the training of our model with different learning rates we analyzed that when the learning rate decreases the training accuracy and validation accuracy increase this is because the model optimizes weight in small step size and reverses for a high learning rate.

So, the best learning rate is the one which gives maximum accuracy and minimum loss of training and validation. To optimize the learning rate, we trained our model on different learning rates and its results are shown in below bar chart.

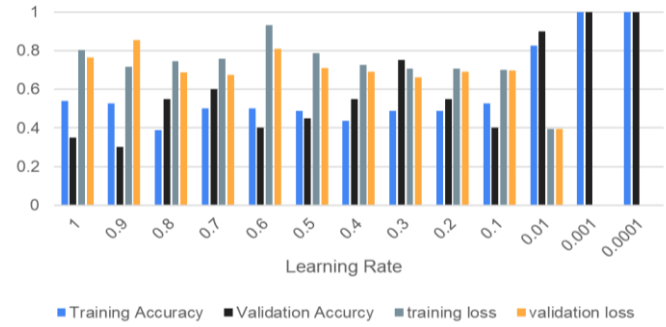


Fig.13: Learning Rate V/s Accuracy and loss

### B. Learning Rate v/s Training Time (seconds)

Training time is the time a CNN model takes to learn the features of any input and it depends on many other variables. Keeping every other thing constant, the learning rate increases training time decreases as the weights are optimized in small step size and, this needs more time to train the model. Inversely if the learning rate is so high it is obvious that the time needed to train the model decreases as model weights drastically updated. Hence, an optimum learning rate is essential to train the model faster and achieve the highest accuracy.

To find the optimum value of the learning rate we tried various values of learning rate with Adam optimizer and its training time in seconds is shown in the below graph.



Fig.14: Learning Rate V/s Training Time in seconds

By comparing the accuracy of the model from analysis 1 and the time taken for training the model in the above graph. It is noteworthy that a learning rate of 0.001 the be results for our classification problem.

### C. Activation function v/s accuracy for CNN model

The activation function plays a vital role in the performance of the model. It introduces nonlinearity into the model. It describes how well the model learns complex characteristics from the given data set.

There are several activation functions for the training of the CNN model. The below table represents the well-known activation function for non-linearity and its equation to the output of convolutional layers. Selection varies as per the task to perform and the accuracy to achieve.

Activation Function	Equation
Sigmoid	$S(x) = \frac{1}{1 + e^{-x}}$
SoftMax	$\frac{e^{x_i}}{\sum_{j=1}^J e^{x_j}} \quad \text{for } i = 1, \dots, J$
Relu	$f(x) = x^+ = \max(0, x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$
Tanh	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Swish	$\frac{x}{1 + e^{-x}}$

Table 1. Activation functions

The above activation functions give different results on the same input data size. Thus, selecting the best from the above will be beneficial to reach accuracy with less training time and the highest accuracy.

Sigmoid predicts maps the output into 0 or 1 which is generally the best function for binary classification problems like our case. Tanh is similar to sigmoid but it maps output in the range -1 to 1 which is somewhat less accurate than sigmoid. The softmax function is a kind of sigmoid function but for multi-class classification. After training the model with the above activation function we get different training and validation accuracy which is shown below.

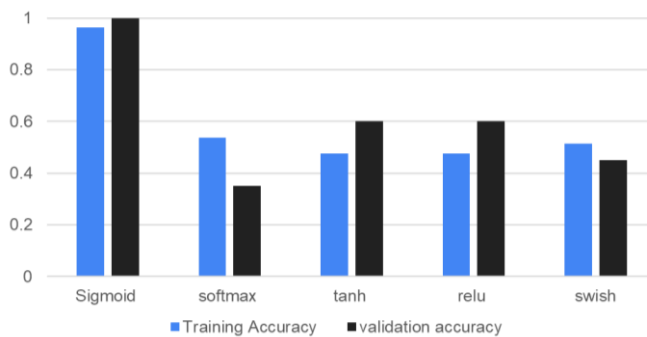


Fig.15: Activation Function V/s Accuracy

After training the model with different activation functions it can be clearly seen that the sigmoid activation function gives the best training and validation accuracy, followed by the tanh function. Thus, the sigmoid function is chosen to be for the highest training and validation accuracy.

#### D. Activation Function Vs Training Loss and validation loss in CNN model

Validation loss can be described as the value of an objective function or loss function of the model. After each epoch, the model updates its weights and the difference between actual labels vs predicated labels can be seen as the loss of the model.

Apart from accuracy, training loss also plays a significant role in selecting the activation function. Training loss describes how the model is performing on training data while validation loss is how well the model performs on new data.

So while deciding the activation function not only accuracy but losses are also taken into consideration. From the analysis, the sigmoid and SoftMax have almost negligible losses but the other functions tanh, relu and swish have considerable losses of more than 6 in number.

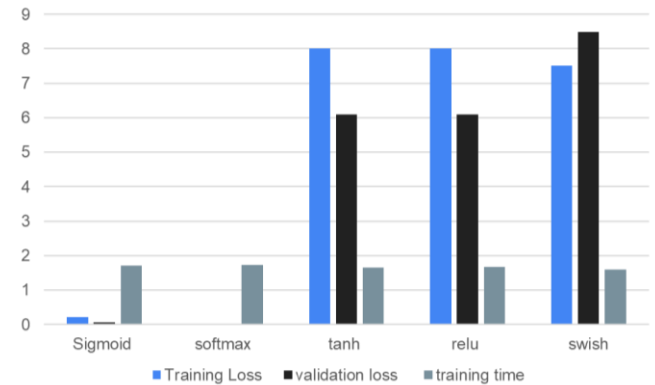


Fig.16: Activation function vs Training loss

Thus, sigmoid is the best choice with the highest training and validation accuracy and least loss accuracy. We have gained the insight that training losses vary, and they can be greater than 1 for certain activation functions as shown in the above chart. Of them, the least is for Sigmoid and hence it is used for further analysis. It is noteworthy that there is not a major change in training time with the change in the activation function as they are not directly affecting the training parameters of the model.

#### E. Image Size v/s training of model and its accuracy

Input size, which in this case is the image size, has an enormous impact on the accuracy and training time of the model. The larger the image size, the more data a model has to learn. As the image size increases, the number of pixels increases, which ultimately increases the number of parameters to train and, finally, the training time.

Another conclusion that can be drawn is that as the image size increases, the features are clearly visible and model accuracy increases to some extent. Moreover, larger image size leads to overfitting, as the model tries to learn too many things and is unable to generalize the data.

Image Size (X, Y, 3)	Number of Pixels
32,30 (original/4)	2880
64,60 (original/2)	11520
128,120 = original size	46080
256,240 (original×2)	184320
512,480(original×4)	737280

Table.2: Image size and Pixels

Therefore, it is important that the image size is optimum so that there is a good balance between training time and accuracy. The below graph shows the changes in image size and its impact on each epoch up to 5 epochs.

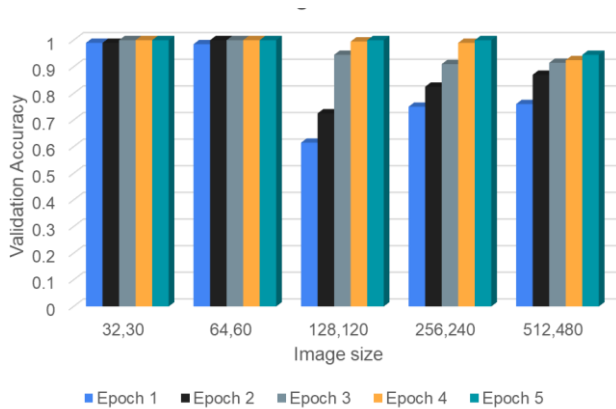


Fig. 16: Epoch Vs accuracy

As discussed above higher the image size, high the training time to reach a particular value as the model needs more time to understand a large image size compared to a small one.

So, here one can easily see that the image size of (32,20) has the highest accuracy in the first epoch while the image size of 512,480 attains the same accuracy after 5 epochs.

From the above graph and other analyses, the original image size gives the best results with a filter size of (3,3). The model was coded to resize the new image to match the input size required by our trained model so that any new image can be tested and predicted by the model.

#### F. Kernel size v/s accuracy of the model

Kernel size is also one of the key factors to decide while training a model. If the kernel size is too small it is difficult to capture features. Hence the kernel size, the more is the accuracy

of the model. Nonetheless, it is possible that if the kernel size is too high it will eventually make the model overfit.

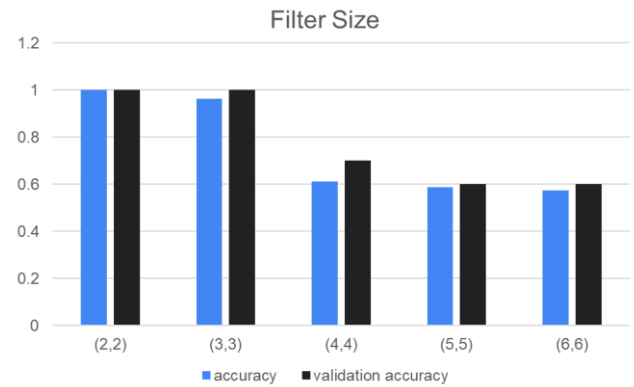


Fig.17: Kernel size v/s accuracy of the model

It is always advisable to do experimentation to find the right balance between kernel size and accuracy. and by doing so we find that a kernel size of 3, 3 gives the best accuracy while changing all the parameters of our model.

So, here one can easily see that the image size of (32,20) has the highest accuracy in the first epoch while the image size of (512,480) attains the same accuracy after 5 epochs.

#### G. Number of filters V/s accuracy of CNN model

The number of filters used in a convolutional neural network has a significant impact on its accuracy. Increasing the number of filters can improve accuracy, as the model learn more characteristics as we apply more and more filter, but it can also lead to overfitting. Therefore, it is recommended to experiment with different combinations of filter number and size through iteration to achieve the desired level of accuracy.

Increasing the number of filters can improve accuracy, as the model learn more characteristics as we apply more and more filter, but it can also lead to overfitting. Therefore, it is recommended to experiment with different combinations of filter number and size through iteration to achieve the desired level of accuracy

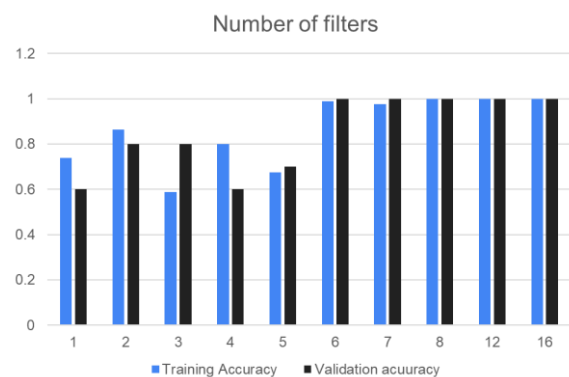


Fig. 18: No. of filters vs accuracy



During our experimentation with different numbers of filters in a convolutional neural network, we found that as we increased the number of filters, there was an increase in both training and validation accuracy. However, after reaching 8 filters, accuracy reached a peak and did not improve further. To ensure the best results, we decided to use 16 filters, as this number was on the safer side and had shown promising results.

#### H. Number of layers in CNN and its accuracy

Convolutional Neural Networks are composed of various layers, including convolutional, pooling, and fully connected layers, all of which can impact the overall performance of the model. To replicate our 100% accurate results with a learning rate of 0.1, we initially added one more conv2d and one more max pool layer but did not achieve our desired accuracy, as the model may not have been strong enough to predict the images. However, we were able to achieve maximum accuracy after adding an additional conv2d layer and one more max pool layer. These experiments showed that different combinations of learning rates and layer numbers can lead to desired outcomes, and we need to select the model with the highest accuracy and the least training time.

Hidden layers	Accuracy	Validation accuracy
Conv2D (3,3) Maxpooling (2,2) activation='relu'	0.5125	0.4500
Conv2D(3,3) Maxpooling(2,2) Conv2D(3,3) Maxpooling(2,2) activation='relu'	0.6625	0.5000
Conv2D(3,3) Maxpooling(2,2) Conv2D(3,3) Maxpooling(2,2) Conv2D(3,3) Maxpooling(2,2) activation='relu'	0.8625	1.0000

Table 4: Layers vs accuracy

So, here one can easily see that the image size of (32,20) has the highest accuracy in the first epoch while the image size of 512,480 attains the same accuracy after 5 epochs.

#### I. Comparison of our CNN model with the existing model

Below are some of the well-known models for binary classification for problems with desired results Yes/No, True/False, Male/Female etc.

- 1)Naive Bayes
- 2)Logistic Regression
- 3)K Nearest Neighbor
- 4)Random Forest
- 5)Support Vector Machine

When compared to the above models, the accuracy and loss of the present CNN model for the data size of 1000 images give the best prediction of the test set.

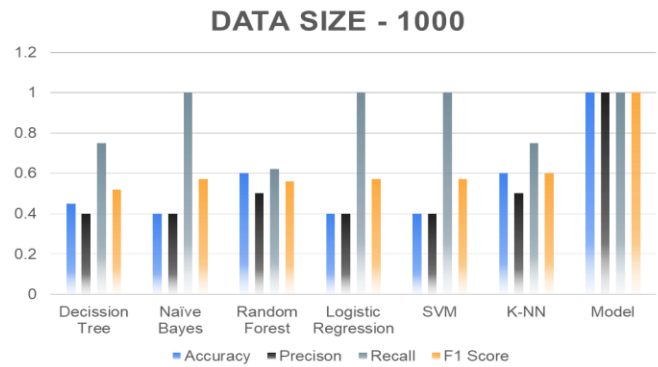


Fig. 19: Data size vs accuracy for various models

The above part compares the accuracy, precision, recall and F1 score of our CNN model to the existing machine-learning, and our model gave the best performance among all other models because of complex architecture than machine learning models

#### J. Training time for different models for the same number of images

Model	Training Time (milliseconds)
Naive Bayes	00.08
Logistic Regression	00.68
K Nearest Neighbor	00.10
Random Forest	00.81
Support Vector Machine	03.09
Decision tree	01.82
CNN with 1 conv 2d & 1 max pool (lr=0.001)	20:00

Table.5: Training time Vs accuracy

From the above table, it can be clearly seen that other machine learning models need less time to train for 1000 images compared to the CNN model because they have simpler algorithms. However, the CNN model has complex architecture and needs only to give the best accuracy so it's a plus point to use CNN instead of others so that it will predict the right class even if it takes more training time.

## VI. CONCLUSION

This project has helped to study the efficiencies of the model in classifying the images from the dataset. Various analyses were done by varying process parameters and using different classification models. The model was found to give the best results with the training set having 80% of images from the data set and using one convolution and one pooling layer by applying a 3\*3 filter size at a learning rate of 0.1. The results have shown better accuracy than existing supervised models and the accuracy of the model was best when using sigmoid as an activation function. In addition to that, initial trials have

proved that using a balanced dataset for training resulted in more accuracy compared to the biased dataset.

## REFERENCES

- [1] [1] N. S. Johnson et al., "Invited review: Machine learning for materials developments in metals additive manufacturing," *Addit. Manuf.*, vol. 36, Dec. 2020, doi: 10.1016/J.ADDMA.2020.101641.
- [2] [2] M. Valizadeh and S. J. Wolff, "Convolutional Neural Network applications in additive manufacturing: A review," *Adv. Ind. Manuf. Eng.*, vol. 4, p. 100072, May 2022, doi: 10.1016/J.AIME.2022.100072.
- [3] [3] M. A. Ansari, A. Crampton, R. Garrard, B. Cai, and M. Attallah, "A Convolutional Neural Network (CNN) classification to identify the presence of pores in powder bed fusion images," *Int. J. Adv. Manuf. Technol.*, vol. 120, no. 7–8, pp. 5133–5150, Jun. 2022, doi: 10.1007/S00170-022-08995-7/FIGURES/16.
- [4] [4] "Convolutional Neural Network. Learn Convolutional Neural Network from... | by dshahid380 | Towards Data Science." [https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529#:~:text=When we talk about computer,to the basic neural network. \(accessed Apr. 10, 2023\).](https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529#:~:text=When we talk about computer,to the basic neural network. (accessed Apr. 10, 2023).)
- [5] [5] "Accuracy, Precision, Recall & F1-Score - Python Examples - Data Analytics." [https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/ \(accessed Apr. 10, 2023\).](https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/ (accessed Apr. 10, 2023).)