Master's thesis 2024

Digitizing Mt. Washington INPs Time Series Data

DENVER STRETESKY

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Space Earth Environment
Chalmers University of Technology
Gothenburg, Sweden 2024

Abstract

As technology in computing has advanced, it has opened many new tools to aid researchers. Among these tools is neural networks. These will be utilized to digitize an old handwritten dataset taken from Mt. Washington Observatory. This dataset spans over 20 years and has the potential to aid our understanding of ice nucleating particles and their effects. Three overarching steps were identified when digitizing handwritten data: pre-processing, classification, and post-processing. The first two will be focused on here as they are the most important to the digitization process. It was also discovered how the errors in the pre-processing and classification affect the final model accuracy.

Contents

1 Problem

There is a vast amount of handwritten datasets in various fields that have the potential to develop new understanding. In order to use these datasets, they must first be digitized. A dataset from Mt. Washington Observatory will be used to explore the techniques required for this task. The format of the data is PDF scans of the original handwritten tables and covers nearly 20 years of recordings. Due to the size, digitizing this by hand would take a prohibitive amount of time. A convolutional neural network (CNN) will be used to classify the characters.

There are a few potential errors that can occur when digitizing the data. A standard CNN will not be able to determine if a digit is meant to be an exponent. This will lead to a nonsensical measurement of concentration of INPs. Additionally, the CNN may have trouble differentiating between characters such as 1,l,/. If the writing includes cursive then the model will become more complex and more error prone. A few ways to combat these errors are to add a post-processing step to clean and correct the characters. Another option is to develop separate models for each column of the data. This would allow the models to more accurately classify the characters at the expense of training time.

2 Goal

The aim of the thesis is to digitize the Mt. Washington dataset. A secondary goal is to make a handwritten text recognition program that is easily expanded to other unseen data sets. By creating this program, it is hoped to answer how well machine learning techniques can be used to digitize historical handwritten documents and how the uncertainty propagates into the dataset.

3 Background

3.1 Ice Nucleating Particles

The Mt. Washington dataset consists of the concentration of Ice Nucleating Particles (INPs). The dataset also consists of other independent variables such as temperature and humidity that have an effect on the concentration of INPs. INPs are particles in the atmosphere that facilitate the formation of ice crystals by acting as a nucleation site. This is known as heterogeneous ice formation. These particles can come from different sources including dust, sea spray, fossil fuels and many others [1]. These particles are necessary for the formation of ice crystals in atmospheric temperatures above $-36°C$ [2]. Conversely, homogeneous ice formation is when ice crystals form without the aid of INPs. Homogeneous ice formation occurs when the temperature is below $-36°C$. This critical temperature where heterogeneous formation is required to where homogeneous ice formation is allowed is dependent on the size of the droplets [3]. Additionally, the rate at which homogeneous nucleation occurs increases as temperature decreases [3]. This thesis will focus on a dataset of heterogeneous ice nucleation rather than homogeneous nucleation.

INPs can have an impact on the climate through their role in cloud formation. Clouds can be made up of either water droplets, ice crystals or a mixture. These clouds have varying properties. One of the most significant differences is the reflective properties of the clouds. Clouds formed from water droplets are more reflective than those formed out of ice particles [4]. Therefore, more radiation from the sun will reach the surface of the earth through ice clouds than through water droplet clouds. Being

able to determine if a cloud is formed mainly of ice particles is important in modelling the climate. At a set temperature, increasing the number of INPs will allow ice clouds to form more easily. This could lead to climate change by allowing more of the sun's radiation to reach the surface of the earth.

3.2 Dataset

Mt. Washington is the tallest mountain in New Hampshire at 1,916.7 m. There is an observatory on the peak of the mountain that measures different weather parameters. From 1948 to 1970 the concentration of INPs was measured nearly every three hours. This represents the longest time series of this type. Notably, the record low is $-44°$C with the average mean low in the coldest month is $-36°$C. As the temperatures on the peak of the mountain are generally higher than those that allow for homogeneous formation ($-36°$C) of ice crystals, the crystals formed in the experiment are proportional to the number of INPs in the atmosphere. The series is largely handwritten in pre-made tables though some of the data is typed and some has hand drawn tables. The recorded data includes date/time, air temperature, wind direction and velocity, relative humidity, concentration of INPs and the visibility.

4 Literature Review

4.1 INPs

The 1948 to 1952 part of the dataset used in this thesis has already been analysed once by Schaefer [5]. The main discoveries were that there was a seasonality to the concentration of INPs with summer months having fewer particles than winter months. Additionally, it was observed that it was not uncommon to observe a "complete lack of ice nucleating particles at a temperature of $-20°$C". Another observation was that in just over 50% of observations, the concentration of the particles did not exceed 100 per m^3. Furthermore, it was determined that during times of high counts, the air had passed over the semiarid regions of the western United States. The highest recorded count was determined to be caused by a forest fire in Ontario. Overall, three atmospheric events were considered to have an effect on the concentrations: dust from arid areas of the continent, wildfires and volcanic dust. Schaefer also updated the analysis with data including 1953 [6]. In this year there was no decrease in concentration in the summer months. It is theorized that this is due to drought in the southwest. By analysing the rest of the time series, it will become clear if this is an anomaly or there is an underlying cycle to the data.

In order to aid the analysis, the concentration was classified as "low" $(0 - 1 \times 10^3 \mathrm{m}^{-3})$, "moderate" $(1 \times 10^3 - 5 \times 10^5 \mathrm{m}^{-3})$ and "high" $(5 \times 10^5 - 1 \times 10^7 \mathrm{m}^{-3})$. Then the percentages of each concentration is calculated for each month. This discretization of the data is supported by [7] where it is explained that the technique used to detect the nuclei is only accurate enough for qualitative data. By setting levels at "low", "moderate" and "high" the errors in counting are mitigated so the data and insights become more reliable. It is also suggested that meteoric dust could increase the number of INPs [7]. However, by 1985 it is suggested that extra-terrestrial sources do not have a significant impact on the number of INPs [8].

The authors [1] outline a six stage development for past, current and future analysis and modelling of INPs. By digitizing the Mt. Washington data set, this thesis will aid in stage 3 of model development by providing a long time series of observed concentrations. Furthermore, the models in the review include more types of INPs. For instance some models take into account sea spray or fossil fuel combustion. Neither of these were considered in [5].

4.2 Handwriting Recognition

Handwriting recognition can be split into two components, character extraction and character recognition. Extraction can be done by a bounding box algorithm using a scale space technique has successfully been applied to papers written by George Washington [9]. The text is written in cursive and has variable height. Additionally, the documents have various levels of degradation and the algorithm is still successful. This is significant due to the fact that the Mt. Washington dataset has pages that suffered from water damage or were scanned poorly. An assumption of the algorithm is that the space between words is larger than the space between letters. The bounding boxes are placed by first blurring the image horizontally to determine where the lines of the text are. Then using the extracted lines without any blurring, blur the text less than before creating blobs with white space between them. A challenging part the algorithm is to determine how much to blur by as too much will recognize too few words and too little may partition words.

The second component is the actual recognition of the characters themselves. Convolutional Neural Networks are one of the best handwritten character recognition techniques available [10]. It is possible to achieve a greater than 99% accuracy on the MNIST dataset which consists of 70000 images of handwritten digits with a CNN. The authors provide an overview of models that have performed better than 99% accuracy. Due to this, classification of digits is considered solved. A new dataset, EMNIST, includes both characters and digits. This creates a more challenging problem. Notably, CNNs still provide good accuracy (90%) on this dataset. An accuracy of 94% has been obtained using a CNN on the NIST dataset [11].

5 Methodology

There are three steps to digitize the dataset, pre-processing, character recognition and post-processing. The longest and most difficult part is the pre-processing as this step that culminates in extracting the characters from the datasheets. It is paramount to cleanly extract the characters as the recognition model will benefit from clean data. In the pre-processing, many techniques will be used to accurately extract the characters. Additionally, as the data spanned over 20 years, the way the data is recorded is not consistent. This means that the hyperparameters used to pre-process the sheets are different depending on the year the data was recorded. The differences in the years affect the first step of pre-processing but is most noticeable when identifying the columns. The sheets from 1948 to 1954 follow the same structure, 1955 is typed and split over two pages, 1957 to 1964 and 1969 and 1970 are the same style and 1968 was written on graph paper. Character recognition will use a neural network. The post-processing step will correct any systematic errors made by the character recognition and reconstruct the datasheet into a CSV file.

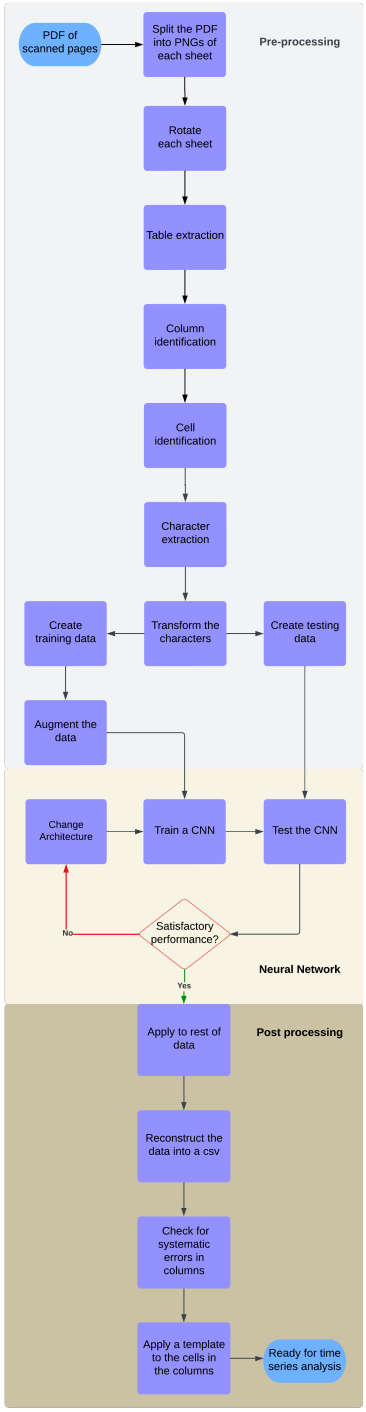
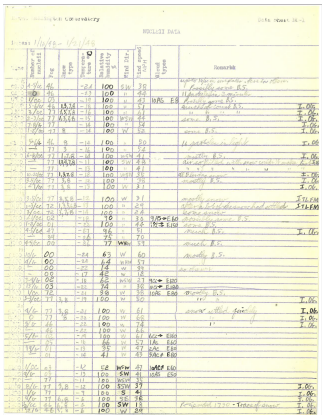


Figure 1: Steps taken to digitize the Mt. Washington data.

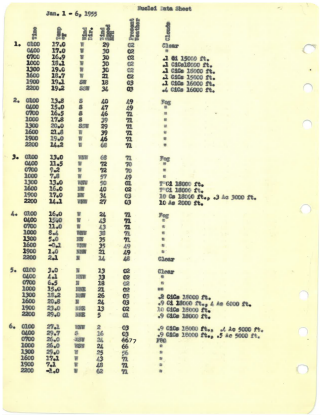
5.1 Page Types

Due to the different page types as shown in Figure 2, different parameters will need to be used in the pre-processing step. The majority of pages are in the 1957 to 1964 style, so these were the first ones worked on. These sheets (Figure2b) are characterized by a header on top consisting of the dates of the recorded data and a section on the bottom for any remarks. They contain 16 columns of data and five days worth of observations. Additionally, the times and the column headers are typed, the former of which will be utilized in identifying the rows of the table. The first type of sheets (Figure 2a) consist of 11 columns of data and six days of observations. Here the remarks are done for every row and contain codes. The key difference between this type and the others is that it lacks the temperature measurements corresponding to the Schaefer box, the time to wait between filling the box and counting the particles. The 1955 sheets (Figure 2c) are unique in that they span two pages and are typed. They contain the same information as the 1957 to 1964 sheets.

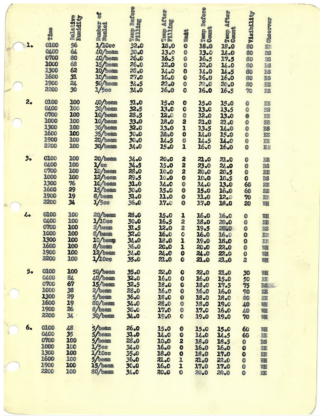


(a) Datasheet from 1948.

(b) Datasheet from 1968.



(c) Datasheet from 1955.



(d) Datasheet from 1955.

Figure 2: Examples of datasheets throughout the years.

5.2 Pre-processing

The majority of the techniques focus on transforming the image in order to correctly extract the characters. Binarization, blurring, dilation and erosion are all used to extract contours. Each step in the pre-processing relies on the correct extraction of contours. Blurring, dilation and erosion use a convolution to achieve their goals. Furthermore, the general structure of each pre-processing step is blurring $\rightarrow$ binarization $\rightarrow$ dilation $\rightarrow$ erosion $\rightarrow$ contour extraction.

5.2.1 Convolution

A convolution of a sub matrix over a matrix is crucial for many of the methods to work. Consider a $k \times k$ sub matrix or kernel. The base matrix has dimensions N$\times$N. Additional terms required are the stride and padding. The stride represents how many cells the kernel moves after each step. Padding p adds extra rows and columns to the base matrix in order to preserve its size after convoluting. The resulting matrix ends up with dimensions $(\mathrm{N}+2p-k+1) \times (\mathrm{N}+2p-k+1)$ assuming a stride length of 1. For example, let N=5, $k=3$ and $p=0$. The resulting matrix will be 3×3.

$$\mathrm{N} = \begin{matrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

$$\mathrm{kernel} = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Starting with N_{22} as the centre, overlay the kernel matrix and multiply each pair then sum all the products. This will give a value of 6 to go into the new matrix. Then move the kernel to centre on N_{23} and do the same calculations. Continue for the rest of the matrix to end with:

$$\begin{matrix} 6 & 5 & 6 \\ 6 & 5 & 6 \\ 6 & 6 & 6 \end{matrix}$$

The kernel need not be symmetrical in dimensions or in entries. This allows for great flexibility in how they are used as different kernels are used in each step of the pre-processing. Convolutions will also be used in character recognition.

5.2.2 Gaussian Blur

Gaussian blur is a smoothing technique that is often used before Otsu's binarization. It also has the effect of increasing the threshold level and allowing more text to be darkened. It can also be used to reduce noise in the image by smoothing over unwanted spots. This is done by a convolution matrix that is applied to the greyscale datasheet. The matrix replaces the current pixel with the weighted average of the neighbouring pixels with the areas closer to the centre being weighted more. The hyperparameters are the kernel size, representing the size of the neighbourhood and the sigma value representing the standard deviation. The formula to calculate the entries of the kernel matrix are done using the following formula.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\big(- \frac{x^2 + y^2}{2\sigma^2}\big)$$

σ represents the standard deviation, x and y represent the distance from the origin. The larger the standard deviation, the stronger the blur is. This can be observed in column two and column three in Figure 3. Increasing the size of the kernel has diminishing returns as the further away the pixel is from the centre, the less weight it is given. This can be seen in Figure 3 by observing the little difference between column two and column four.

Figure 3: The original cell is on the left, then image is blurred with kernel size 15×15 and $\sigma = 2$. The third image represents increasing σ to 30 and the final image increases the kernel size to 51×51.

5.2.3 Binarization

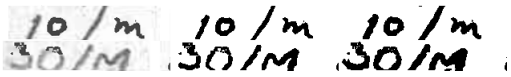


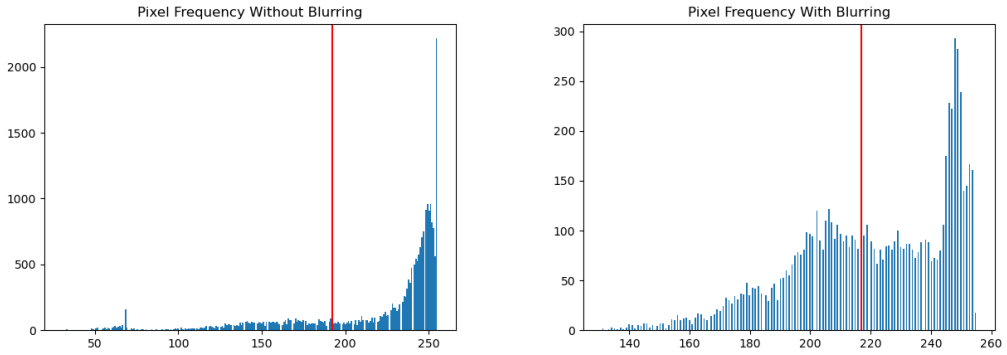
Figure 4: Column one is the original image, column two is binarized without Gaussian smoothing and column three is binarized after smoothing the image.

Binarization is converting the greyscale or colour image to binary black and white. It became clear that selecting a threshold by hand would be infeasible due to the amount of sheets and the fact that a different threshold level is optimal for each sheet. There are a few techniques that are helpful to overcome this. The first technique is to use an adaptive threshold that changes based on neighbouring pixels. This is useful in conditions of varying lighting where a single threshold value could cause large dark or white spaces in the binarized image. The datasheets are evenly illuminated so this technique was not necessary. Instead, Otsu's binarization was used. This is a global thresholding technique where the greyscale value of each pixel is recorded and considered as a data point similar to binary classification. The threshold is chosen by minimizing the intra-class variance. This is equivalent to maximizing the inter-class variance. Figure 5 shows how blurring affects the optimal value of thresholding. As seen in Figure 4 Otsu's binarization works very well to darken the text and whiten the blank space. The third column contains thicker text thicker and removes some gaps in letters. However, there are some cases where the text is too light and is mistaken for the background. It was deemed that this was worth the time savings as opposed to manual threshold selection.

The intra-class variance is defined as follows:

$$\sigma_\omega^2 = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$
$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$
$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

L is the bins in the histogram. Here $L = 256$ (number of values the pixels can take in greyscale). $\omega_{0,1}(t)$ are the probabilities of the respective class given threshold t and $p(i)$ is the probability that a pixel is in bin i. $\sigma_{0,1}^2(t)$ are the variances of the respective classes at threshold t.



(a) Histogram of pixel values without blurring. (b) Histogram smoothed compared to no blurring.

Figure 5: The red line represents the value chosen by Otsu's binarization. The value without blurring the image is 192. The value after blurring the image is 217.

5.2.4 Dilation and Erosion

Dilation and erosion both convolude a matrix over the image that either thickens the white space (dilation) or thins the white space (erosion). In a binary image, they are duals of each other meaning that dilation of the black pixels is the same as erosion of the white pixels. Dilation works by checking if there is a one in the neighbourhood and if changing the pixel to one if there is. Erosion works by convolving a matrix of ones over another binary matrix. If the first matrix "matches" the neighbourhood on the second matrix, the value in the centre is kept as one. If there is a zero in the neighbourhood, the value changes to a zero. A useful property is that given the kernels are the same size, applying an erosion then dilation or dilation then erosion will return to the original image. This is only true however if when the erosion step is first, it does not completely erase the image. This property will be used to help clean the character images by removing unwanted noise or artefacts. Another useful property is that the kernel need not be square. This allows dilation or erosion to be directional which will be used later to aid in finding contours.

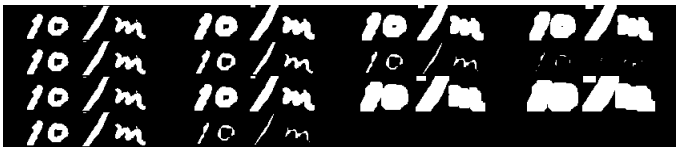


Figure 6: The first two rows show dilation and erosion with a 3×3 kernel and the second two rows show dilation and erosion with a 5×5 kernel.

5.3 Neural Networks

The main tool that is used to classify the characters will be a neural network. Within the neural network, there are many types of layers. A frequently used layer in character recognition is a convolution layer. This layer consists of learnable kernels or filters that are convolved over the input image. The model learns which kernels are activated when there is a feature at a position in the input. The pooling layer first "splits" the matrix into sub-matrices of a given size. Then within the blocks, the maximum value (in max pooling) is chosen used to create a new matrix. This new matrix is smaller than the original and thus cheaper computationally to compute. The fully connected layer has each neuron connected to every one in the previous layer. This is computationally heavy so is only done at the end of the network as it is required for classification. Take Figure 7 as an example to walk through. A single channel 28×28 matrix is the input matrix. Then 28 different 5×5 kernels are convoluted on the image to create 28 24×24 feature maps. This is known as a convolutional layer. Then a max pooling layer occurs in which each feature map is split into 6 4×4 sections. Then the maximum of each chunk is chosen and assembled to create 28 6×6 feature maps. Finally, there are two fully connected layers with the final layer consisting of ten values.

Another component of the network is that is the ReLU activation function. This layer applies the function $\max(0, x)$ which replaces negative values in the matrix with 0 where x is an entry in the matrix. After every convolution layer, ReLU is applied to help with computation. The final part needed to get an output is to interpret the last layer. To do this, a log softmax transformation(Equation 1) was applied to the outputs which mimics the properties of probability and allows for the prediction of characters. The output of the model, $\mathbf{x}$, is a vector of size determined by the number of classes to be classified. The entries x_i represent the strength of the prediction an image has to class i. However, without the log softmax transformation, these results are difficult to interpret because they are difficult to assign a confidence score to. In the toy example, $i, j \in \{1, ..., 10\}$ and the model is able to classify ten objects. The object that the model predicts is whichever index corresponds to the greatest log

softmax value. As a note the softmax$(x) \in [0, 1]$ and log softmax$(x) \in (-\infty, 0]$. Exponentiating the value given by log softmax can be thought of the model's confidence that the input matrix belongs to that class.

$$\log(\text{Softmax}(x_i)) = \log\left(\frac{\exp(x_i)}{\Sigma_j \exp(x_j)}\right) \tag{1}$$

The final part of the neural network is the training of it. For this a loss function and optimization algorithm must be chosen. The loss function is minimized during training and the negative log likelihood loss was used. This works well for classification problems with many classes. In Equation 2, y represents the true class label and $\hat{y}$ represents the predicted class label. A simple way to minimize this function is gradient decent. Gradient descent works by taking steps in the opposite direction of the gradient to reach a local or global minimum. More advanced techniques can also be used and decrease the number of training epochs needed to reach the optimum.

$$\text{Negative Log Likelihood} = -\sum_{i=1}^{n} \left(y_i \log \hat{y}_{\theta,i} + (1 - y_i) \log(1 - \hat{y}_{\theta,i}) \right) \tag{2}$$

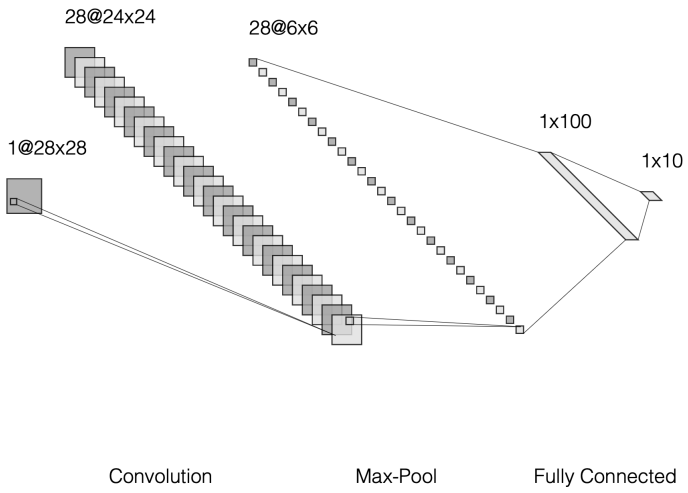


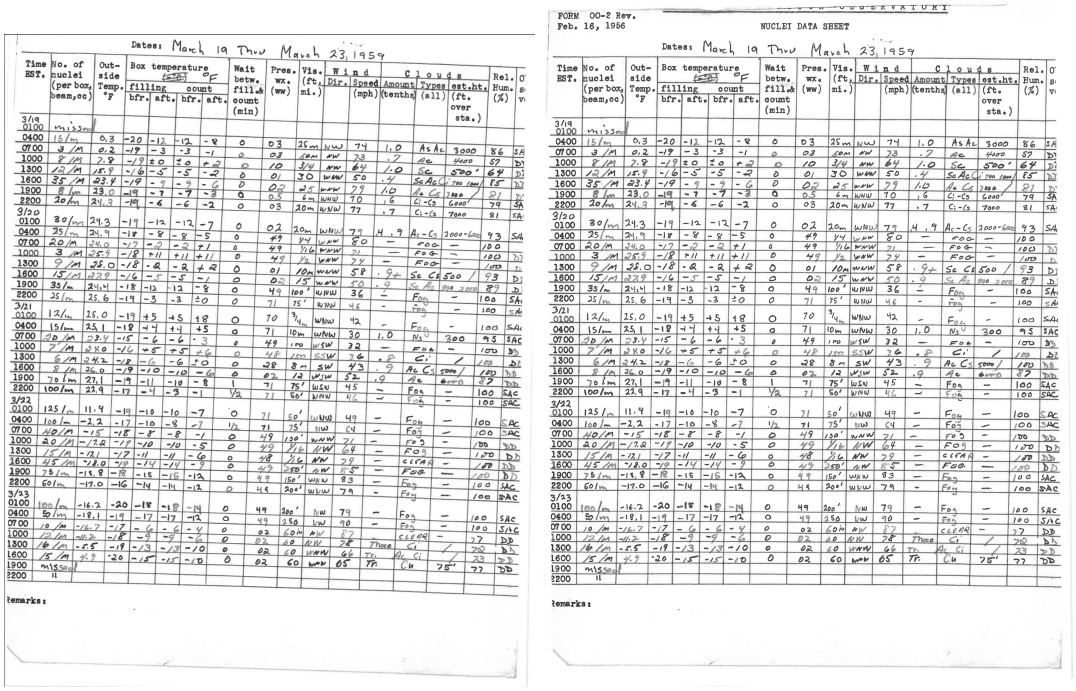
Figure 7: Toy example showing the types of layers that can be used in a neural network.

6 Pre-processing Results

The techniques in the previous section are used in each step in the pre-processing of the Mt. Washington datasheets.

6.1 Page Rotation

Many of the pages are nearly vertically aligned however some are very rotated. The assumption in this step is that the pages are not rotated more than 90 deg clockwise or counterclockwise. The pages are rotated by blurring the image using a Gaussian filter and then dilating the image to create rectangular blocks. Then using the largest contour of the bocks, the "horizontal" axis of the bounding box will be used as the line to calculate how rotated the image is. The image is then rotated in the opposite direction and becomes unrotated.

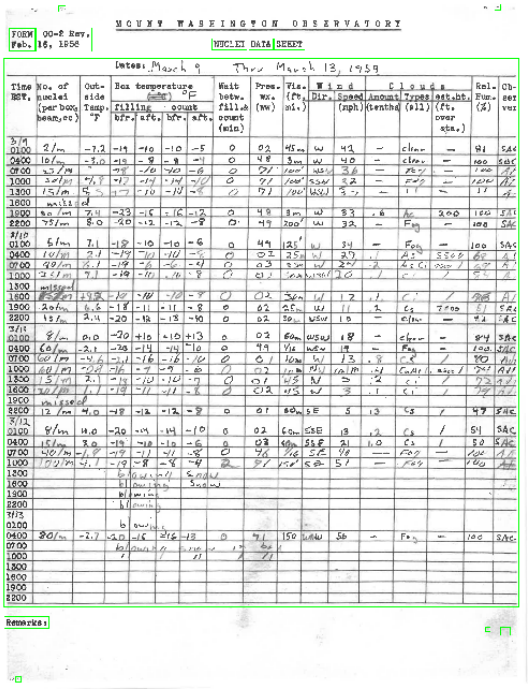


(a) Original sheet. (b) Rotated sheet.

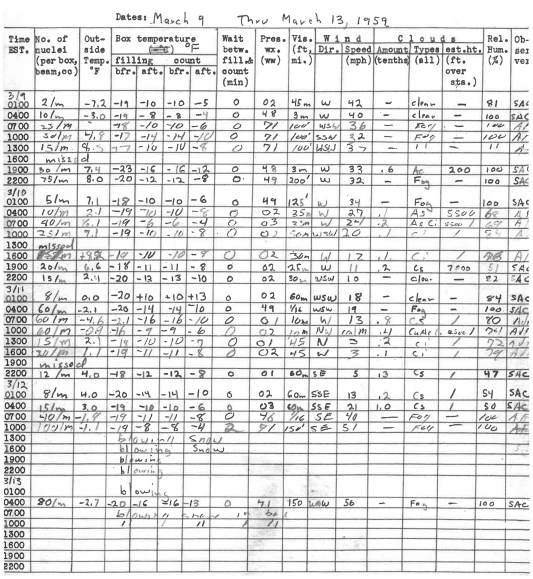
Figure 8: Rotating the datasheet.

6.2 Table Extraction

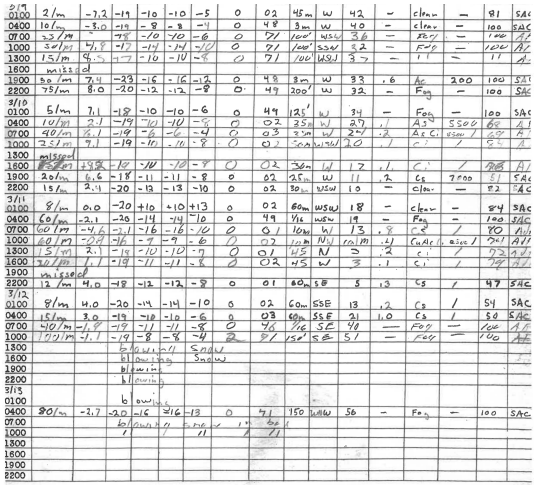
The pages contain some whitespace and repeated information across each one that is not needed to be detected and recognized. Specifically the header and the remarks are removed from the image leaving just the table. This is done by finding the largest contour and extracting it. Additionally, in early tests, the headers of the table were interfering with the number of rows detected so they are removed. Unfortunately this is currently done by hard coding their height and then cropping the table down.



(a) Found contours noted by the green lines.



(b) Kept the largest contour removing the remarks and title of the page.

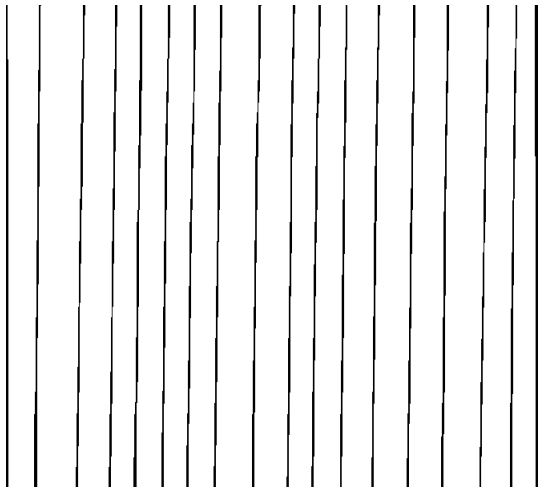


(c) Table without headers.

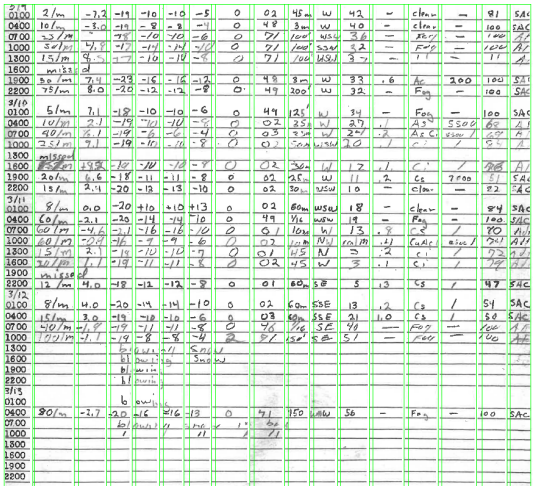
Figure 9: Cropped table to only contain handwritten the data.

6.3 Word Isolation

One challenge encountered was preserving the structure of the table when extracting the words and characters. This is an important part as the main goal will be to analyse the resultant time-series, so the data must be reconstructed correctly. This is the reason that the header was removed in the previous section. Additionally, it was considered to remove the times from the left hand of the table as they are not handwritten, but they are kept in as they are useful in the creation of the rows of data. Another way to keep the structure was to use the table lines to help determine where the rows and columns are. Some pages contain empty columns and/or rows. Initially, the method used to extract columns would not detect these columns as there was no text to dilate and extract the contour. By removing the vertical lines, gaps form between each column. When a dilation is applied with a vertical kernel, the columns become solid and the contours are easily extracted.

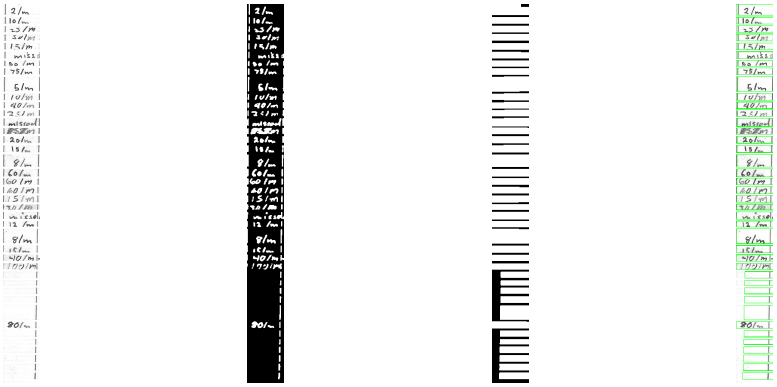


(a) Columns Dilated.



(b) Contours overlaid on the table with no vertical lines.

The next step is to extract the rows of the individual columns. This corresponds to extracting one cell of the table. Similarly, for the columns, the horizontal table lines are removed to create gaps between the rows. Then a dilation with a horizontal kernel is applied and the contours are extracted. The vertical lines must remain in order for blank cells to have a contour and thus not be skipped when reconstructing the table. The end result of this is an image of the individual cell with its location easily accessible.



(a) Column showing particle counts.

(b) Converted to binary.

(c) Dilated to detect rows.

(d) Contours to be extracted.

Figure 11: Steps taken to extract the cells or rows from individual columns.

6.4 Segmentation

Two segmentation techniques were considered to extract the characters. The first technique finds the bounding boxes around the characters and is known as contour extraction. The second technique known as histogram segmentation separates the characters by considering how many character pixels are in each column.

Finding contours can be expanded to extract handwritten characters with some initial steps. For the contours to accurately trace the individual letters, the letters need to be spaced out and not connected. This is not true for many of the words that are present in the dataset. Thus, in order to use contour based segmentation, some steps must be taken to first separate the characters. The first step is to erode the word image. This decreases the thickness of the letters and increases the distance between already separated letters. If the letters become well separated after this step, then the contour algorithm easily detects the correct area. However, if there is too much erosion, there is a chance that the contour boundary will not cover the whole letter in the original image during extraction or.

As outlined by the authors of [12], the first step to histogram based extraction is to thin the text to one pixel wide. Then the number of filled pixels is counted in each column. If there are zero pixels, the column is considered as a space between characters. The authors take this a step further and also consider columns containing one pixel as potential segmentation columns. This provided good performance on cursive words as the letters are linked with a one pixel thick line. Additionally, the letters are often swooped and so the columns containing letters have more than one pixel filled. This rule was relaxed to only consider segmentations at columns with zero pixels as the numbers contain many one pixel columns.

Some examples are present in Figure 12. Neither method achieves perfect segmentation and both are stronger in certain cases. Histogram segmentation works well in cases like the third image where the 2 has a gap between the upper and lower part. Contour extraction detects this as two separate characters but due to the histogram method looking at only one dimension, the whole character is extracted. Comparing the first image, contour extraction performs much better. Specifically, the zeros in the histogram segmentation will be segmented into multiple "half" zeros. This is caused by the gaps in the character lines widening as it is thinned to one pixel. Contour segmentation has no problem because the character lines are thicker and connected. When the characters are well spaced and the lines are full, both methods successfully extract the characters as seen in the fourth image. Contour based extraction was chosen as it performed similarly to histogram based extraction with each method having differing strengths.

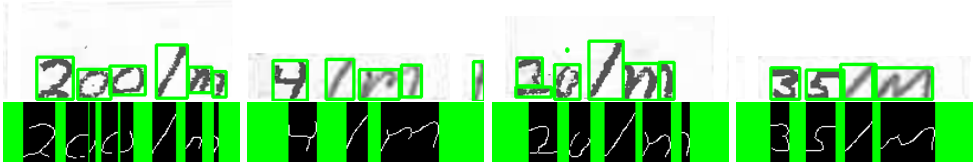


Figure 12: Comparison of the two techniques. Contour based on top and histogram based on the bottom. The green represents the potential segmentation area for histogram based segmentation and the bounding boxes in contour based segmentation.

6.5 Character Transformations

6.5.1 Dimension Resizing

The contours are different sized images and must be scaled before entering the neural network. The dimensions chosen for this were 28×28 in order to maintain compatibility with the MNIST dataset. In the event that the detected contour is larger than 28×28 in both axis, the image is scaled down using bicubic interpolation. It is hoped that this yields the least amount of distortion. If either or both of the axis are less in size than 28, blank columns and/or rows are added to preserve the size of the detected character. It is worth noting that scaling can not be used as it would change the size of the character too much. This is most notable for the characters like 1,- and . as they will be stretched when scaled to be unrecognizable. If one axis is larger than 28 and the other less, the following occurs: firstly, the smaller axis has blank columns or rows added to it to make it 28. Then the larger axis is scaled down using same technique as before to make the image 28×28. This limits the amount of scaling necessary and the character only becomes squished in one dimension rather than squished in one and stretched in another.

6.5.2 Binarization

After the images are all the correct dimensions, they must be converted to binary. This step is done in order to standardize the text (white) and background (black) colour. These correspond to a value of 255 and 0 respectively. The method for selecting the threshold is Otsu's method. This works well because the text darkness in each image is not always the same as the other images, so a fixed threshold cannot be used.

6.5.3 Skeletonization

The final step of preprocessing is to skeletonize the characters. The goal of this is to thin the characters down and make them more distinct. This was done by dilation and erosion. The characters were first dilated to remove any unwanted gaps then eroded to thin the lines.

6.6 Data Augmentation

Due to the time-consuming nature of hand labelling the data, the training set will be augmented to increase its size. As the text is written by pencil and/or pen, the width of the letters can vary. The first augmentation technique will attempt to account for this by creating two extra images. One will use erosion to thin the character down and the other will use dilation to thicken the lines. Additionally, some characters contain gaps in their strokes so some columns and rows were converted to 0 to give the appearance of a broken character. One final augmentation converts rows and columns to 1 giving the illusion that the character has been written over or scratched out. The inspiration for these techniques was taken from [13] in which the authors use similar augmentation to bolster their training data. Another technique is to introduce "salt and pepper" noise before binarizing the image. This simulates random artefacts or noise in the scanned datasheets, but it was deemed unrepresentative as the sheets were largely noiseless.

Figure 13: Example of augmenting a letter m.

7 Model Development

7.1 Datasets

Three different datasets were used to help digitize Mt. Washington data. The first dataset used was the MNIST dataset. This consists of 70000 handwritten digits (10 classes) and was used primarily to check if the model's architecture is capable of classification. The next dataset used was hand labelled from the Mt. Washington consisting of 830 training characters and 200 testing characters covering 18 classes.

7.2 Character Recognition Models

The architecture that model one follows is three convolution, ReLU, max pooling blocks into a fully connected layer outputting 10 values corresponding to the 10 digits. Models two and three use a similar model to this however there is an additional block before the fully connected layer.

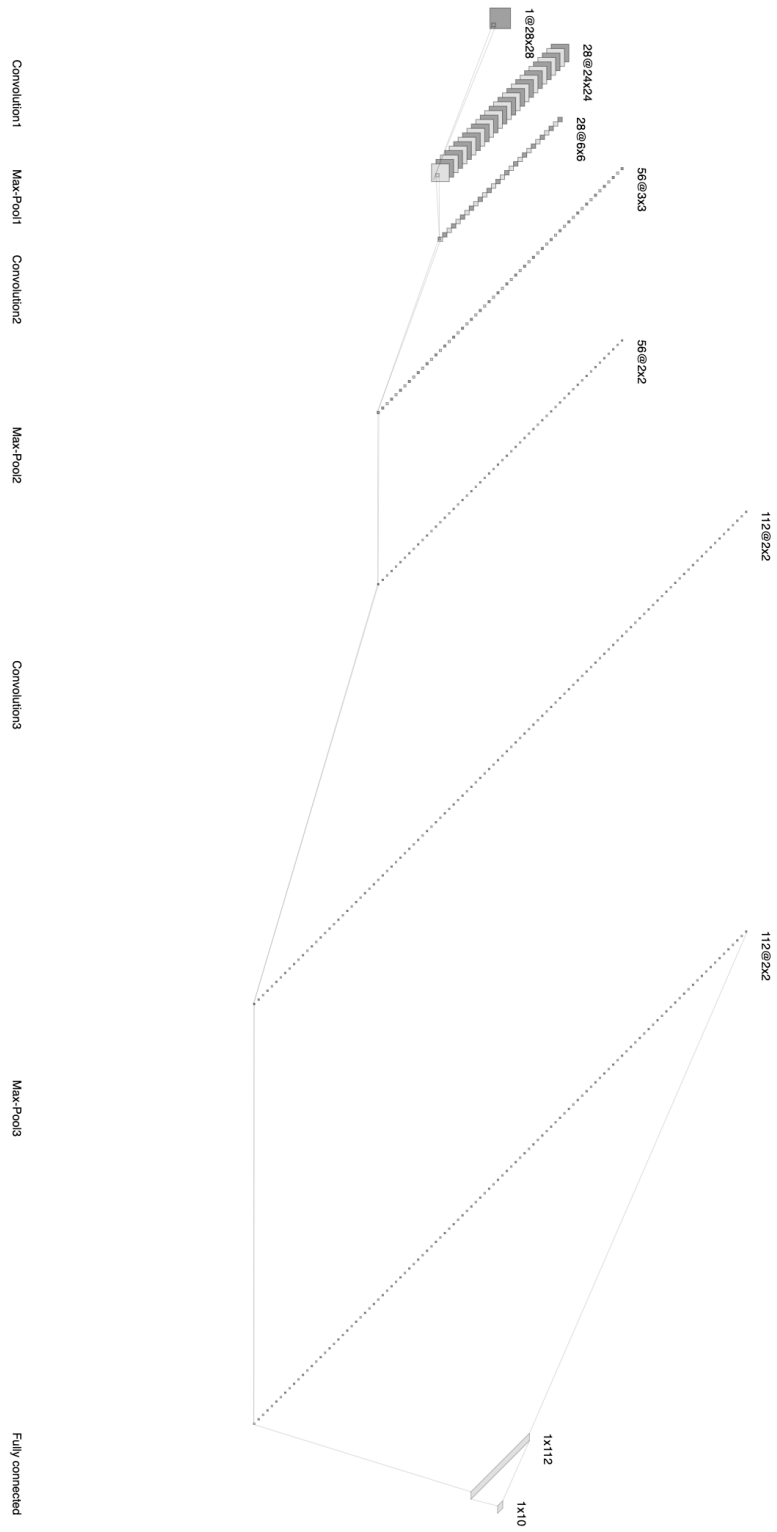


Figure 14: Architecture for model 1.

7.2.1 Model 1

The first model was developed using the MNIST dataset. This dataset consists of 70000 handwritten digits taken from zip codes. A softmax transformation is applied after the final layer in order to convert the output into a value between 0 and 1. This allows the use of the negative log likelihood loss function. The dataset is normalized by subtracting the dataset mean value and then dividing by the standard deviation. The goal of this step is to speed up the training of the model. The accuracy of this model was 98.28% indicating that the architecture is sufficient for character recognition. This corresponds to a clean diagonal confusion matrix seen in Figure 15. Additionally, as seen in Figure 16, this accuracy was achieved in only 20 training epochs. The model and weights were then saved in order to use with the Mt. Washington dataset.

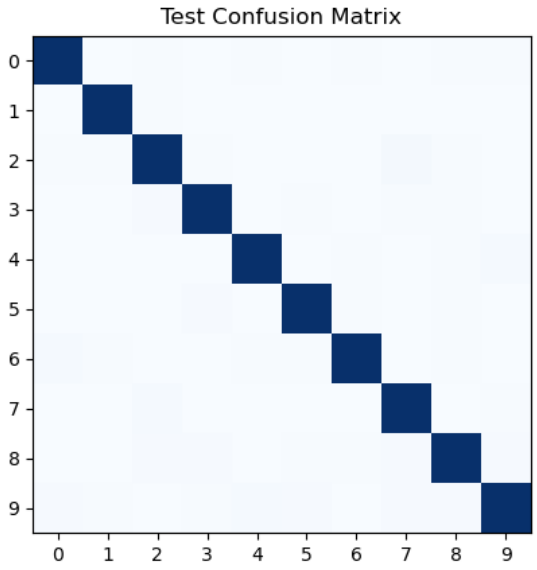


Figure 15: Confusion matrix shows very little miss-classified digits as expected due to the 98.28% accuracy.

Following on from the success on the MNIST dataset, the Mt. Washington dataset was fed through the model using weights trained from the MNIST dataset. An accuracy of 22.6% was obtained. This is visually represented in Figure 17 where the majority of predicted values were 8's and not many values were along the diagonal. This is most likely due to the fact that the MNIST dataset consists of very clean and clear digits whereas the Mt. Washington on the real data is not as clean. Additionally, the real dataset consists of images that contain broken digits or only half of a digit as seen in Figure 18.

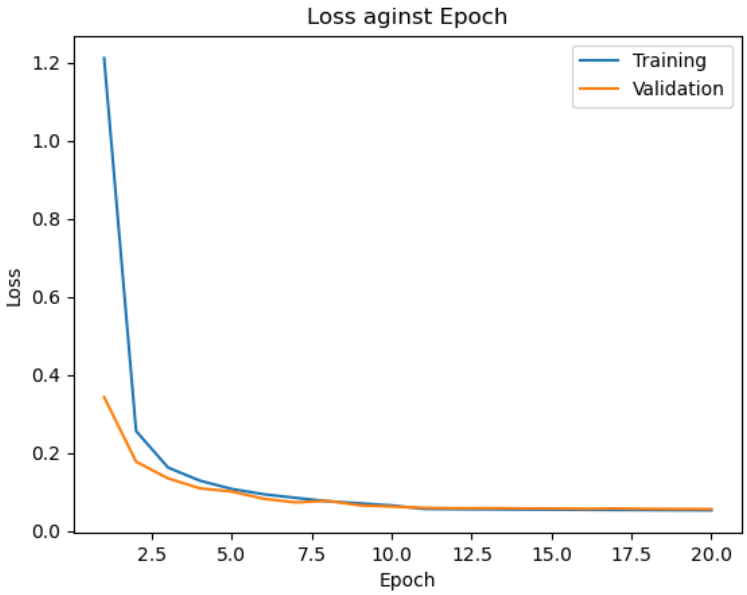


Figure 16: Loss during training.

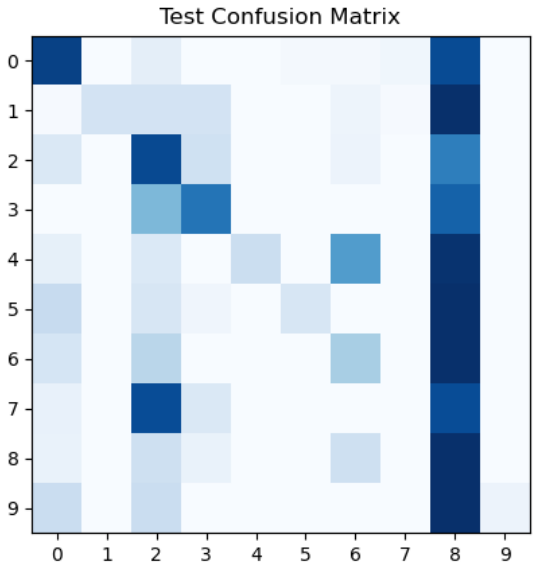


Figure 17: MNIST weighted model provides only 22.6% accuracy when Mt. Washington data is fed through.

Figure 18: A 5 and 0 that are incomplete due to either a too low of threshold in binarization or part of the image removed because it was written over a table line.

7.2.2 Model 2

The architecture presented in Figure 14 provided excellent(98.28%) results on the MNIST dataset, so it was used as a starting point to create a model for the Mt. Washington data. The architecture was adjusted slightly to add in an additional convolutional block and the number of classes was changed to 18 as letters and special characters are added to the training set. This results in an accuracy of 76.44%. There are two reasons for the lower accuracy compared to the MNIST data tested on the MNIST trained model. Firstly, the amount of training data greatly reduced from 60000 to 830. The second reason is that the number of classes to classify increased from 10 to 17 which is inherently a more difficult problem to solve. The confusion matrix (Figure 19) shows that there is a clear diagonal meaning that characters are correctly identified. It also shows that there are certain characters that are commonly misclassified such as predicting an m when the true character is a w. Some other notable misclassifications are 3 and 5, and 1,2 being misclassified as 7. Additionally, as seen in Figure 20, this model took over 300 epochs to train which is over ten times as many as Model 1.

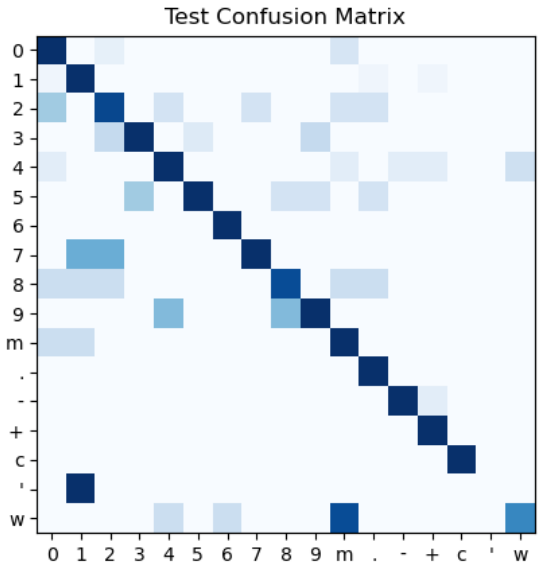


Figure 19: Confusion matrix showing the results of testing the Mt. Washington test data on the model trained on Mt. Washington data.

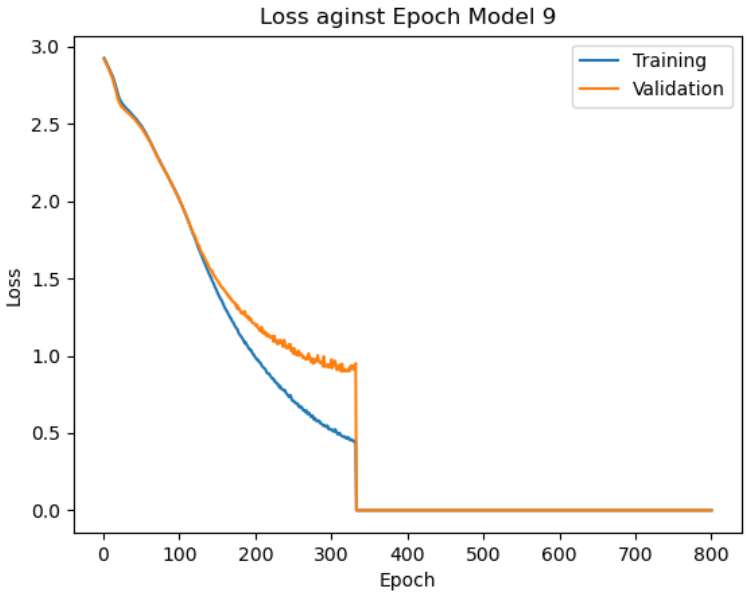


Figure 20: Takes around 300 epochs to train before stopping due to a risk of the overfitting characterized by the training and test loss diverging.

7.2.3 Model 3

The amount of classes to be classified can not be lowered so the only other options to improve performance are to increase the training data or change the model architecture. The architecture is more difficult to change, so the training set will be increased. In order to increase the amount of training data, it is augmented with the techniques outlined earlier. This increases the size and variability of the training set creating a more rounded set. The size of the training set increased from 830 to 4151. This is still notably smaller than that of MNIST training set but is a vast improvement. The alternative to augmentation is to label more images, but this is much more time-consuming. An accuracy of 80.77% was recorded which is 4% higher than model 2's. As seen in the confusion matrix (Figure 21), the misclassification of w improves and there is overall fewer misclassifications. Moreover, the model trains in much fewer epochs (around a third) than model 2 despite the larger training set. This shows that augmentation can not only improve accuracy by increasing variability but also improve training time by reducing the number of training epochs (Figure 22).

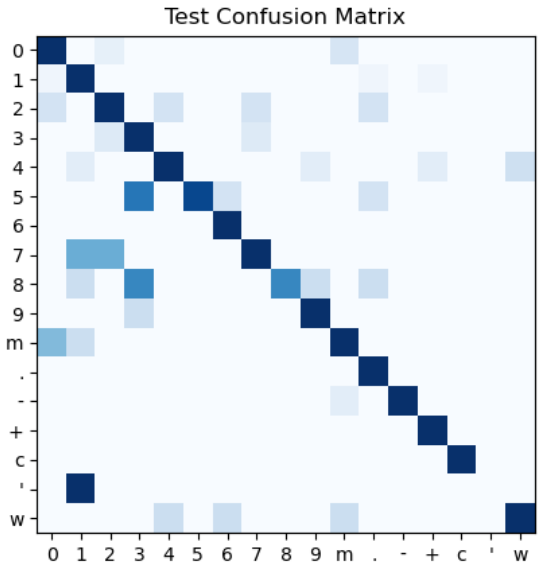


Figure 21: Similar matrix to model 2 with most of the improvement coming from correctly classifying m and w.

Since model 3 has the best accuracy, it was chosen to test how well the sheets could be digitized.

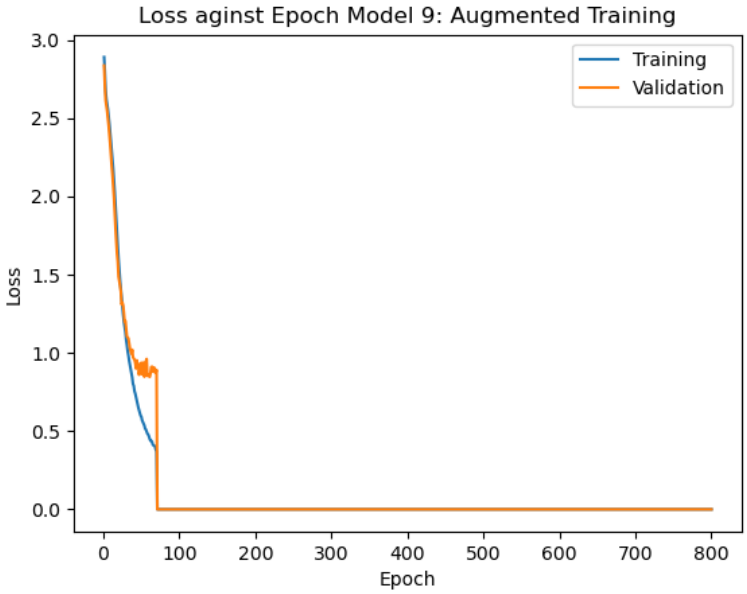


Figure 22: Around 80 training epochs before stopping due to the risk of overfitting.

7.3 Model Comparisons

The addition of augmented data sped up the training time and created a more accurate model compared to the second model. The first model has such good performance primarily due to the large amount of data available and the high quality of the data.

Table 1: Summary of the model performances.

Model Architecture	Training Dataset	Testing Dataset	Accuracy	Training Epochs
Model 1	MNIST	MNIST	98.28%	20
Model 1	MNIST	Mt. Wash.	22.60%	-
Model 2	Mt. Wash.	Mt. Wash.	76.44%	320
Model 3	Mt. Wash.	Mt. Wash.	80.77%	80

8 Results

Using the model trained with augmented data (model 3), the characters from each datasheet were classified. The performance of the model is difficult to quantify because the contours do not extract the correct amount of characters. Traditional accuracy measures like character error rate (CER) can not be used. Character error rate measures the proportion of correct characters to total characters. CER would be artificially lowered due to the over segmentation. Instead, word error rate (WER) can be used. It measures the proportion of words or in this case cells that are correct. This means the cell must have the right number of characters and the characters must be identified correctly. The WER for one sheet was 6.25% which is very low but expected. The majority of this accuracy comes from correctly identifying where no data is present. The rest likely comes from single character cells as they were more likely to be segmented correctly. See the appendix for information on sample results.

Overall, it is clear that a better approach to character segmentation is needed. In addition, a better trained neural network is needed as the 80% is not sufficient to identify characters even if they are segmented correctly.

9 Future Work

There are a few components that can be improved upon. The first one is increasing the training data size. This can be done by hand labelling more data or by including more augmented data. Two techniques that could be promising are to introduce rotation in the characters and to cover parts of the characters with a "block". Rotating the characters represents different writers slants which would be useful as there are many writers throughout the years in the Mt. Washington data. Introducing a "block" to cover part of the character would be useful by simulating smudges or other objects that can obscure the characters. Hand labelling is very time-consuming but is the best way to increase performance of the model. It is possible to create a program that shows a single image on the screen and a text box to input the character in the image. Once a character is input, a new image pops up. This would drastically reduce the time to label the characters. Both of these methods will improve the accuracy of the neural networks as there is still accuracy to gain (80.77% with a theoretical maximum of 98%).

If the accuracy of the model is improved the next step is to improve the character extraction. Currently, the method over segments characters leading to too many characters in each cell. One option is to avoid the problem all together and perform word recognition instead. This shifts the feature (character) extraction to a neural network but is notably more difficult and requires much more training data. Another option is to explore different options to prepare the characters for segmentation or improve on histogram segmentation.

From start to finish, classifying the characters for a year's worth of data takes very long and there numerous optimizations that can be done to the code. The most time-consuming part is the pre-processing step taking over 20 minutes to extract every character. The other part of the code that can be improved is the post-processing as there are many writing to disk operations that slow it down.

References

[1] S. M. Burrows, C. S. McCluskey, G. Cornwell, I. Steinke, K. Zhang, B. Zhao, M. Zawadowicz, A. Raman, G. Kulkarni, S. China, A. Zelenyuk, and P. J. DeMott, "Ice-nucleating particles that impact clouds and climate: Observational and modeling research needs," *Reviews of Geophysics*, vol. 60, no. 2, p. e2021RG000745, 2022, e2021RG000745 2021RG000745. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021RG000745

[2] P. J. DeMott, A. J. Prenni, X. Liu, S. M. Kreidenweis, M. D. Petters, C. H. Twohy, M. S. Richardson, T. Eidhammer, and D. C. Rogers, "Predicting global atmospheric ice nuclei distributions and their impacts on climate," *Proceedings of the National Academy of Sciences*, vol. 107, no. 25, pp. 11 217–11 222, 2010. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0910818107

[3] W. Cantrell and A. Heymsfield, "Production of ice in tropospheric clouds: A review," *Bulletin of the American Meteorological Society*, vol. 86, no. 6, pp. 795 – 808, 2005. [Online]. Available: https://journals.ametsoc.org/view/journals/bams/86/6/bams-86-6-795.xml

[4] W. H. Knap, P. Stammes, and R. B. Koelemeijer, "Discriminating between water and ice clouds using near-infrared aviris measurements," in *Proceedings of the 9th AVIRIS Earth Science and Applications Workshop, NASA Jet Propulsion Laboratory, Pasadena, USA*, 2000, pp. 23–25.

[5] V. J. Schaefer, "The concentration of ice nuclei at the summit of mt. washington," 1952.

[6] ——, "The concentration of ice nuclei in air passing the summit of mt. washington," 1954.

[7] E. K. Bigg, "Meteorology," *Science Progress (1933-)*, vol. 49, no. 195, pp. 458–475, 1961. [Online]. Available: http://www.jstor.org/stable/43425202

[8] G. Vali, "Atmospheric ice nucleation-a review," *J. Rech. Atmos.*, vol. 19, pp. 105–115, 1985.

[9] R. Manmatha and N. Srimal, "Scale space technique for word segmentation in handwritten manuscripts," 10 2000.

[10] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3169

[11] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (cnn)," *Sensors*, vol. 20, no. 12, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/12/3344

[12] A. Choudhary, R. Rishi, and S. Ahlawat, "A new character segmentation approach for off-line cursive handwritten words," *Procedia Computer Science*, vol. 17, pp. 88–95, 2013.

[13] K. Chaudhary and R. Bali, "Easter2.0: Improving convolutional models for handwritten text recognition," 2022. [Online]. Available: https://arxiv.org/abs/2205.14879

Appendix

The code for each step is provided on GitHub through the following link:
https://github.com/Denver972/Handwriting-recognition.git Additionally, sample results are included in the results folder.