# Can Machine Learning be an effective tool for handwriting digitization?

by

Afonso Ginjeira

A Dissertation

Submitted to the School of Computing, Engineering and Mathematics

Coventry University

BSc (Hons) Computer Science

6th December 2021

Supervisor: Bob Bird

SID: 9009826

## 6001CEM Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

### Statement of Copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialize products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information, please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

### Statement of Ethical Engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: Date: 06/12/2021

| | |
|---|---|
| First Name: | Afonso |
| Last Name: | Ginjeira |
| Student ID Number: | 9009826 |
| Ethics Application Number: | P130217 |
| 1st Supervisor Name: | Bob Bird |

## *Table of Contents*

# 1 List of Figures

# 2 Abstract

This report focuses on a project which evaluates the efficiency of Machine Learning (ML) Algorithms when predicting what character is depicted in an image of handwritten text.

The recognition and digitization of handwritten text, unlike printed text, is still a challenge for computers today, as many factors can impact the ability of the machine to make an accurate prediction. These primarily include letter slanting, multi-stroke characters and scanning errors. As such, data tends to require pre-processing techniques to make it more comprehensible to computers.

For this purpose, three machine learning classification models were applied to a dataset of images of handwritten alphanumeric characters. The models were trained using a dataset containing images of every English language character and the ten numerical digits. The models utilised during this project are classified as Convolutional Neural Networks (CNN). The prediction accuracy of all models was greater than 70%. The three machine learning models used are ResNet50, MobileNetV2 and Keras Sequential with the two formers utilising transfer learning, meaning they are pre-trained models available for public use. The models were trained with hundreds of images of individual alphanumeric characters, each labelled accordingly to what they represent. From the models tested, the pre-trained MobileNetV2 achieved the best results, with an accuracy of 93%. Both other models underperformed in comparison with ResNet50 achieving 89% accuracy and Keras Sequential achieving an accuracy of 72%.

Based on the results a conclusion was drawn that machine learning, especially when paired with transfer learning, can be an effective tool for handwritten character recognition.

Key-Words: Machine Learning, Transfer Learning, Handwritten Character Recognition, Recognition Model, Digitization

# 3 Acknowledgements

Throughout the progression of this project, I have received a lot of support from people I would now like to thank.

I would like to begin by thanking my supervisor, Bob Bird, for the initial support given to me when selecting a project idea.

Furthermore, I would like to thank the module leader, Peter Every, for being so approachable and available throughout the whole process.

I would also like to thank my family and friends who encouraged me during the entire process and are the main reason I was able to complete this project.

# 4 Introduction

The Technological Revolution has changed the way the world works in ways that were unimaginable just a few decades ago. More recently, the field of Machine Learning (ML) has, again, come to disrupt the way our machines work. As opposed to having to teach a computer how to perform certain tasks and recognize certain patterns and images, computers can now learn by themselves how to recognize patterns in data. With these advancements in computing, a lot of new functions for computer programs have emerged and are being implemented in all fields, like medicine, aviation, finance and transportation services.

"Machine Learning is a branch of Artificial Intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy." (IBM Cloud Education, 2020) As previously stated, machine learning depends on the recognition of patterns to make assumptions, and as computing power increases, so does the accuracy and speed at which these predictions can be made.

The work provided in this project aims to study the effectiveness of Machine Learning at predicting what alphanumeric character is represented in an image. Currently, the problem of printed character recognition can be seen as solved, but the recognition of handwritten characters is still an open problem and as such, new solutions are always welcome. This is due to the impossibility of standardization when it comes to handwriting, as there as many variables such as slope and the thickness of the stroke, as well as errors during the scanning process, in which the images can be negatively affected.

The effectiveness of these programs can have a big impact on different applications and uses. Handwritten character digitization can be useful for people with disabilities, impaired reading ability or even to digitize and archive old documents. This technology is of great use for archiving purposes, as when a document is digitized the letters and consequently the words can then be searched digitally, making the information instantly accessible.

Machine learning classification models can be created and tested to recognise specific patterns, in this case, alphanumeric characters, but there are a few issues that need to be kept in mind as the project progresses. The data utilised for training purposes can contain images that have been wrongfully labelled, making the model less effective or even invalid.

To solve the proposed question, since different algorithms can provide different results, multiple classification models will be created, and their performance analysed, two of them making use of transfer learning technology. All models will be trained using a dataset retrieved from the internet which contains hundreds of images of isolated handwritten characters. The accuracy of these models' predictions will then be compared to see which one would be most suited for the intended purpose.

This report also contains a literature review into the topic of handwritten character digitization, as well as classification algorithms and Convolutional Neural Networks specifically, which all proved to be useful topics to research during the process.

# 5 Literature Review

## 5.1. Machine Learning

In the field of computer science, Machine Learning has been a major topic of conversation, but what is machine learning?

Machine Learning has been defined by IBM as “a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.” (IBM Cloud Education, 2020)

Another key issue is the quantity and quality of input data. Machine learning algorithms are highly “data-hungry” often requiring millions of observations to reach acceptable performance levels. (Obermeyer & Emanuel, 2016) In addition, biases in data collection can substantially affect both performance and generalizability.

## 5.2 Recognition Algorithms

The problem of pattern recognition and image categorization has been around for a long time and a lot of effort has gone into trying to solve it. Many different algorithms and ML models have been developed and their speed and accuracy have only gotten greater. Despite this development, a lot of problems are still considered open and unresolved. Additionally, as with most subjects in computer science, all problems could benefit from an increase in speed and efficiency.

### 5.2.1. Handwritten Character Recognition

Although many of the recognition and classification problems in ML have been solved, “The recognition of unconstrained handwritten text is a challenging pattern recognition problem. While the recognition of machine printed text can be considered solved for Latin languages this is not the case for handwritten text.” (Pesch, Hamdani, Forster, & Ney, 2012) “The most obvious difference is the number of classes that can be up to 52, depending if uppercase (A–Z) and lowercase (a–z) characters are distinguished from each other. Consequently, there is a larger number of ambiguous alphabetic characters other than numerals.” (Zafar, Mohamad, & Othman, 2007) The issue is further complicated by the existence of hundreds of different writing styles and patterns for every single character, such as cursive representation, and letters that are disconnected or multi-stroke.

“High inter-class variability poses a challenge to statistical text recognition systems forcing the development of robust features, robust classifiers or data preprocessing steps. Since it is difficult to design robust features for a high number of different writing styles and it is even more difficult to gain performance in the classification process that has already been lost in the features, data preprocessing and normalization are one of the most convenient methods to improve recognition performance.” (Pesch, Hamdani, Forster, & Ney, 2012) Other issues include image noise and scanning errors.

## 5.2.2. Data Pre-Processing

As mentioned above, handwriting contains a large amount of variability, even within each separate class and as such, data pre-processing can be a very useful tool to boost prediction accuracy.

For the proposed problem of handwritten character digitization, there are a few techniques that can be used. The first step to be taken in character recognition should be to convert all images to grey-scale as this immediately removes layers of difficulty from the problem.

“The pre-processing is one of the most crucial steps in Optical Character Recognition, which includes smoothing, binarization, skew detection, slant correction, thinning or skeletonization, filtering, base line detection, etc. The pre-processing is a initial step which converts the input data in a format which can be processed more efficiently with ease.” (Sharma, Kaushik, & Gondhi, 2020)

Contrast Normalisation is one of the first techniques to be applied, assuming the images are grey-scale. It works by mapping the lightest pixels all to white and the darkest pixels all to black, leaving some in between for grey-scale. These percentages may vary but as the images being dealt with contain black handwriting on a white background normally 70-90% of the lightest pixels would be mapped to white, and 5-20% of the darkest ones would be mapped to black, leaving what's left as grey-scale.

“To reduce noise in the text images we apply a median filter in a window of 3×3 pixels centered on the current pixel to all images. (..) Although a visual inspection of the image before and after median filtering shows hardly any difference, experimental results (...) show that preprocessing steps such as slant correction benefit from the noise removal.” (Pesch, Hamdani, Forster, & Ney, 2012)

For slant removal, “The text image is sheared for a discrete number of angles (typically from -45 to 45 degrees with 1 degree step) with respect to the vertical direction. Assuming that the slant angle for the original image is $\alpha$, the sheared image with $-\alpha$ angle will produce a non slanted text. This happens if $|\alpha|$ is less or equal to maximum angle used in the shearing process (usually 45 degrees). To discriminate this deslanted image from the rest of sheared images, the vertical projection profile for every sheared image and the corresponding vertical projection histogram are built. The vertical projection histogram which presents the maximum variability is chosen as corresponding to the deslanted text.” (Fukushima, 1980)

## 5.3. Convolutional Neural Networks

Convolutional Neural Networks are a special type of multi-layer neural network. Kunihiko Fukushima introduced recognition which is a multi-layered neural network capable of recognizing visual patterns hierarchically through learning. This network is considered as the theoretical inspiration for CNN. (Fukushima, 1980)

“A typical CNN is composed of single or multiple blocks of convolution and sub-sampling layers, after that one or more fully connected layers and an output layer.” (Sultana, Sufian, & Dutta, 2018)

“In a large image, we take a small section and pass it through all the points in the large image (Input). While passing at any point we convolve them into a single position (Output). Each small section of the image that passes over the large image is called filter (Kernel). The filters are later configured based on the back propagation technique.” (Sultana, Sufian, & Dutta, 2018)

The pooling layer consists of downsizing the image. It takes the output of the convolution layer as input and sub-samples it. It works by taking the largest value in a region of pixels (e.g. 3x3) and using that in the output as the value for a single pixel.

The last piece of a CNN is the Fully Connected layer which takes the output of all the neurons in the previous layer as input and performs the operation with the current layers neurons.

# 6 Methodology

## 6.1 Dataset

The dataset selected for this project contains a total of 3410 images of the characters in the English alphabet, both upper (A-Z) and lowercase(a-z), as well as the 10 numerical digits (0-9) and was obtained from Kaggle.com which is a database containing various open datasets. This dataset seemed to be the best as the images were clear and it contained quite a few examples of each character. However, upon further inspection, most of the images within each of the represented classes seemed to be lacking variation as they are visually very similar. The data is provided in a folder named “IMG” containing 62 different folders, one for each class of both letters and numbers.

## 6.2 Data Pre-Processing

Before any machine learning techniques can be applied to the desired dataset, the former needs to be prepared for processing. Data splitting is a process applied to datasets before they can be used to train CNN models. In this case, the dataset was split into two distinct datasets, a training dataset and a validation dataset. The training dataset is composed of 80% of samples and is the dataset that is utilised to, as the name implies, actually train the CNN, whilst the validation dataset utilises the remaining 20% to minimise error and improve accuracy. Simultaneously to the creation of these subsets, both an image and batch size were defined, corresponding to (180, 180) and 32 respectively. The image size is defined so that all images passed to the model have the same shape. When training a CNN, batch size is an important attribute, as it defines how many training examples are propagated through the neural network at once. By training the model using a batch, the processing of the data can be made faster however, this entails the added risk of reduced accuracy.

## 6.3 Framework and API

All of the models in this project were created in the python language and take advantage of TensorFlow, as well as the Keras API. “TensorFlow is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming.” (About Keras, n.d) The use of TensorFlow provides multiple advantages, such as high efficiency when processing tensor operation on a GPU, CPU or TPU, it can be used to scale computation to many devices, such as GPU clusters and it can export programs to external servers and devices, (About Keras, n.d) “Keras API is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.” (About Keras, n.d) The combination of both the TensorFlow platform and Keras API allows for a simple, yet powerful and efficient way to define, train and test machine learning models, as well as to visualise the data resulting from this process.

## 6.4 Model Creation

To attempt to answer the question posed at the beginning of this project, machine learning models had to be created. The last two models took advantage of transfer learning. "Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains. In this way, the dependence on a large number of target-domain data can be reduced for constructing target learners." (Zhuang, et al., 2021) In essence, transfer learning takes advantage of machine learning models which have been pre-trained with very large datasets. This technique can be responsible for a dramatic increase in prediction accuracy.

## i. Sequential Model

The first model was created as a base model, using TensorFlow's Keras Sequential. A Sequential model is one of the most basic models that can be created as it consists of a stack of layers, each with a single input tensor and a single output tensor.

The model created consists of three pooling layers and three convolutional layers, which are laid out in an alternating fashion. The model also consists of a dense layer, which feeds all outputs from the previous layer to all of its neurons, each providing an output. Once all the layers in a model have been defined, the model can be compiled utilizing a built-in function and the 'accuracy' attribute, which allows for future inspection of the accuracy of the model throughout each of its training stages.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0

 rescaling_2 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_3 (Conv2D)           (None, 180, 180, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16)       0
 2D)

 conv2d_4 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 dropout (Dropout)           (None, 22, 22, 64)        0

 flatten_1 (Flatten)         (None, 30976)             0

 dense_2 (Dense)             (None, 128)               3965056

 dense_3 (Dense)             (None, 62)                7998

=================================================================
Total params: 3,996,638
Trainable params: 3,996,638
Non-trainable params: 0
```

*Figure 1 - Base Sequential Model Summary*

### ii. MobileNetV2

The second model to be created for this project is one that makes use of transfer learning. It takes advantage of the pre-trained MobileNetV2 model. MobileNetV2 is a CNN architecture based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. (Sandler, Howard, Zhu, Zhmoginov, & Chen, n.d)

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 1280)             2257984
 ional)

 flatten (Flatten)           (None, 1280)              0

 dense (Dense)               (None, 512)               655872

 dense_1 (Dense)             (None, 62)                31806

=================================================================
Total params: 2,945,662
Trainable params: 687,678
Non-trainable params: 2,257,984
_________________________________________________________________
```

*Figure 2 - MobileNetV2 Model Summary*

### iii. ResNet50

The final model to be developed took advantage of the award-winning pre-trained ResNet50 model, created by Google. In addition to the standard ResNet50 model, a Dense layer was added with 512 neurons that will be utilised to make the predictions.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 62)                31806

=================================================================
Total params: 24,668,606
Trainable params: 1,080,894
Non-trainable params: 23,587,712
_________________________________________________________________
```

*Figure 3 – ResNet50 Model Summary*

## 6.5 Model Methods

### 6.5.1 Feature Extraction

“Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.” (DeepAI, n.d.) In essence, the top layer of the pre-trained CNN is very specific to the original dataset that trained it, which makes it less relevant to the current problem. Feature extraction calls for the addition of a new classifier, which will be trained from scratch to the pre-trained model, making it more pertinent. To do this, the original top layer must be frozen, so that it does not get processed when the modified model is trained and evaluated.

### 6.5.2 Fine-Tuning

For the benefit of added accuracy and efficiency, fine-tuning techniques are then applied to both ResNet50 and MobileNetV2 models. During the “Feature Extraction” section, it was explained that for the benefit of the new models the top layers of the pre-trained model had to be frozen before it could be trained on the selected dataset, the one containing images of handwritten characters. Once the model, with the added prediction layer is trained with the new dataset and, since the classification head has been trained, the previous layers can be unfrozen. Next, the newly unfrozen layers of the base model and the newly trained classifier are jointly trained. This is what permits the “fine-tuning” in the higher-order feature representations in the base model, making them more relevant to the current problem. (TensorFlow.com, 2021)

### 6.5.3 Overfitting

Overfitting is a phenomenon in machine learning that can happen when the model does not generalise well when presented with unseen data. This can be observed when running the model, if after several epochs, one epoch being a full iteration over samples, the model accuracy begins to stagnate or even decrease. In an attempt to fix this problem, two techniques were used, Data Augmentation and Dropout.

### i. Data Augmentation

Data augmentation (DA) is a technique that can be used to artificially expand the size of the training set by creating modified data from the existing one. (Lyashenko, 2021) Generally larger datasets provide the key to developing effective and accurate ML models, as they can train said models with abundant information and a lot of variation. If a model's performance is to be enhanced, then DA is a great tool to use as it requires no further data to be collected or added to the dataset, it simply creates new, slightly altered data from the one already available.

### ii. Dropout

Dropout is another technique that attempts to increase the model’s performance. In essence, dropout means that a few outputs during the CNN training are ignored or dropped, which in effect makes the layer have a different number of nodes than it does. This means that during training, nodes are, randomly, forced to take on more or less responsibility, in an attempt to avoid situations where layers might become “complacent” and adapt to fixing the previous layers errors, creating overall a more robust, efficient and effective system.

# 7 Results and Evaluation

## 7.1 Non-transfer Learning Model

In the first stage of training the model, before applying the dropout layer and data augmentation techniques and after training of 10 epochs, the mean training accuracy of the model was 32.67%, with the maximum achieved being 66.02% and the mean validation accuracy 21.27%, with a single maximum of 36.66%. The mean validation loss was 323.43%, with a minimum of 263.83% and the mean training loss was 274.76%, with a minimum of 132.83%. The training of this model took approximately 10 minutes.

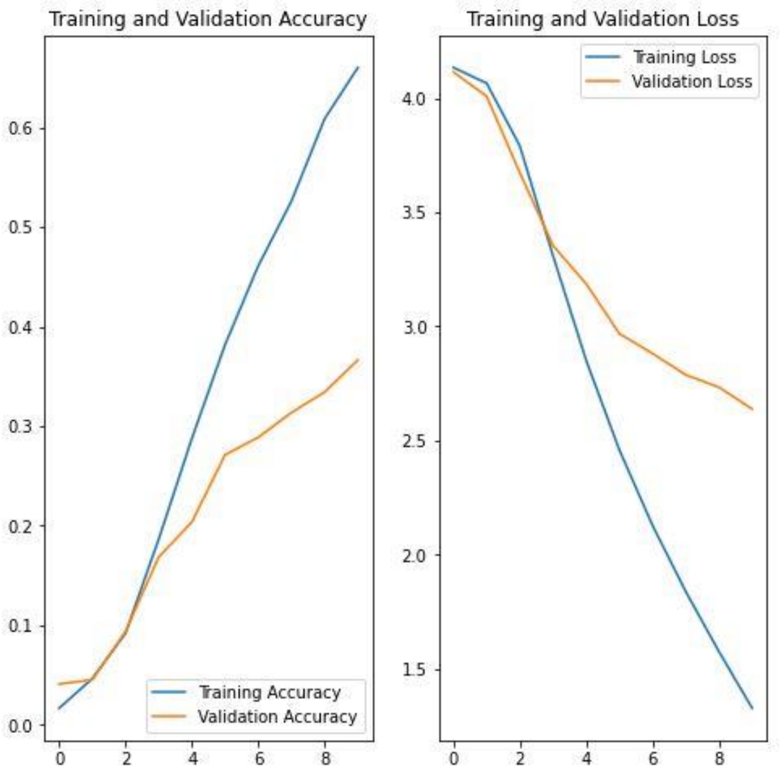


*Figure 4 - Non-Transfer Learning Accuracy and Loss before Data Augmentation and Dropout*

Once techniques to alleviate overfitting had been done and this time with the training being during 15 epochs, the mean training accuracy reached 41.67%, with a maximum of 61.91% and the mean validation accuracy was 40.08% with a maximum of 54.4%. The mean validation loss was 230.57%, with a minimum of 170.13% and the mean training loss was 219.66% with a minimum of 128.47%. The training for this model took approximately 16 minutes.

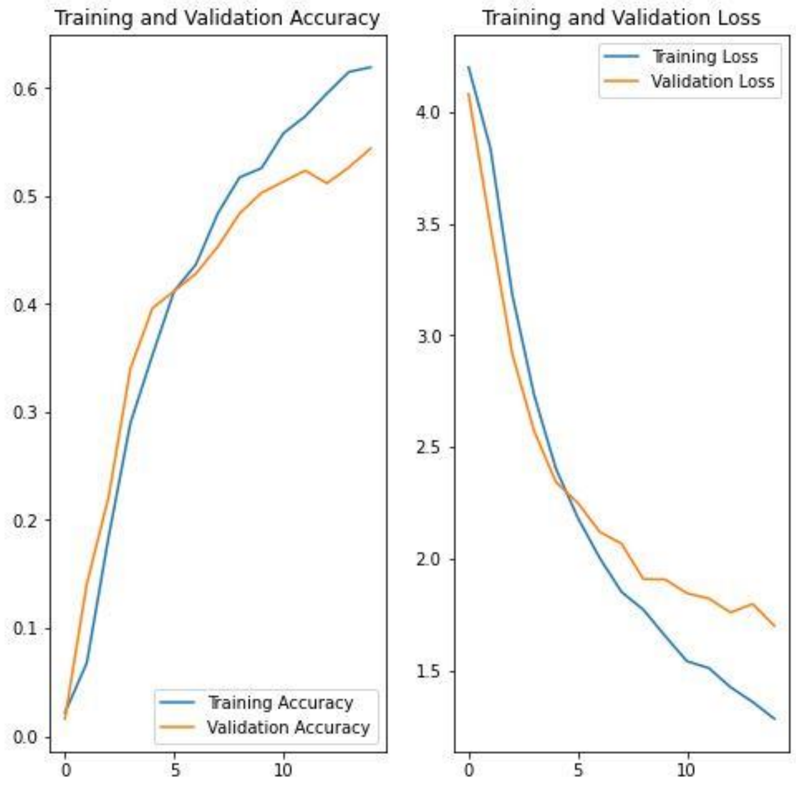


*Figure 5 - Non-Transfer Learning Accuracy and Loss after Data Augmentation and Dropout*

**Base Model Complete Results**

| Aa Stage / Attribute | # Pre Fine-Tuning | # Post Fine-Tuning | Σ Difference |
|---|---|---|---|
| Mean Traning Accuracy | 32.67% | 41.67% | 0.09 |
| Top Training Accuracy | 66.02% | 61.91% | -0.0411 |
| Mean Validation Accuracy | 21.27% | 40.08% | 0.1881 |
| Top Validation Accuracy | 36.66% | 54.4% | 0.1774 |
| Mean Training Loss | 274.76% | 219.66% | -0.551 |
| Minimum Training Loss | 132.83% | 128.47% | -0.0436 |
| Mean Validation Loss | 323.43% | 230.57% | -0.9286 |
| Minimum Validation Loss | 263.83% | 170.13% | -0.937 |

*Figure 6– Base Model Complete Results Table*

As can be inferred by the information above, the use of Data Augmentation and a Dropout layer, resulted in a decrease of 4.11% in the models training accuracy, but in a 17.74% increase in its validation accuracy. In addition, both validation and training losses also decreased in value by 93.7% and 4.36%, respectfully.

The testing accuracy for this model was evaluated and it achieved 71.99% accuracy with a 94.49% loss.

## 7.2 ResNet50

As with the previous model, pre and post-fine-tuning results were gathered, and in this instance, before fine-tuning, the model achieved a mean training accuracy of 67.13% with the last epoch achieving 83.61% accuracy and a mean validation accuracy of 60.11% with a maximum of 68.48% accuracy. The mean validation loss was registered at 149.87% with a low of 105.27% and the mean training loss was registered at 139.94% with a low of 68.87%. The training for this model took approximately 43 minutes.

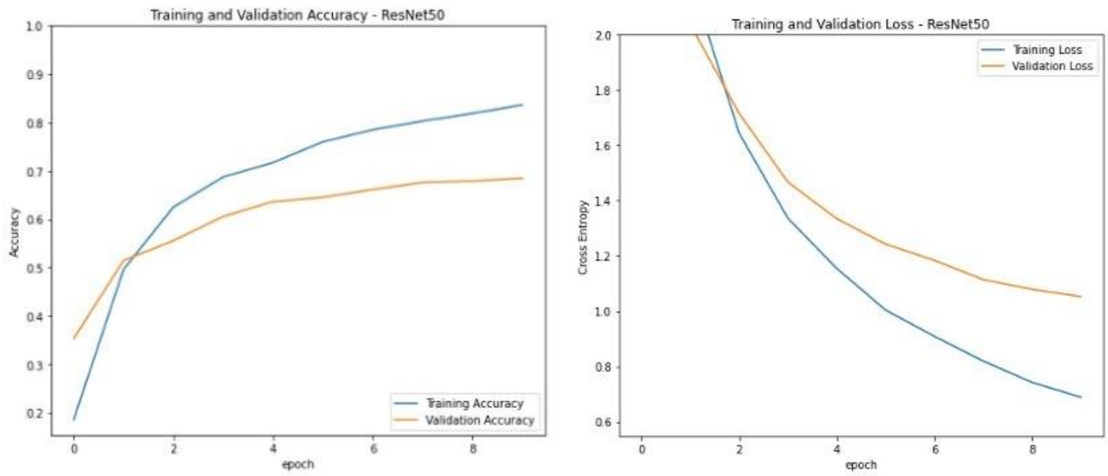


*Figure 7 - ResNet50 Accuracy and Loss Pre Fine-Tuning*

After fine-tuning the mean training accuracy was 88.71%, recording a maximum of 92.23% and the mean validation accuracy was 71.08% with a maximum of 73.46%. While the mean training loss was 46.17% with a minimum of 32.62% and the mean validation loss was 93.87% with a minimum of 85.19%. This model took approximately 48 minutes to train.

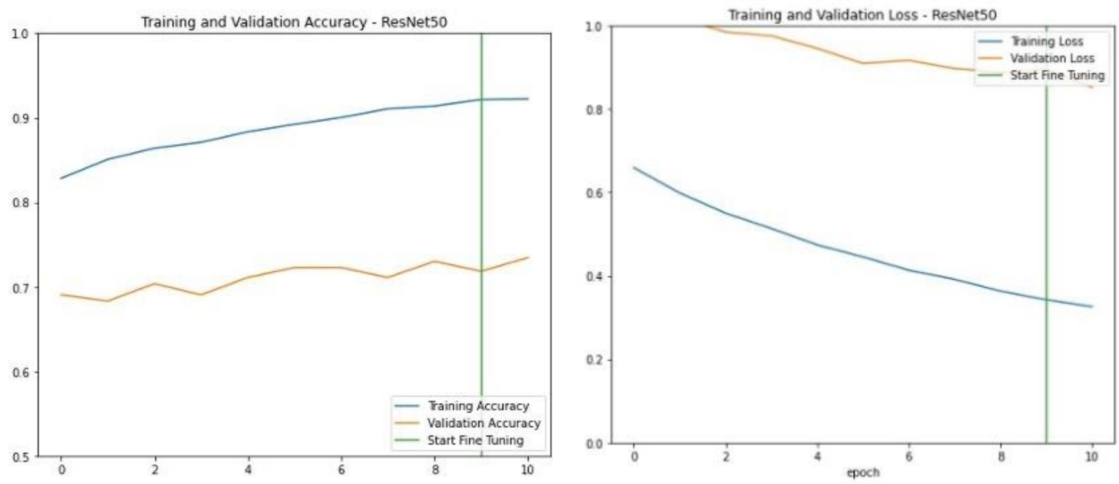


*Figure 8 - ResNet50 Accuracy and Loss Post Fine-Tuning*

**ResNet50 Complete Results**

| Aa Stage / Attribute | # Pre Fine-Tuning | # Post Fine-Tuning | Σ Difference |
|---|---|---|---|
| Mean Traning Accuracy | 67.13% | 88.71% | 0.2158 |
| Top Training Accuracy | 83.61% | 92.23% | 0.0862 |
| Mean Validation Accuracy | 60.11% | 71.08% | 0.1097 |
| Top Validation Accuracy | 68.48% | 73.46% | 0.0498 |
| Mean Training Loss | 139.94% | 46.17% | -0.9377 |
| Minimum Training Loss | 68.87% | 32.62% | -0.3625 |
| Mean Validation Loss | 149.87% | 93.87% | -0.56 |
| Minimum Validation Loss | 105.27% | 85.19% | -0.2008 |

*Figure 9 - ResNet50 Model Complete Results*

As can be observed in the information above, the use of fine-tuning contributed to an increase of 8.62% for training accuracy and 4.98% for validation accuracy and a decrease of 36.25% for training loss and 20.08% for validation loss. Meaning fine-tuning proved to be an effective method at enhancing the model's performance.

The training accuracy for this model was recorded at 89.12% with a loss of 41.07%.

## 7.3 MobileNetV2

Following the same method as both previous models, the accuracy and loss for the MobileNetV2 transfer learning model were recorded. The pre-fine-tuning values are as follows. The mean training accuracy recorded was 70.03% with a maximum of 87.50% and the mean validation accuracy was 66.99% with a maximum of 75.81%. Mean training loss was recorded at 131.25% a minimum of 53.38% and mean validation loss was 133.59% with a minimum of 85.47%. Pre Fine-Tuning training took approximately 13 minutes.

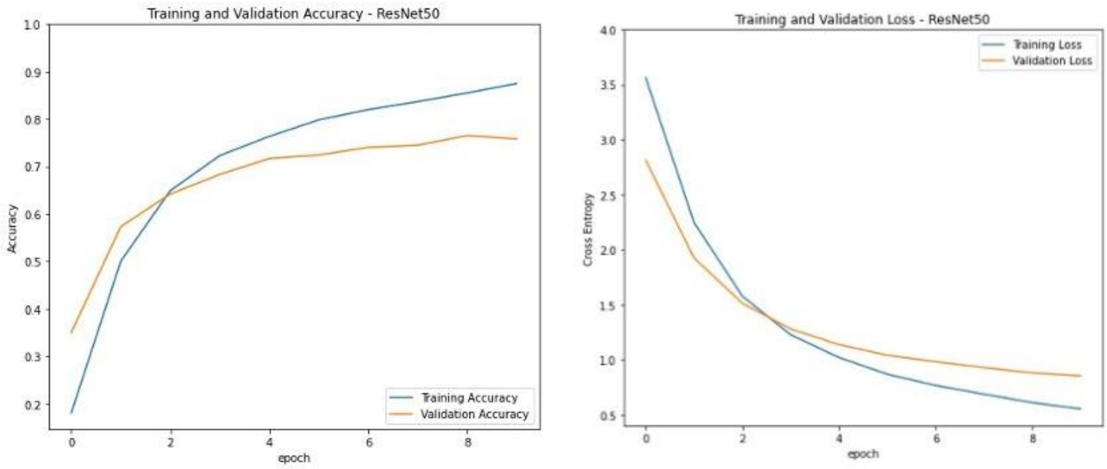


*Figure 10 - MobileNetV2 Accuracy and Loss Pre Fine-Tuning*

After fine-tuning had been applied, mean training accuracy was 92.93% with a maximum of 96.44% and mean validation accuracy was 77.91% with a maximum of 78.89%. The mean training loss was 33.64% with a minimum of 20.79% and the mean validation loss was 74.29% with a minimum of 69.92%. Post-fine-tuning training took approximately 13 minutes.

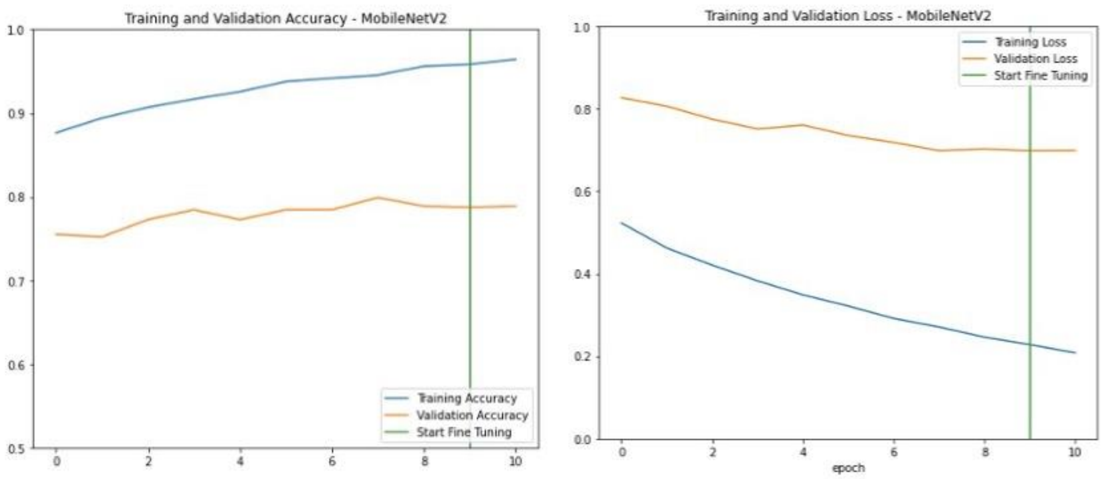


*Figure 11 - MobileNetV2 Accuracy and Loss Post Fine-Tuning*

**MobileNetV2 Complete Results**

| Stage / Attribute | Pre Fine-Tuning | Post Fine-Tuning | Difference |
|---|---|---|---|
| Mean Traning Accuracy | 70.03% | 92.93% | 0.229 |
| Top Training Accuracy | 87.5% | 96.44% | 0.0894 |
| Mean Validation Accuracy | 66.99% | 77.91% | 0.1092 |
| Top Validation Accuracy | 75.81% | 78.89% | 0.0308 |
| Mean Training Loss | 131.25% | 33.64% | -0.9761 |
| Minimum Training Loss | 53.38% | 20.79% | -0.3259 |
| Mean Validation Loss | 133.59% | 74.29% | -0.593 |
| Minimum Validation Loss | 85.47% | 69.92% | -0.1555 |

*Figure 12 - MobileNetV2 Model Complete Results*

As can be observed in the information above, the use of fine-tuning contributed to an increase of 8.94% for training accuracy and 3.08% for validation accuracy and a decrease of 32.59% for training loss and 15.55% for validation loss. Meaning that once more fine-tuning proved to be an effective method at enhancing the model's performance.

Once the model was evaluated, the testing accuracy was 93.22% with a loss of 29.40%.

## 7.4 Result Evaluation

| Model / Attribute | Training Accuracy | Validation Accuracy | Testing Accuracy | Training Loss | Validation Loss | Testing Loss |
|---|---|---|---|---|---|---|
| Base Model | 61.91% | 54.44% | 71.99% | 128.47% | 170.13% | 94.49% |
| ResNet50 | 92.23% | 73.56% | 89.12% | 32.62% | 85.19% | 41.07% |
| MobileNetV2 | 96.44% | 78.89% | 93.22% | 20.79% | 69.92% | 29.40% |

Table 1 - Results for all models

The above table brings all of the important values obtained so that a comparison can be made and consequently a conclusion can be drawn. As can be observed the MobileNetV2 pre-trained model is the one that provided the highest testing accuracy (93.22%) and the lowest testing loss (29.40%). Additionally, it presented the highest validation and training accuracy, as well as the lowest validation and training loss.

# 8 Discussion

The main objective of this project was to answer the question of machine learning effectiveness when it comes to handwritten alphanumeric character recognition and categorization. With the use of the TensorFlow platform and Keras API three Convolutional Neural Network machine learning models were created and trained on a data set containing images of all 62 alphanumeric characters in the English language. All models accomplished a minimum of 70% accuracy when predicting which handwritten character was depicted on an image. Two of the models made use of transfer learning technology, which in essence allows for a model which has been previously trained and tested, and that has a proven high accuracy rate, to be adapted and utilised for the needs of new machine learning problems and datasets. This proved to be an effective method at boosting the accuracy of both models as well as decreasing their loss. MobileNetV2, one of the transfer learning models, reached the highest prediction accuracy of 93.22%, making it the best performer out of the three and the most suited for the problem at hand.

Despite the ResNet50 model not being the most effective model out of the three, for the task at hand, it still produced an accuracy rate of 89.12%, which is only 4.1% less than MobileNetV2. Despite the gain in accuracy from ResNet50 to Mobilenet being in the single digits, meaning they are comparable, there is a larger disparity between their loss values, having an 11.67 percentage point difference.

The fact that MobileNetV2 was able to outperform ResNet50 was not expected from the beginning of the project, as ResNet50 is known to be one of the best performing ML models available. According to the Keras website, the top accuracy for ResNet50 is 92.10%, whilst this value for MobileNetV2 is 90.10%. The values obtained during the project for both these models are a good example of how fine-tuning is an effective tool for boosting prediction models accuracy.

When observing the measured metrics of accuracy and loss in the first model, it can be inferred that the model is not fit for purpose, as it produces quite a low prediction accuracy and a very high loss rate. Once data augmentation was applied and a dropout layer was added, there was a significant boost in its accuracy, and a large decrease in loss. However, despite these improvements, the values are not such that the model could be used. It should be considered that using more or better data augmentation techniques could have helped the model's accuracy, but more importantly, the implementation of these methods could have led to the lowering of the loss.

One problem that should be addressed is the fact that in the Resnet50 section of this report, both the training and validation loss graphs have been plotted, without taking into account the high values that these presented, resulting in the plotted functions that represent this data actually breaking the scale and not being displayed properly. This issue could not be resolved, as the generation of a new graph requires the model to be trained again for the values of each attribute to be re-generated. However, as with any computer program, and especially ML-based software, the results are not always the same and contain some variation, so the graphs that were produced after this mistake was spotted do not represent the exact data this report talks about.

Something else that needs to be noted is the fact that this entire project, including the training of the models, was performed on a laptop with an Nvidia Geforce 940MX graphics card and an intel i5 mobile processor. If the same tasks were to be undertaken in a much more powerful computer, the results would vary, mainly in the time it takes to process each epoch.

When discussing the limitations of this project, the dataset that was chosen to train the data has to be addressed. This dataset “contains 62 different classes with 55 images in each class” (Dave, 2021). Although the data proved to be good at training the prediction models, ideally the dataset would have contained a much larger number of samples. Additionally, within each category, there is not much variance between characters, meaning the models might struggle to predict handwriting of a different nature. if the study were to be conducted again, a larger dataset would be chosen, containing images of multiple writing styles, where letters presented a much larger variety of slant, stroke thickness and proportions.

In future, if the work started in this project were to be continued, an expanded dataset that incorporates other languages could be considered. Although the English alphabet is the basis for a lot of languages, it might prove to be quite interesting to test the developed models with Russian or Cyrillic alphabets, or even introduce logographic scripts like Japanese and Korean.

# 9 Project Management

Just as important as the actual study conducted, during the development of any project, it is important to keep a log of all the tasks that need to be completed, as well as a timeline for project progression. For these purposes, the application “Notion” was selected as the best option, as the author already possessed previous experience with the platform and it is very accessible.

In the first stage of project management, a list was compiled, which encompassed all of the tasks necessary for the project. These tasks were then sorted in chronological order and labelled. Three different categories were created: Literature Review, Methodology and Report.

## 9.1 Timeline and Organization

For the initial Project Proposal, a primary research plan was developed, which predicted the evolution and development of the project during 13 weeks, and had time-slotted tasks. This was the proposed timeline.

Week 1 - Complete a preliminary literature review, and submit an ethics application.

Week 2 - Continue Literature Review.

Week 3 - Begin the search for datasets and image classification ML models.

Week 4 - Select dataset and format files and images in preparation for processing.

Week 5 - Begin development, pre-process data

Week 6 - 8 - Create ML models.

The process of creating the machine learning programs and fitting them to the data set is predicted to be the one that takes the longest and as such it has been allocated to several weeks.

Week 9 - Structure the final report and prepare code for the end of the project. (i.e add comments and formatting)

Week 10 - Request Feedback for the report and continue type-up.

Week 11 - Finish Literature Review and continue the report.

Week 12 - Finalise Report and request final feedback from supervisor.

Week 13 - Final adjustments and project submission.

As the project progressed it became clear that this list was no longer fit for purpose and as such, other methods for project management had to be developed.

To keep a record of the project timeline a Gantt Chart was created. Gantt Charts are “one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflect the start date, duration and end date of the activity." (Duke, 2021) this allows for the entirety of the project to be seen at a glance.

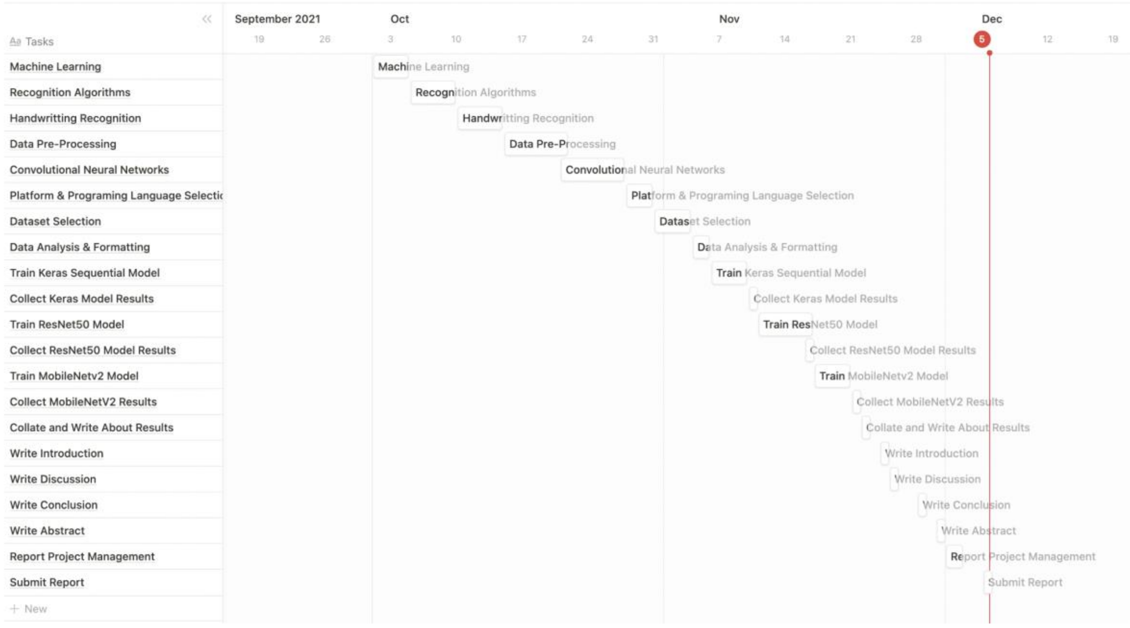


*Figure 13 - Gantt Chart*

As well as creating a Gantt Chart, a table was made that contained the same information, with additional attributes. In addition to the timescale of tasks, this view allows for the status of each task, "Completed", "In Progress" or "Not Started" to be visualised with a coloured label, as well as what category the task belongs to. The colouring of the labels is also important as it facilitates the visualisation of the progress of the project.

| Tasks | Type | Timeline | Sub-Section |
|---|---|---|---|
| Machine Learning | Completed | October 1, 2021 → October 4, 2021 | Literature Review |
| Recognition Algorithms | Completed | October 5, 2021 → October 9, 2021 | Literature Review |
| Handwritting Recognition | Completed | October 10, 2021 → October 14, 2021 | Literature Review |
| Data Pre-Processing | Completed | October 15, 2021 → October 21, 2021 | Literature Review |
| Convolutional Neural Networks | Completed | October 21, 2021 → October 27, 2021 | Literature Review |
| Platform & Programing Language Selecti | Completed | October 28, 2021 → October 30, 2021 | Methodolgy |
| Dataset Selection | Completed | October 31, 2021 → November 3, 2021 | Methodolgy |
| Data Analysis & Formatting | Completed | November 4, 2021 → November 5, 2021 | Methodolgy |
| Train Keras Sequential Model | Completed | November 6, 2021 → November 9, 2021 | Methodolgy |
| Collect Keras Model Results | Completed | November 10, 2021 | Methodolgy |
| Train ResNet50 Model | Completed | November 11, 2021 → November 16, 2021 | Methodolgy |
| Collect ResNet50 Model Results | Completed | November 16, 2021 | Methodolgy |
| Train MobileNetv2 Model | Completed | November 17, 2021 → November 20, 2021 | Methodolgy |
| Collect MobileNetV2 Results | Completed | November 21, 2021 | Methodolgy |
| Collate and Write About Results | Completed | November 22, 2021 | Report |
| Write Introduction | Completed | November 24, 2021 | Report |
| Write Discussion | Completed | November 25, 2021 | Report |
| Write Conclusion | Completed | November 28, 2021 | Report |
| Write Abstract | Completed | November 30, 2021 | Report |
| Report Project Management | Completed | December 1, 2021 → December 2, 2021 | Report |
| Submit Report | Not Started | December 5, 2021 | Report |
| + New | | | |

*Figure 14 - Time Management Table*

Furthermore, a Kanban board was developed. Kanban boards are a very popular project management tool that is part of Agile Methodology. Kanban boards essentially use different columns to monitor work progress and each task is a card within a column. Kanban boards can be very useful when the project is a team effort, as they are a great way for tasks to be distributed since normally the first column is a task dumping ground, where anyone can pick one to start completing. Once the selected task is moved to the "In Progress" column, the name of the engineer is attached to that card, meaning the board displays not only what is being done and at what stage of development it is at, but also who is responsible for each section of the project.

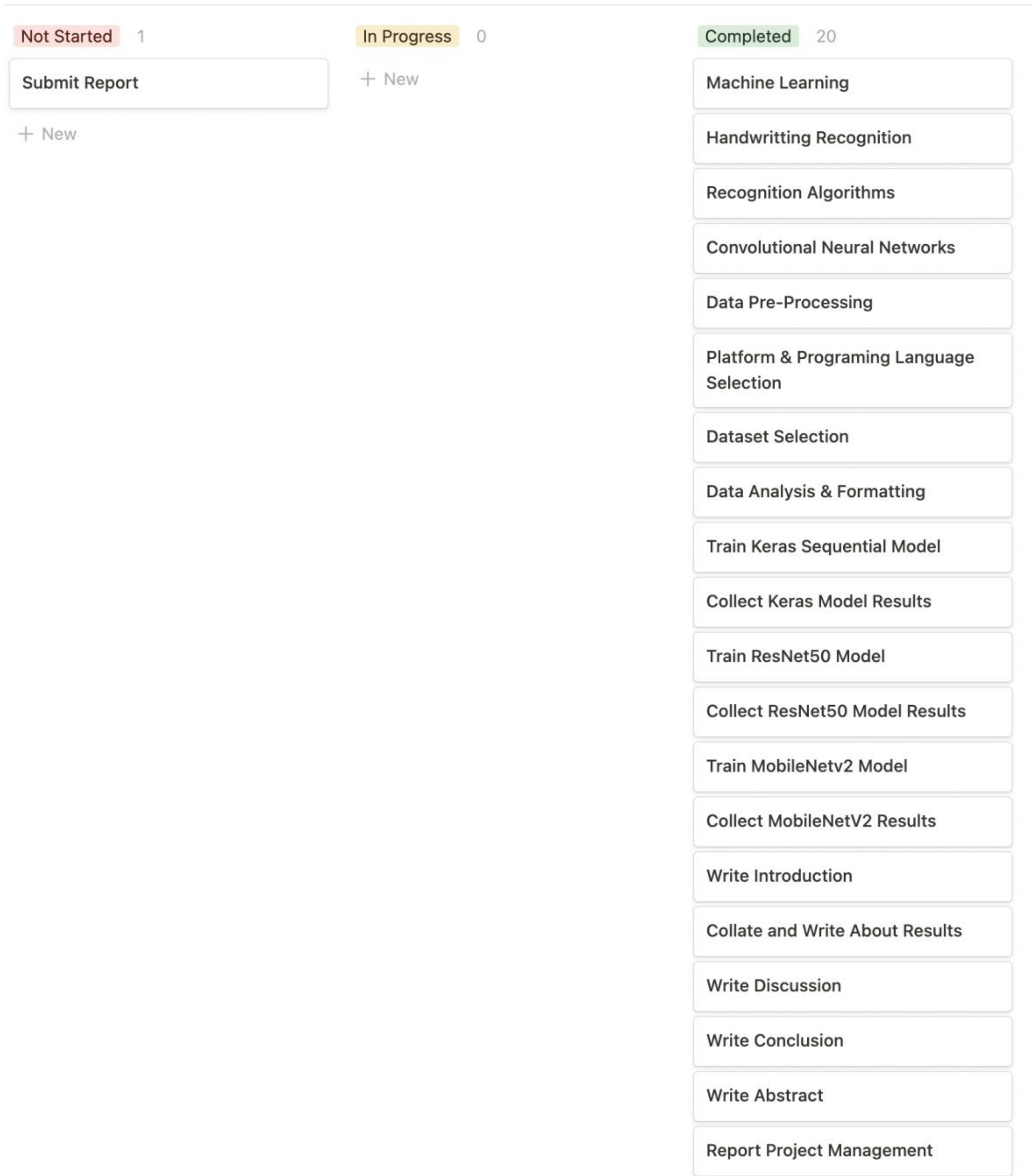


*Figure 15 - Kanban Board*

## 9.2 Ethical Considerations

When referring to the topics of ethical considerations, this project does not have any major issues, as it does not involve any sensitive data. The only issue that could arise with the data handled along the project would be if it had been used unlawfully or without a license, but this is not the case.

## 9.3 Reflection

Unfortunately due to unforeseen circumstances, this project faced quite a few difficulties. The project was initially meant to be done during the final semester of the Computer Science degree, but due to outside forces, it was deferred. The time management portion of the project could have been executed better, as can be seen above there is quite a difference between the proposed timescale and the final one.

Concerning communication with the project supervisor, there were not a lot of meetings or support. Before the project was deferred in the first semester of 2021, there were two meetings between the author and the supervisor. During these meetings, project topics were discussed, but not much else. Once the project did get deferred, unbeknownst to the author, the right to get support from a supervisor was withdrawn, so henceforth the rest of the work was completed without much external support. However, the leader of the module for which this paper has been developed did make himself available for the entire development.

# 10 Conclusion

From the findings in this study, it can be concluded that yes, machine learning is an effective tool for handwriting digitization and recognition. Although not all models created presented a high accuracy rate, and despite the limitations explained in the section above, analysis of the data produced during the project supports this conclusion. Having achieved a 93.22% accuracy rate, the MobilenetV2 transfer learning model proved to be an effective tool for the digitization and recognition of handwritten characters.

When analysing all of the results compiled for this report, an observation can be made that all three ML models achieve accuracies above 70% with the best performing model achieving 93% accuracy. Meaning it is viable to utilise MobileNetV2 or machine learning models more generally, to distinguish between handwritten characters.

Comparing the three models that were created, certain things can be inferred. The first is the fact that models benefitting from transfer learning show better performance and less error than those who do not take advantage of it. Secondly, data augmentation and fine-tuning techniques proved to be key at boosting a models performance, which is essential to ascertain the right predictions and for the study of ML models.

Based on the findings in this study, it is possible to say that machine learning models are key elements when it comes to recognising and predicting what a character in a picture is. In future, this research could be expanded, as previously stated, with the adoption of larger datasets with more categories, alphabets and writing systems, but additionally, the models created could be trained and utilised to recognise words as well. This would likely pose a few more issues that would have to be addressed.

If the recognition of individual letters and numbers is already a big task with a lot of variability within its scope, then the recognition of words would only exacerbate said complications. In essence, although words are composed of characters that our model already knows and recognises, the way in which these individual characters come together is what adds a new layer of complexity. Taking another look at a study, previously mentioned in the Literature Review section of this paper, conducted at Aachen University in Germany, it was stated that "several challenges in the recognition of unconstrained handwritten text is the high variability in the text appearance caused by effects such as image noise from scanning or errors from potential layout analysis but more important intrinsic properties such as differences in writing styles." (Pesch, Hamdani, Forster, & Ney, 2012)

In essence, the answer to the question posed as the research topic is yes, machine learning can be advantageous in handwritten text digitization and recognition. This study shows that out of three models, a transfer learning pre-trained model called MobileNetV2 was the best performer for the task.

# 11 Bibliography

*About Keras*. (n.d). Retrieved from Keras: https://keras.io/about/

Albawi, S., Mohammed, T. A., & Al-zawi, S. (2017). Understanding of a Convolutional Neural Network. *International Conference on Engineering and Technology.* Antalya.

Dave, D. (2021, February 24). *English Handwritten Characters*. Retrieved November 2021, from Kaggle.com: https://www.kaggle.com/dhruvildave/english-handwritten-characters-dataset

DeepAI. (n.d.). *Feature Extraction*. Retrieved November 2021, from DeepAI: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

Duke, R. (2021). *Welcome to Gantt.com*. Retrieved from Gantt.com: https://www.gantt.com/

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position . In *Biological Cybernetics.* Tokyo, Japan: Springer-Verlag.

IBM Cloud Education. (2020, July 15). *Machine Learning* . Retrieved from IBM: https://www.ibm.com/cloud/learn/machine-learning

Lyashenko, V. (2021, October 6th). *Data Augmentation in Python: Everything You Need to Know*. Retrieved November 2021, from NeptuneBlog: https://neptune.ai/blog/data-augmentation-in-python

Pastor, M., & Toselli, A. H. (2004). Projection Profile Based Algorithm for Slant Removal. *Image Analysis and Recognition: International Conference.* Porto, Portugal.

Pesch, H., Hamdani, M., Forster, j., & Ney, H. (2012). Analysis of Preprocessing Techniques for Latin Handwritting Recognition. *2012 International Conference on Frontiers in Handwritting Recognition.* Bari, Italy.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (n.d). MobileNetV2: Inverted Residuals and Linear Bottlenecks.

Sharma, R., Kaushik, B., & Gondhi, N. (2020). Character Recognition using Machine Learning and Deep Learning - A Survey. *2020 International Conference on Emerging Smart Computing and Informatics (ESCI).*

Sultana, F., Sufian, A., & Dutta, P. (2018). Advancements in Image Classification using Convolutional Neural Network. *Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN).*

TensorFlow.com. (2021, November 26). *Transfer learning and fine-tuning*. Retrieved November 2021, from TensorFlow: https://www.tensorflow.org/tutorials/images/transfer_learning

Zafar, M. F., Mohamad, D., & Othman, R. M. (2007). On-line Handwritten Character Recognition: An Implementation of Counterpropagation Neural Net. *International Journal of Computer, Electrical, Automation, Control and Information Engineering, 1*(10).